Virtuoso Space-based Router Command Reference

Product Version ICADVM20.1
October 2020

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

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

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

11

Route Commands

Preface

This manual describes the Tcl commands used with Virtuoso® Space-based Router. Tcl is a general purpose, popular scripting language that can be used to automate and control Space-based Router. Tcl reference books are widely available. In addition to the commands described in this manual, the Tcl Core commands are supported by the built-in Tcl interpreter. Tcl commands can be issued interactively via the Space-based Router Command line and by using scripts.

This reference is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- Tcl/Tk programming
- The Virtuoso design environment technology file.

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions
- Identifiers Used to Denote Data Types

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node and methodologies (for example, ICADVM20.1) releases.

Label	Meaning
(ICADVM20.1 Only)	Features supported only in the ICADVM20.1 advanced nodes and advanced methodologies release.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

For information about licensing for Virtuoso Space-based Router, see Virtuoso Space-based Router Tokens in the Virtuoso Software Licensing and Configuration User Guide.

Related Documentation

What's New and KPNS

- *Virtuoso Space-based Router What's New*
- *Virtuoso Space-based Router Known Problems and Solutions*

Technology Information

- *Virtuoso Technology Data User Guide*
- *Virtuoso Technology Data ASCII Files Reference*
- *Virtuoso Technology Data SKILL Reference*
- *Virtuoso Technology Data Constraints Reference*

Virtuoso Tools

IC6.1.8 Only

- *Virtuoso Layout Suite L User Guide*

- *Virtuoso Layout Suite XL User Guide*
- *Virtuoso Layout Suite GXL Reference*

ICADVM20.1 Only

- *Virtuoso Layout Viewer User Guide*
- *Virtuoso Layout Suite XL: Basic Editing User Guide*
- *Virtuoso Layout Suite XL: Connectivity Driven Editing Guide*
- *Virtuoso Layout Suite EXL Reference*
- *Virtuoso Concurrent Layout User Guide*
- *Virtuoso Design Planner User Guide*
- *Virtuoso Multi-Patterning Technology User Guide*
- *Virtuoso Placer User Guide*
- *Virtuoso Simulation Driven Interactive Routing User Guide*
- *Virtuoso Width Spacing Patterns User Guide*
- *Virtuoso RF Solution Guide*
- *Virtuoso Electromagnetic Solver Assistant User Guide*

IC6.1.8 and ICADVM20.1

- *Virtuoso Abstract Generator User Guide*
- *Virtuoso Custom Digital Placer User Guide*
- *Virtuoso Design Rule Driven Editing User Guide*
- *Virtuoso Electrically Aware Design Flow Guide*
- *Virtuoso Floorplanner User Guide*
- *Virtuoso Fluid Guard Ring User Guide*
- *Virtuoso Interactive and Assisted Routing User Guide*
- *Virtuoso Layout Suite SKILL Reference*
- *Virtuoso Module Generator User Guide*

- *Virtuoso Parameterized Cell Reference*
- *Virtuoso Pegasus Interactive User Guide*
- *Virtuoso Space-based Router Constraint Reference*
- *Virtuoso Space-based Router User Guide*
- *Virtuoso Symbolic Placement of Devices User Guide*
- *Virtuoso Voltage Dependent Rules Flow Guide*

Relative Object Design and Inherited Connections

- *Virtuoso Relative Object Design User Guide*
- *Virtuoso Schematic Editor L User Guide*

Additional Learning Resources

Video Library

The Video Library on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see Virtuoso Videos.

Rapid Adoption Kits

Cadence provides a number of Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

To explore the full range of training courses provided by Cadence in your region, visit Cadence Training or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text

Indicates names of manuals, menu commands, buttons, and fields.

A string of italics separated by dashes represent the graphical interface version of a command. For example, *File – Open* indicates that you must choose the *File* menu, followed by the *Open* command.

`text`

Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.

`z_argument`

Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.

For more information on argument values, refer to Identifiers Used to Denote Data Types.

|

Separates a choice of options.

`{ }`

Encloses a list of choices, separated by vertical bars, from which you **must** choose one.

`[ ]`

Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.

{ }

Indicates braces that must be entered with the command syntax.

In the following example, you must type the braces:

```
command arg1 { x y }
```

...

Indicates the following:

- In the GUI, a form will be opened if you choose an option ending with three periods.
- In a Tcl command, you can repeat the previous argument.

`-argName` *`s_arg`*

Denotes a *key argument.* The hyphen and argument name must be typed as they appear in the syntax, followed by the argument value, if required.

#

Precedes comments in command files.

If a command-line is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

Identifiers Used to Denote Data Types

Data type identifiers are used to indicate the type of value required by an API argument. These data types are denoted by a single letter that is prefixed to the argument label and is separated from the argument by an underscore; for example, *s* is the data type in *s_viewName*. Data types and underscores are used only as identifiers; they must not be typed when specifying the argument in a function.

Data Type	Prefix
Internal Object	`d`
Real number	`f`
Integer	`i`
Character String (text)	`s`

Internal Object

In the command syntax, internal objects are represented as follows:

Internal Object	Variable
Object identifier	*d_ctuObj*
Inspector identifier	*d_inspObj*
OpenAccess cellview identifier	*d_oaObj*
Set object identifier	*d_setObj*

Character String

Strings are sequences of characters, for example, `"123"` or `abc`. Strings with embedded spaces must be enclosed in double quotes.

1

File Commands

This chapter describes the File commands.

Note: Tcl commands in this chapter are available when running Virtuoso® Space-based Router except as noted in the description for the command.

The following sections support high capacity loading and saving of GDSII and SEMI® Open Artwork System Interchange Standard (OASIS™) stream files:

- Stream/OASIS-to-OpenAccess Layer Map File on page 68
- read_stream on page 70
- write_stream on page 76
- write_stream_by_net on page 79
- write_stream_by_set on page 81

Other file commands are presented in alphabetical order:

- cellview_exists on page 33
- checkpoint on page 34
- choose_file on page 35
- close_db on page 36
- exit_no_save on page 37
- export_jpeg on page 38
- export_svg on page 40
- list_cellviews on page 41
- new_cellview on page 42
- print_window on page 43

cellview_exists

```
cellview_exists
     -lib s_libName
     [ -cell s_cellName [ -view s_viewName ] ]
```

Checks for the existence of a library, cell or cellview.

Arguments

`-cell` *`s_cellName`*	Specifies the name of the cell. If `-view` is not specified, a match occurs if the given cell exists in the given library.
`-lib` *`s_libName`*	Specifies the name of the library. If this is the only argument given, a match occurs if the given library exists.
`-view` *`s_viewName`*	Specifies the name of the view. A match occurs if a cellview exists with the given library, cell and view names.

Value Returned

`0`	A cellview exists with the given argument values.
`-1`	A cellview does not exist with the given argument values.

Example

The following example checks for the existence of the `myLib` library.

```
cellview_exists -lib myLib
```

The following example checks for the existence of the `myLib/myCell/layout` cellview.

```
cellview_exists -lib myLib -cell myCell -view layout
```

Related Information

Tcl Commands	get_occurrence

checkpoint

```
checkpoint
```

(Virtuoso Routing IDE only) Synchronizes the Virtuoso Layout Editor database with the current Space-based Router and Chip Optimizer database and sets a checkpoint in the Virtuoso Layout Editor transaction history that records the routing changes in Space-based Router and Chip Optimizer. When you return to Virtuoso Layout Editor, checkpoints can be undone in reverse order.

Arguments

None

choose_file

```
choose_file
```

Opens the Choose File dialog that lets you browse the hierarchy and select a file. The full path to the file is returned by the command. You can use this command in a Tcl script to interactively choose a file and then use the returned string as the input for another command.

Arguments

None

Value Returned

s_fileName	Returns the full path to the selected file.
-1	No file was chosen.

Example

The following command in a Tcl script lets you interactively choose an annotations file to open.

```
read_annotations -file [choose_file ]
```

close_db

```
close_db
     [ -lib s_libName -cell s_cellName -view s_viewName | -window_id i_windowID
     | -cell_view_id d_oaObj ]
     [ -no_prompt ]
```

Closes the specified window or cellview. If no argument is given, the current artwork window is closed. If Space-based Router and Chip Optimizer detects that a change was made in the database and not saved, the Save on Exit dialog appears, allowing you to save data before closing the database.

Arguments

-cell_view_id d_oaObj	Specifies the object identifier for the cellview to close.
-lib s_libName -cell s_cellName -view s_viewName	
	Specifies the cellview to close. All windows displaying the given cellview are closed.
-no_prompt	Forces the database to be closed without saving.
-window_id i_windowID	Specifies the window to close.

Example

The following example closes the cellview in the active window.

```
close_db
```

The following example closes all windows displaying the given cellview `mylib/NAND/abstract`.

```
close_db -lib myLib -cell NAND -view abstract
```

Related Information

Tcl Commands	read_db write_db
Menu Command	*File—Close*

exit_no_save

```
exit_no_save
```

Closes the open cellviews and shuts down the application without attempting to save changes. This command is equivalent to `quit -no_save`.

Arguments

None

Related Information

Tcl Commands	quit
Menu Commands	*File—Exit*

export_jpeg

```
export_jpeg
     [ -file s_fileName ]
     [ -imageQuality i_ratio ]
     [ -jpeg | -png ]
```

Creates a graphics file for the active window. The image quality (i.e., compression ratio) can optionally be set. If no arguments are specified, the Save JPEG dialog box appears, prompting you to specify the output image file to create using the default image quality and the current active window.

Important

This command cannot be used in batch/non-graphics mode.

Arguments

`-file` *`s_fileName`*	Specifies the name of the graphics file to create.
`-imageQuality` *`i_ratio`*	Specifies the image quality/compression ratio. The lower the number, the lower the quality (and the higher the compression). Zero (0) specifies maximum compression (which seriously degrades quality), 100 specifies the maximum quality, -1 specifies the default ratio (currently set to 100). Default: -1
`-jpeg | -png`	Specifies the graphics file format as JPEG or PNG. Default: JPEG

Example

The following example generates the JPEG file `newEdits.jpg` with an image quality of 90.

```
export_jpeg -file newEdits.jpg -imageQuality 90 -jpeg
```

Related Information

Menu Command	*File—Export JPEG…*

export_svg

```
export_svg
     [ -file s_fileName ]
```

Creates a Scalable Vector Graphics (SVG) file for the active window. Alpha blending is not supported.

Important

This command cannot be used in batch/non-graphics mode.

Arguments

`-file` *`s_fileName`*	Specifies the name of the SVG file to create. If this argument is given, the Save SVG dialog box appears, prompting you to specify the output image file to create.

Example

The following example generates the SVG file `mydesign.svg`.

```
export_svg -file mydesign.svg
```

Related Information

Tcl Command	export_jpeg
Menu Command	*File—Export JPEG...*

list_cellviews

```
list_cellviews
     -lib s_libName
```

Lists all cellviews in the given library.

Arguments

`-lib s_libName` Specifies the name of the library.

Important

The library must be defined in the `lib.defs` file.

Value Returned

`"s_libName/s_cellName/s_viewName"`...

Returns the lib/cell/view for all cellviews in the given library.

Example

The following example shows transcripted output for this command.

```
list_cellviews -lib mylib
"mylib/ck2/abstract" "mylib/ssad2d2/abstract" "mylib/ssad4/abstract” …
```

new_cellview

```
new_cellview
     -lib s_libName
     -cell s_cellName
     -view s_viewName
     [ -type {maskLayout | schematic | schematicSymbol | netlist} ]
```

Creates a new and empty OpenAccess cellview in the given library.

Important

> The given library must exist and be defined in the `lib.defs` file.

Arguments

`-cell` *`s_cellName`*	Specifies the cell name.
`-lib` *`s_libName`*	Specifies the library name.
`-type [ maskLayout | schematic | schematicSymbol | netlist ]`	Specifies the cellview type. If not specified, the cellview type defaults to `maskLayout`.
`-view` *`s_viewName`*	Specifies the view name.

Example

The following command creates a new cellview in the `mylib` library. By default, the new cellview type is `maskLayout`.

```
new_cellview -lib mylib -cell mycell -view layout
```

Related Information

Tcl Commands	read_db

print_window

```
print_window
     [ -printer s_name ]
```

Outputs the current window's view to a printer.

Note: Alpha blending is not supported on print devices. Space-based Router and Chip Optimizer simulates opacity by using hatch patterns. If you do not want any fill, lower your opacity (set_layer_opacity on page 255) towards zero before printing.

Important

This command cannot be used in batch/non-graphics mode.

Arguments

`-printer` *`s_name`*	Specifies the output printer. If you do not specify this argument, the Print dialog appears and lets you specify the printer.

Example

The following example outputs the current artwork window to `myBWprinter`.

```
print_window -printer myBWprinter
```

Related Information

Menu Command	*File—Print*

quit

```
quit
     [ -no_prompt [ true | false ] ]
     [ -no_save [ true | false ] ]
     [ -retcode i_value ]
```

Closes the open cellviews and shuts down the application. By default, a dialog box opens to confirm the exit and choose files to save.

Arguments

`-no_prompt [ true | false ]`

When set to `true`, exits without displaying the dialog box. When set to `false`, shows the dialog box, except when `-no_save` is `true`.

Default: `false`

`-no_save [ true | false ]`

When set to `true`, no changes are saved and the dialog box does not appear, even with `-no_prompt false`. When set to `false` and `-no_prompt` is `true`, changes are saved automatically before exiting. Otherwise, when set to `false` the dialog box appears to specify the files to save before exiting.

Default: `false`

`-retcode` *`i_value`*

Returns the given integer value. Use this when you run Space-based Router and Chip Optimizer from a script, to indicate what should be performed next.

Example

The following command is the equivalent to the *File—Exit* menu command. A dialog box appears, requesting confirmation to exit or files to save.

```
quit
```

The following command causes Space-based Router and Chip Optimizer to exit immediately, without saving changes.

```
quit -no_prompt
```

The following command causes Space-based Router and Chip Optimizer to save changes, then exit immediately.

```
quit -no_prompt -no_save false
```

When Space-based Router and Chip Optimizer is run from a script, the following commands cause Space-based Router and Chip Optimizer to save changes, then exit immediately, returning the value from `verify_connectivity`.

```
set retcode [verify_connectivity -all]
quit -no_prompt -no_save false -retcode $retcode
```

Related Information

Tcl Commands	exit_no_save
Menu Commands	*File—Exit*

read_annotations

```
read_annotations
     [ -file s_fileName ]
     [ -force ]
```

Reads annotations from the specified file.

Arguments

`-file` *`s_fileName`*	Specifies the file from which to read the annotations. If you do not specify this argument, the Read Annotations form appears, where you can choose the filename.
`-force`	Forces the annotations in the file to be added to the design in the active window regardless of whether they are associated with the design. If this argument is not specified and the active design is different from the annotations' design, the Lib/cell/view mismatch form is displayed. This form indicates which design the annotations are associated with, and allows you to import the annotations to the current design, or without associating with the active design. Unassociated annotations are listed in the Annotation Browser User page for the *Unassociated* design.

Example

The following command loads the `my_annotations.xml` file containing the annotations.

```
read_annotations -file my_annotations.xml
```

Related Information

Tcl Commands	write_annotations
Menu Commands	*File—Read Application Files—Annotations*

read_bindkeys

```
read_bindkeys
     [ -file s_fileName ]
```

Reads bindkeys and their definitions from the specified file, giving you one-key access to the Space-based Router and Chip Optimizer functions that you frequently use. If a bindkey is already defined, the previous definition is overwritten.

Arguments

`-file` *`s_fileName`*	Specifies the file from which to read the bindkeys. If you do not specify this argument, the Read Bindkeys form appears, letting you browse for the file.

Example

This command loads the `my_bindkeys.xml` file containing bindkey definitions.

```
read_bindkeys -file my_bindkeys.xml
```

Related Information

Tcl Commands	write_bindkeys
Menu Commands	*File—Read Application Files—Bind Keys...*

read_colors

```
read_colors
     [ -file s_fileName ]
     [ -window_id i_windowID ]
```

Reads the layer order, and the colors and opacities of layers and objects from the specified file, and applies them to the active cellview or cellview in the given window.

Arguments

`-file` *`s_fileName`*

Specifies the file from which to read the settings. If you do not specify this argument, the Read Colors form appears, where you can choose the filename.

`-window_id` *`i_windowID`*

Loads the settings from the file to the cellview in the given window. If this argument is not given, the active cellview is used.

Example

This command loads the `my_colors.txt` file containing the color definitions into the active window.

```
read_colors -file my_colors.txt
```

Related Information

Tcl Commands	write_colors
Menu Commands	*File—Read Application Files—Colors...*

read_db

```
read_db
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -mode { read | write | append} ]
```

Reads an existing cellview into memory.

Note: The cellview library must be specified in the `lib.defs` file.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the cellview to open. If these arguments are not specified, the Open form appears, allowing you to choose the cellview interactively.

`-mode {read | write | append}`

Specifies the read mode.

Valid values: `read` (read-only), `write`, `append`

Value Returned

`d_oaObj`	Specifies the OpenAccess identifier for the cellview.

Example

This example opens the `top_chip/layout` cellview from the `chiplib` library in read-only mode.

```
read_db -lib chiplib -cell top_chip -view layout -mode read
```

Related Information

Tcl Commands	close_db new_window

Menu Commands	*File—Open* (equivalent) *Window—New Window*

read_nets

```
read_nets
     -file s_fileName
     [ -force_load [ true | false ] ]
```

Returns a set that contains the nets named in the file. If a net named in the file cannot be found in the active design, a warning message is output.

```
Entry 'name' on line x was skipped. Failed to find net in design.
```

The command will automatically stop after 100 warnings are issued, unless you specify `-force_load`.

Arguments

`-file s_fileName` Specifies the file from which to read the net names. If a net name is not found in the active design, a warning is issued and the name is skipped.

`-force_load [ true | false ]`

When `true`, all net names in the file will be processed.

Default: (`false`) After 100 nets cannot be found, the command stops processing names.

Value Returned

d_setObj Is the set containing the nets in the design that matched net names in the file.

`-1` The command failed due to syntax errors.

Example

These commands create a variable named `mySet` that contains the nets listed in `fileA`, then highlights those nets in the artwork.

```
set mySet [ read_nets -file fileA ]
add_highlight -name myNets -set $mySet -color cyan
```

Related Information

Tcl Commands	find_net

read_view_contexts

```
read_view_contexts
     [ -file s_fileName ]
```

Reads view contexts from the specified file.

Arguments

`-file` *`s_fileName`*	Specifies the file to read view contexts from. If you do not specify a filename, the Read View Contexts form appears, where you can choose the filename interactively.

Example

The following example reads view contexts from the file `myViewContexts.xml`.

```
read_view_contexts -file myViewContexts.xml
```

Related Information

Tcl Commands	write_view_contexts
Menu Commands	*File—Read Application Files—View Contexts*

replay

```
replay
     -file s_fileName
     [ -debug_log ]
     [ -echo [ true |false ] ]
     [ -delay i_millisec ]
     [ -interactive ]
```

Runs a script of Space-based Router and Chip Optimizer Tcl commands. You can create your own text file of commands or use a Space-based Router and Chip Optimizer log file from a previous session.

Arguments

`-debug_log`

Specifies whether commands preceded by the string `#!r` should be invoked in the replay file. Lines that begin with `#!r` represent commands that were executed from a replay file. If this option is not specified, only uncommented commands are invoked. If this option is specified, file command lines beginning with `#!r` are also invoked.

`-delay` *`i_millisec`*

Specifies the length of time, in milliseconds, to pause between commands. The default is no delay.

`-echo [ true | false ]`

Specifies whether the replayed commands should be shown in the Transcript area. If the argument is given with no setting, commands from the replay file are echoed in the Transcript area. If you specify `false`, commands are not echoed when replayed until you explicitly enable them in a subsequent `replay` command.

Default: Commands are echoed in the Transcript area.

`-file` *`s_fileName`*

Specifies the name of the file to read.

`-interactive`

Specifies whether the system must pause after each command is executed. If this option is specified, the system will pause after executing each command in the replay file. When paused, the Status Indicator will display `Continue` in yellow. Click the `Continue` button to execute the next command in the replay file.

Example

The following command replays the commands in the file `mylog.txt` and echoes the commands in the Transcript area.

```
replay -file mylog.txt
```

stream_in

```
stream_in
     -gds s_fileName
     -lib s_libName
     [ -cell_map s_fileName ]
     [ -font_map s_fileName ]
     [ -hier_depth i_level ]
     [ -ignore_boxes ]
     [ -layer_map s_fileName ]
     [ -no_overwrite ]
     [ -prop_map s_fileName ]
     [ -prop_separator s_char ]
     [ -ref_lib_list s_fileName ]
     [ -tech_lib s_techName ]
     [ -tech_refs {s_techName…} ]
     [ -to_upper | -to_lower ]
     [ -top_cell s_cellName ]
     [ -view s_viewName ]
     [ -log s_fileName ]
```

(Space-based Router and Chip Optimizer only) Translates a Stream (GDSII) file to an OpenAccess library.

Following this command, you can issue the recognize_vias command to automatically recognize and mark vias.

Arguments

`-cell_map` *`s_fileName`*
Specifies the cell mapping file to use. You must specify the full path to the file.

`-font_map` *`s_fileName`*
Specifies the font mapping file to use. You must specify the full path to the file.

`-gds` *`s_fileName`*
Specifies the Stream file to translate. You must specify the complete path to the Stream file.

`-hier_depth` *`i_level`*
Specifies the hierarchical depth to translate to.

Default: 20

`-ignore_boxes`
Specifies that Stream Box records be ignored. By default, Box records are translated as rectangles.

`-layer_map` *`s_fileName`*

Specifies the layer mapping file to use. You must specify the full path to the file.

`-lib` *`s_libName`*
Specifies the output OpenAccess library to create or append to.

`-log` *`s_fileName`*
Specifies the log file to receive all output Space-based Router and Chip Optimizer messages from this command. You must specify the complete path to the log file.

`-no_overwrite`
Specifies that the translator must not overwrite existing cells. By default, existing cells are overwritten.

`-prop_map` *`s_fileName`*
Specifies the input property mapping file to use. You must specify the complete path to the file.

`-prop_separator` *`s_char`*

Specifies the property separator character used to interpret property records.

`-ref_lib_list` *`s_fileName`*

Specifies a file containing the list of libraries with cells referenced in the design. A library name in the file must match the library name in the `lib.defs` file. If this argument is not included, cells that are referenced in the Stream file but not defined, are instantiated by name and remain unbound.

`-tech_lib` *`s_techName`*
Specifies the technology library to use during the translation. If the library does not exist, it is created.

`-tech_refs` **{***`s_techName…`***}**

Creates an incremental technology database that derives from one or more of the specified parent technology databases (`parentTechs`), given by the argument value list.

Notes:

- If the library being translated does not have a technology database, this option creates an incremental technology database that has references to the specified `parentTechs`.
- If the library being translated already has a standalone technology database, this option converts that technology database to an incremental type that has references to the specified `parentTechs`.
- If the library being translated has an attached techology database, that attached technology database must be included in the list of `parentTechs`, or the translator issues an error.
- If the library being translated already uses an incremental technology database, the current references from the incremental technology database must be included in the specified `parentTechs`, or the translator issues an error. Additional reference can be provided.

`-to_lower`

Specifies that all cell and instance names be converted to lower case. This argument does not affect stream structures found in the cell mapping file, if you specify one.

`-to_upper`

Specifies that all cell and instance names be converted to upper case. This argument does not affect stream structures found in the cell mapping file, if you specify one.

`-top_cell` *`s_cellName`*

Specifies the top-level cell to translate. Using this argument is an effective way to stream in part of a design. By default, all cells are in the Stream file are translated.

`-view` *`s_viewName`*

Specifies the destination view name for the translation.

Default: `layout`

Example

The following example imports the Stream file, `superchip.gds`, and creates an OpenAccess library named `superchipoa` using the layer map file, `all_layers`.

```
stream_in -gds superchip.gds -layer_map all_layers -lib superchipoa
```

Related Information

Menu Command	*File—Import—Stream*
Tcl Command	recognize_vias *read_stream*

view_file

```
view_file
     [ -file s_fileName ]
```

Opens and displays an ASCII text file in a separate window.

(Space-based Router and Chip Optimizer only) You can use this command to view log files that you create using the `stream_in` command.

Arguments

`-file` *`s_fileName`*	Specifies the file to open and view in a separate window. If you do not specify this argument, a View File dialog appears, and lets you choose the file from your directory.

Example

The following example opens a text file, `mylog.txt`, in a separate window.

```
view_file -file mylog.txt
```

write_annotations

```
write_annotations
     [ -file s_fileName ]
     [ -type { user | violation | via | all } ]
```

Writes annotations to the specified file.

Arguments

`-file s_fileName`	Specifies the name of the file to write annotations to. If you do not specify this argument, the Write Annotations dialog appears, allowing you to interactively specify the annotations file.	
`-type s_type`	Specifies the type of annotations to include from the following:	
	`all`	Includes user, via, and violation annotations.
	`user`	Writes user annotations. This is the default.
	`via`	Writes annotations created by the `remaster_via` command.
	`violation`	Writes violation annotations.

Example

This command writes user annotations to the `my_annotations.xml` file.

```
write_annotations -file my_annotations.xml
```

Related Information

Tcl Commands	read_annotations
Menu Commands	*File—Write Application Files—Annotations*

write_bindkeys

```
write_bindkeys
     [ -file s_fileName ]
```

Writes the current bindkey definitions to the given file.

Arguments

`-file` *`s_fileName`*	Specifies the name of the file to write the bindkey definitions to. If you do not specify this argument, the Write Bindkeys dialog appears, allowing you to interactively specify the bindkeys file.

Example

This command writes the bindkey definitions to `my_bindkeys.xml`.

```
write_bindkeys -file my_bindkeys.xml
```

Related Information

Tcl Commands	read_bindkeys
Menu Commands	*File—Write Application Files—Bind Keys...*

write_colors

```
write_colors
     [ -file s_fileName ]
     [ -window_id i_windowID ]
```

Saves the current layer order, and the colors and opacities of layers and objects for the active cellview or for the cellview in the given window to the named file.

Arguments

`-file` *`s_fileName`*	Saves the settings to the given file. If you do not specify this argument, the Write Colors File dialog appears, where you can specify the colors file.
`-window_id` *`i_windowID`*	Saves the settings for the cellview in the given window. If this argument is not given, the active cellview is used.

Example

This command writes the colors definitions from the active window to the `my_colors.txt` file.

```
write_colors -file my_colors.txt
```

Related Information

Tcl Commands	read_colors
Menu Commands	*File—Write Application Files—Colors...*

write_db

```
write_db
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -hierarchical [ true | false ] ]
     [ -immediate [ true | false ] ]
     [ -overwrite_existing [ true | false ] ]
     [ -vmonly [ true | false ] ]
```

Saves the active cellview database to a file or only to memory. If no arguments are given, the Save As form appears that lets you specify the parameters.

Arguments

`-hierarchical [ true | false ]`

When `true` and saving to an existing file, all modified ctuOccurrences are written to OpenAccess. When `false` (default) and saving to an existing file, only the top-most design is saved.

Important

You cannot save hierarchically to a different cellview.

`-immediate [ true | false ]`

When set to `true` and the file already exists, the file will automatically be overwritten. If the file exists and this argument is not included or is set to `false`, a dialog box will appear to notify you that the file exists and that nothing is saved.

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the file hierarchy to save the active cellview to. If no options are given, a Save As dialog appears, letting you specify the library name, cell name, and view for the file, as well as other arguments.

`-overwrite_existing [ true | false ]`

When set to `true` and the file already exists, the file will automatically be overwritten. If the file exists and this argument is not included or is set to `false`, a dialog box will appear to notify you that the file exists and that nothing is saved.

`-vmonly [ true | false ]`

When `true`, the OpenAccess database is written to memory, but not to a disk file. When `false` (default), the memory image is updated and then the database is written to the disk file.

Example

This example saves the database for the active cellview to the *`chiplib/top_chip/layout`*.

```
write_db -lib chiplib -cell top_chip -view layout
```

Related Information

Tcl Commands	close_db new_window read_db
Menu Commands	*File—Save As* (equivalent) *File—Close* *File—Open* *Window—New Window*

write_log

```
write_log
     -file s_fileName
```

Writes the contents of the Output Transcript area, including this command, to the given file.

(Virtuoso Routing IDE only) If the MAKE_RDE_LOG Unix variable is set, log files (`virtuosoRDE`*`yyyymmdd_hhmmss`*`.log`) for Space-based Router and Chip Optimizer are automatically created in the directory where you start the software.

(Space-based Router and Chip Optimizer only) Log files for Space-based Router and Chip Optimizer are automatically created in the directory where you start the software.

Arguments

`-file` *`s_fileName`*	Specifies the name of the file to write the log to.

Example

The following example writes the contents of the log to the file `mylog.xml`.

```
write_log -file mylog.xml
```

write_view_contexts

```
write_view_contexts
     [ -file s_fileName ]
```

Writes view contexts to the specified file.

Arguments

`-file` *`s_fileName`*	Specifies the name of the file to write view contexts to. If you do not specify this argument, a Write View Contexts dialog appears, letting you specify the filename.

Example

The following example writes view contexts to the file `latestViews.xml`.

```
write_view_contexts -file latestViews.xml
```

Related Information

Tcl Commands	read_view_contexts
Menu Commands	*File—Write Application Files—View Contexts*

Stream/OASIS-to-OpenAccess Layer Map File

Note: Tcl commands in this section are not supported by Virtuoso® Routing IDE.

The layer map file maps stream (GDSII or OASIS) layer and data type numbers to OpenAccess layer and purpose names. The file format is the same as the layer map file format used with the OpenAccess stream translators (`strm2oa` and `oa2strm`), with some added optional arguments:

- *`oaMaterial`* specifies the material type as `metal`, `cut` or `other`.
- *`oaMaskNumber`* specifies the fabrication order.
- *`direction`* specifies the preferred direction as `horizontal` or `vertical`.

```
oaLayer oaPurpose strmLayer strmDataType [oaMaterial] [oaMaskNumber]
    [direction]
```

The Stream/OASIS-to-OpenAccess layer map file is required to run read_stream and write_stream and is specified using the `-layer_map` argument.

> **Note:** Each complete line in the layer map file must be unique. For `read_stream`, you can specify multiple stream (*strmLayer*/*strmDataType*) entries for the same LPP (*oaLayer*/*oaPurpose*) entry. For `write_stream`, you can specify multiple LPP (*oaLayer*/*oaPurpose*) entries for the same stream (*strmLayer*/*strmDataType*) entry.

If you use `read_stream` to detect vias (`-detect_vias)`, for optimal performance you should do the following:

- Specify the appropriate material types, `metal` and `cut`, for the via metal and cut layers in the *`oaMaterial`* column. Other layers can be type `other`.
- Specify the fabrication order for the layers using ascending mask numbers in the *`oaMaskNumber`* column. For example, metal1 would be 1, via1 would be 2, metal2 would be 3, and so on.

Figure 1-1 Example of a Stream/OASIS-to-OpenAccess Layer Map File

```
#oaLayerName  oaPurpose  strmLayer strmDataType oaMaterial oaMaskNumber direction
PImp          drawing         13        13         PImplant   3
Metal1        drawing1        1         1          metal      6        horizontal
Via1          drawing2        2         2          cut        7
Metal2        drawing3        3         3          metal      8        vertical
Via2          drawing4        4         4          cut        9
Metal3        drawing5        5         5          metal      10
Via3          drawing6        6         6          cut        11
Metal4        drawing7        7         7          metal      12
Via4          drawing8        8         8          other      13
Metal5        drawing9        9         9          other      14
```

For this example, vias on `Via1`, `Via2`, and `Via3` could be automatically detected by read_stream but vias on `Via4` could not be detected because the *`oaMaterial`* values are not set properly for `Via4` and `Metal5`.

read_stream

```
read_stream
     -lib s_libName
     -cell s_topCellName
     [ -view s_viewName ]
     -layer_map s_fileName
     { -gds s_fileName | -oasis s_fileName }
     [ -refgds {s_fileName…} | -refoasis {s_fileName…} ]
     [ -area {f_xlo f_ylo f_xhi f_yhi} ]
     [ -calibre_run_dir s_dir ]
     [ -calibre_run_prefix s_prefix ]
     [ -detect_vias [ true | false ] ]
     [ -device_prop_number s_annotated_stream_device_prop_number ]
     [ -instance_prop_number s_annotated_stream_instanc_prop_number ]
     [ -load_layers {{i_layerNumber:i_dataTypeNumber}…}
     | -ignore_layers {{i_layerNumber:i_dataTypeNumber}…}
     | -load_layers_file s_fileName
     | -ignore_layers_file s_fileName ]
     [ -net_prop_number s_annotated_stream_net_prop_number ]
     [ -threads i_count]
     [ -gds2 s_fileName | -oasis2 s_fileName ]
     [ -cell2 s_topCellName ]
     [ -refgds2 {s_fileName…} | -refoasis2 {s_fileName…} ]
     [ -bump i_layerBump ]
```

(Space-based Router and Chip Optimizer only) Opens a GDSII or OASIS stream file and creates an OpenAccess technology library for the data.

Required Arguments

`-cell` *`s_topCellName`* Specifies the top-level cell to load. If you want to stream in only part of a design, specifying a top cell is one way to accomplish this.

`-gds` *`s_fileName`* Specifies the name of the GDSII stream file to open. Cannot be used with the `-oasis` argument.

`-layer_map` *`s_fileName`* Specifies the Stream/OASIS-to-OpenAccess Layer Map File to use to translate the stream layers and data types in the stream file to their respective OpenAccess layer names and purposes.

`-lib` *`s_libName`*

Specifies the name of the OpenAccess technology library to create when processing the stream file.

Important

> If the library already exists, the technology information in the library will be overwritten by this command.

`-oasis` *`s_fileName`*

Specifies the name of the OASIS stream file to open. Cannot be used with the `-gds` argument.

Optional Arguments

`-area` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Loads only data in the rectangular area given by the lower left and upper right x and y coordinates.

Default: Loads entire area

`-bump` *`i_layerBump`*

Specifies the number by which the layer numbers in the secondary stream input files are incremented to find the corresponding layer purpose pair for Space-based Router and Chip Optimizer.

`-calibre_run_dir` *`s_dir`*

If either `-calibre_run_dir` or `-calibre_run_prefix` is given, annotation files generated by Mentor Graphics® Calibre® LVS will be loaded, if they exist, including:

- Calibre Layout Netlist Names (LNN) files that are used to change the names of nets from their stream property number annotation to their net list names.
- Port cell files and Spice network files that are used to create instance terminals and device names for instances.

This argument specifies the directory to search for these files.

Default: current working directory

`-calibre_run_prefix` *`s_prefix`*

If either `-calibre_run_dir` or `-calibre_run_prefix` is given, annotation files generated by Calibre LVS will be loaded, if they exist. This argument limits the files to those that include the given prefix in their name (.*`s_prefix`*`.lnn`, *`s_prefix`*`.devmap`, *`s_prefix`*`.port_cells`, and *`s_prefix`*`_pin-xy.spi`).

Default: *`s_prefix`* is the top cell name given by `the -cell` argument.

`-cell2` *`s_topCellName`* Specifies the top-level cell to load for the secondary design given by `-gds2` or -`oasis2`.

`-detect_vias [ true | false ]`

When `true`, enables the automatic detection of vias.

Important

> The layer map file, given by the `-layer_map` argument, must specify the appropriate material types, `metal` and `cut`, for the via metal and cut layers. Other layers can be type `other`.
>
> The layer map file must also specify the fabrication order for the layers using ascending mask numbers. For example, metal1 would be 1, via1 would be 2, metal2 would be 3, and so on.

Default: Via detection is enabled.

`-device_prop_number` *`i_annotated_stream_device_prop_number`*

Specifies stream property number that is used to specify devices.

Default: 7

`-gds2` *`s_fileName`* Specifies the name of the GDSII stream file to open for a second design that will be merged with the primary design given by `-gds` or `-oasis`. Cannot be used with the `-oasis2` argument.

`-ignore_layers` **{**`{`*`i_layerNumber`*`:`*`i_dataTypeNumber`*`}`**…}**

Excludes from loading the specified subset of layers, given by *`i_layerNumber`*:*`i_dataTypeNumber`* pairs.

For example,

```
{2:0 4:0 5:0}
```

will exclude layer and data types for

- Layer 2, data type 0
- Layer 4, data type 0
- Layer 5, data type 0

This argument cannot be used with `-load_layers`, -`ignore_layers_file`, or -`load_layers_file`.

Default: No layers are ignored.

`-ignore_layers_file` *`s_fileName`*

Specifies a file that contains the list of layers to be excluded, each represented by a pair of numbers, separated by a space, one per line.

```
i_layerNumber i_dataTypeNumber
```

This argument cannot be used with `-ignore_layers`, -`load_layers_file`, nor -`load_layers`.

`-instance_prop_number` *`i_annotated_stream_instance_prop_number`*

Specifies stream property number that is used to specify instances.

Default: 6

`-load_layers` **{**`{`*`i_layerNumber`*`:`*`i_dataTypeNumber`*`}`**…}**

Loads only the specified subset of layers, given by *`i_layerNumber`*:*`i_dataTypeNumber`* pairs. The listed pairs must be specified in the layer map file.

For example,

```
{0:0 1:0 2:0}
```

will load only layer and data types for

- Layer 0, data type 0
- Layer 1, data type 0
- Layer 2, data type 0

This argument cannot be used with `-ignore_layers`, -`ignore_layers_file`, nor -`load_layers_file`.

Default: All layers are loaded.

`-load_layers_file` *`s_fileName`*

Specifies a file that contains the list of layers to be loaded, each represented by a pair of numbers, separated by a space, one per line.

```
i_layerNumber i_dataTypeNumber
```

This argument cannot be used with `-ignore_layers`, -`ignore_layers_file`, nor -`load_layers`.

`-net_prop_number` *`i_annotated_stream_net_prop_number`*

Specifies stream property number that is used to specify nets.

Default: 5

`-oasis` *`s_fileName`*

Specifies the name of the OASIS stream file to open. Cannot be used with the `-gds` argument.

`-oasis2` *`s_fileName`*

Specifies the name of the OASIS stream file to open for a second design that will be merged with the primary design given by `-gds` or `-oasis`. Cannot be used with the `-gds2` argument.

`-refgds` **{***`s_libName`…***}**

Specifies one or more GDSII libraries in which to search for master cells in the design given by `-gds`.

`-refgds2` **{***`s_libName`…***}**

Specifies one or more GDSII libraries in which to search for master cells in the design given by `-gds2`.

`-refoasis` **{***`s_libName`***…}** Specifies one or more OASIS libraries in which to search for master cells in the design given by `-oasis`.

`-refoasis2` **{***`s_libName`***…}**

Specifies one or more OASIS libraries in which to search for master cells in the design given by `-oasis2`.

`-threads` *`i_count`* Specifies the maximum number of threads, or processors, to run in parallel for this command. Using multiple processors, if available, can decrease processing time.

Default: 1 (single processor)

`-view` *`s_viewName`* Specifies the input view name.

Default: `layout`

Related Information

Tcl Commands write_stream
write_stream_by_net

write_stream

```
write_stream
     {-gds s_fileName | -oasis s_fileName}
     -lib s_libName
     -cell s_cellName
     -view s_viewName
     -layer_map s_fileName
     [ -area {f_xlo f_ylo f_xhi f_yhi} ]
     [ -clip_shapes [ true | false ] ]
     [ -keep_empty [ true | false ] ]
     [ -output_layers {s_lpp…} | -ignore_layers {s_lpp…} ]
     [ -purp_in_oasis_lmap [ true | false ] ]
     [ -rect_as_poly [ true | false ] ]
     [ -v i_level ]
```

(Space-based Router and Chip Optimizer only) Outputs the given cellview `(-lib -cell -view)` to an OASIS or GDSII stream file, using the layer map file to map the OpenAccess layers and purposes to stream layers and data types.

Required Arguments

`-cell` *`s_topCellName`*	Specifies the name of the top cell to output.
`-gds` *`s_fileName`*	Specifies the name of the file to which the GDSII stream data will be written.
`-ignore_layers` **{***`s_lpp…`***}**	Excludes the given layer purpose pairs from processing. By default, data on all layer purpose pairs in the layer map file are written.
`-layer_map` *`s_fileName`*	Specifies the Stream/OASIS-to-OpenAccess Layer Map File to use to translate the stream layers and data types in the stream file to their respective OpenAccess layer names and purposes.
`-lib` *`s_libName`*	Specifies the name of the cellview library.
`-oasis` *`s_fileName`*	Specifies the name of the file to which the OASIS stream data will be written.
`-output_layers` **{***`s_lpp…`***}**	Limits processing to the specified layer purpose pairs. By default, data on all layer purpose pairs in the layer map file are written.
`-view` *`s_viewName`*	Specifies the name of the view to output.

Optional Arguments

`-area {f_xlo f_ylo f_xhi f_yhi}`

Specifies the rectangular area of the top cell to be written out, given by the lower left and upper right x and y coordinates for the area.

Default: Outputs the entire area

`-keep_empty [ true | false ]`

When set to `true`, empty cells and their instances are included.

Default: `true`

`-purp_in_oasis_lmap [ true | false ]`

By default and when set to `true`, layer and purpose names are output to the OASIS file layer map section. When set to `false`, the purpose names are excluded. This is useful if your OASIS file reader cannot process purpose names.

`-rect_as_poly [ true | false ]`

When set to `true`, rectangles are output as polygons.

Default: `true`

`-v i_level`

Specifies the level of verbosity for output messages.

Default: 0

`-voltage_datatype {{f_voltage i_dataType}…}`

Specifies voltage-datatype pairs. When shapes are output, if a shape belongs to a net with a voltage, the datatype for the shape is determined by comparing the net's maximum voltage swing (Vmax-Vmin) with the table voltages, which must be given in increasing order. For example,

```
-voltage_datatype { 0.2 2  0.4 4  0.6 6 }
```

is interpreted as:

- For 0 < voltage swing <= 0.2, use datatype 2.
- For 0.2 < voltage swing <= 0.4, use datatype 4.
- For 0.4 < voltage swing <= 0.6, use datatype 6.

Related Information

Tcl Commands	read_stream write_stream_by_net

write_stream_by_net

```
write_stream_by_net
     {-gds s_fileName | -oasis s_fileName}
     -nets {s_netName…}
     -layer_map s_fileName
     [ -layers {s_layerName…} ]
     [ -purp_in_oasis_lmap [ true | false ] ]
     [ -starting_datatype i_dataType [ -net_datatype_map s_fileName ] ]
```

(Space-based Router and Chip Optimizer only) Creates either a GDSII or OASIS file that contains the shapes that are attached to the given nets. You can optionally limit the output to shapes on specific layers. By default, for each net in the list (-`nets`), shapes in the net are assigned to the datatype corresponding to the `drawing` purpose for the shape's layer, according to the layer map. If `-starting_datatype` is given, the first net will use this number and the datatype number will be incremented by one for each subsequent net.

Required Arguments

`-gds` *`s_fileName`*	Specifies the name of the file to which the GDSII stream data will be written.
`-layer_map` *`s_fileName`*	Specifies the Stream/OASIS-to-OpenAccess Layer Map File to use for mapping OpenAccess layer names and purposes to their respective stream layers and datatypes.
`-nets` **{***`s_netName…`***}**	Limits output to the specified nets.
`-oasis` *`s_fileName`*	Specifies the name of the file to which the OASIS stream data will be written.

Optional Arguments

`-layers` **{***`s_layerName…`***}**

Limits output to shapes on the given layers.

`-net_datatype_map` *`s_fileName`*

Specifies the name of the file to which the mapping of net names and datatype is output. Must be used with `-starting_datatype`.

Default: `.netDatatypeMapFile`

`-purp_in_oasis_lmap [ true | false ]`

By default and when set to `true`, layer and purpose names are output to the OASIS file layer map section. When set to `false`, the purpose names are excluded. This is useful if your OASIS file reader cannot process purpose names.

`-starting_datatype` *`i_number`*

Specifies the starting datatype for the output stream. Shapes for the first net are assigned to this datatype, and the datatype number will be incremented by one for each subsequent net.

Related Information

Tcl Commands	read_stream write_stream_by_set

write_stream_by_set

```
write_stream_by_set
     {-gds s_fileName | -oasis s_fileName}
     -set d_setObj
     -layer_map s_fileName
     [ -layers {s_layerName…} ]
     [ -purp_in_oasis_lmap [ true | false ] ]
```

(Space-based Router and Chip Optimizer only) Creates either a GDSII or OASIS file that contains the shapes for objects in the given set.

Required Arguments

`-gds` *`s_fileName`* — Specifies the name of the file to which the GDSII stream data will be written.

`-layer_map` *`s_fileName`* — Specifies the Stream/OASIS-to-OpenAccess Layer Map File to use for mapping OpenAccess layer names and purposes to their respective stream layers and datatypes.

`-oasis` *`s_fileName`* — Specifies the name of the file to which the OASIS stream data will be written.

`-set` *`d_setObj`* — Specifies the set of objects.

Optional Arguments

`-layers` **{***`s_layerName…`***}**

Limits output to shapes on the given layers.

`-purp_in_oasis_lmap [ true | false ]`

By default and when set to `true`, layer and purpose names are output to the OASIS file layer map section. When set to `false`, the purpose names are excluded. This is useful if your OASIS file reader cannot process purpose names.

Related Information

Tcl Commands	read_stream write_stream_by_net

2

Set Commands

A set is a collection of design objects. Sets can be selected and highlighted. You can have only one selected set for a cellview but can have many highlight sets. When you interactively select objects in a cellview window, you are adding objects to the selected set. Similarly, in highlight mode, you add and remove objects from highlight sets.

Chapter 3, Edit Commands, describes commands that create sets (such as find_instance), and manage the selection set and highlight sets. When a set is created, it is uniquely identified by a *set identifier*. You can save the set identifier in a Tcl variable and use the variable as an argument for the set commands in this chapter.

This chapter describes the Set commands that are used for basic set functions. The commands are presented in alphabetical order.

and_sets

```
and_sets
     -set1 d_setObj
     -set2 d_setObj
```

Creates a new set that contains objects that are in both `set1` and `set2`. This is also known as a *set intersection.*

Arguments

`-set1` *`d_setObj`*	Specifies the first set.
`-set2` *`d_setObj`*	Specifies the second set.

Value Returned

`d_setObj`	Specifies the set identifier for the new set.
`0`	The new set is not created.

Example

```
add_highlight -set [and_sets -set1 [get_highlight -name HL1] -set2
[get_selection_set]] -color blue -name HL4
```

Creates a highlight set, `HL4`, that contains objects that are common to both the `HL1` highlight set and the selected set.

clear_set

```
clear_set
     -set d_setObj
```

Removes all objects contained by the set and redraws the current window if the set is either the selection set or one of the highlight sets.

Arguments

`-set` *`d_setObj`*	Specifies the set to remove objects from.

Value Returned

`1`	Objects are removed from the set.
`0`	No objects are removed from the set.

Example

```
clear_set -set $myset2
```

Clears all objects contained by the set specified by the `myset2` variable.

```
clear_set -set [get_selection_set]
```

Clears all objects from the selected set.

Related Information

Tcl Commands	deselect_set

copy_set

```
copy_set
     -set d_setObj
```

Creates a new set with all objects from the specified set.

Arguments

`-set` *`d_setObj`*	Specifies the set to copy.

Value Returned

`d_setObj`	Specifies the set identifier for the new set.
`0`	The copy is not created.

Example

```
set mynew_set [copy_set -set my_set]
```

Copies the set specified by the *`my_set`* variable into a new set specified by the `mynew_set` variable.

not_sets

```
not_sets
     -set1 d_setObj
     -set2 d_setObj
```

Creates a new set that contains objects in `set1` that are not in `set2`.

Arguments

`-set1` *`d_setObj`*	Specifies the first set. All objects in this set, but not in the second set, are included in the new set.
`-set2` *`d_setObj`*	Specifies the second set. All objects in this set, but not in the first set, are included in the new set.

Value Returned

`d_setObj`	Specifies the set identifier for the new set.
`0`	The new set is not created.

Example

```
select_all
set myset1 [get_selection_set]
set myset2 [find -cellview -inst I110_BLOCKA.*]
not_sets -set1 $myset1 -set2 $myset2
```

Finds the objects that are in `myset1` and not in `myset2`. In this example, the first set contains all the objects and the second set contains only instances that match the expression given as the instance name. The resulting set contains all objects except those in `myset2`.

or_sets

```
or_sets
     -set1 d_setObj
     -set2 d_setObj
```

Creates a new set that contains objects that are in either of the specified sets. This is also known as a *set union.*

Arguments

`-set1` *`d_setObj`*	Specifies the first set for the union.
`-set2` *`d_setObj`*	Specifies the second set for the union.

Value Returned

`d_setObj`	Specifies the set identifier for the new set.
`0`	The new set is not created.

Example

```
set myset1 [find -cellview -inst I90.*]
set myset2 [find -cellview -inst I110_BLOCKA.*]
or_sets -set1 $myset1 -set2 $myset2
```

Creates a new set that is the union of `myset1` and `myset2`. In this example, the first set contains all instances matching `I90.*`. This is combined with the second set that contains all instances matching `I110_BLOCKA.*`.

replace_set

```
replace_set
     -set1 d_setObj
     -set2 d_setObj
```

Replaces objects from one set with objects from another set.

Arguments

`-set1` *`d_setObj`*	Specifies the set to copy objects from.
`-set2` *`d_setObj`*	Specifies the set to replace.

Value Returned

`d_setObj`	Specifies the set identifier for `set2`.
`0`	The objects were not copied.

Example

The following example removes objects from the highlight set named `HL2` and copies objects from the highlight set named `HL1` into `HL2`.

```
replace_set -set1 [get_highlight -name HL1] -set2 [get_highlight -name HL2]
```

The following example is equivalent to choosing *Select Contents* from the highlight set pop-up menu for the highlight set named `HL1` in the Layer Object Display Panel. The command adds objects from `HL1` to the selected set.

```
replace_set -set1[or_sets -set1 [get_highlight -name HL1]
     -set2 [get_selection_set]] -set2[get_selection_set]
```

xor_sets

```
xor_sets
     -set1 d_setObj
     -set2 d_setObj
```

Creates a new set that contains objects that are in either of the specified sets, but not in both.

Arguments

`-set1` *`d_setObj`*	Specifies the first set.
`-set2` *`d_setObj`*	Specifies the second set.

Value Returned

`d_setObj`	Specifies the set identifier for the new set.
`0`	The new set is not created.

Example

```
xor_sets -set1 $myset1 -set2 $myset2
```

Creates a new set that contains objects that are in `myset1` or `myset2`, but not in both.

3

Edit Commands

This chapter describes the Edit commands that perform find, select, highlight, properties, scheme manager, and wire editor functions. The commands are presented in alphabetical order.

- find_by_area on page 123
- find_by_index on page 125
- find_composite_net on page 126
- find_group on page 129
- find_inst_term on page 132
- find_instance on page 134
- find_instance_of on page 137
- find_layer on page 140
- find_neighbor_nets on page 141
- find_net on page 143
- find_net_portion on page 146
- find_scenic_nets on page 150
- find_shape on page 153
- find_shielded_nets on page 157
- find_terminal on page 158
- find_text on page 160
- flatten on page 162
- ge_to_rde_selection on page 163
- get_cell_view_id on page 165
- get_connected_net on page 166
- get_current_highlight on page 167
- get_element on page 168
- get_highlight on page 169
- get_inst_headers on page 170
- get_midpoint on page 171
- get_num_highlights on page 172
- get_occurrence on page 173

add_highlight

```
add_highlight
     -color s_color
     -name s_name
     -set d_setObj
```

Highlights the specified objects. If no arguments are given, an empty highlight set is created. Once created, the highlight set is listed in the Objects section of the Layer Object Display Panel under highlights.

Arguments

`-color` *`s_color`*	Specifies the color to use for highlighting.
	Default: The next available color as shown in the Create Highlight Set form.
`-name` *`s_name`*	Specifies the name to assign to the highlight set.
	Default: `HL`*`x`* where *`x`* is the next available integer value.
`-set` *`d_setObj`*	Specifies a set of objects to highlight.

Value Returned

`1`	The highlight set is created and the given objects are highlighted.

Example

In the following example, the `find` command finds instances matching the given expression. The results are stored in `myset`, then are highlighted in green in the artwork window. The highlight set is given the name `I101s`.

```
set myset [find_instance -name I101*]
add_highlight -color "green" -set $myset -name I101s
```

Related Information

Menu Command	*Edit—Highlight—Create Highlight Set…*

add_object_to_net

```
add_object_to_net
     {-net_name s_netName | -determine_net}
     {-object d_ctuObj | -set d_setObj}
```

Adds objects to a net. You can specify a single object (`-object`) or a set of objects (`-set`).

Arguments

`-determine_net`	Adds each object to a net it touches.
`-net_name` *`s_netName`*	Specifies the name of the net.
`-object` *`d_ctuObj`*	Specifies the identifier for the object to add.
`-set` *`d_setObj`*	Specifies the identifier for a set of objects to add.

Example

The following example adds objects in the selected set to the `mynet` net.

```
add_object_to_net -net mynet -set [get_selection_set]
```

The following example adds each object in the selected set to a net that the object touches.

```
add_object_to_net -determine_net -set [get_selection_set]
```

Related Information

Tcl Command	remove_object_from_net

add_object_to_set

```
add_object_to_set
     -object d_ctuObj
     -set d_setObj
```

Adds an object to a set.

Arguments

`-object` *`d_ctuObj`*	Specifies the object to add.
`-set` *`d_setObj`*	Specifies a set.

Example

The following example adds routes in the selected set to the `HL1` highlight set.

```
foreach item [ml [get_selection_set]] {
  if {[string compare [ip type $item] "ctuRoute"] == 0} {
    add_object_to_set -object $item -set [get_highlight -name HL1]
  }
}
```

Related Information

Tcl Command	remove_object_from_set

check_trim

```
check_trim
     [ -lpp1 layerName ]
     [ -error_types { all | { [space] [shape] [missing] [floating] [overlapping]
     } } ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
```

Classifies the violations that arise in trim checking.

Arguments

`-lpp1` *`layerName`*

Restricts processing to the specified layer.

`-error_types { all | { [space] [shape] [missing] [floating] [overlapping] } }`

Specifies the types of violations that can arise in trim checking. It can be a single violation or combination of many violations.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the boundary of the area to process, given the lower-left and upper-right coordinates. By default, the entire design is processed.

Example

The following example checks all violations associated to M1 layer inside the given region.

```
check_trim -lpp1 M1 -error_types { all space shape missing floating overlapping }
-region {100.1 100.1 150.4 18.80}
```

copy_net_routes

```
copy_net_routes
     {-sourceNets {s_netName…} | -sourceSet d_setObj}
     {-targetNets {s_netName…} | -targetSet d_setObj}
     [ -type {exact | automatic} ]
     [ -allowViolations [ true | false ] ]
     [ -allowCrossings [ true | false ] ]
     [ -minSpacing [ true | false ] ]
```

Copies routes in the source nets to one or more target nets. By default, the copied routes will have the exact same topology as the source net routes but will have the connectivity of the target net. When an exact copy is not possible, `-type automatic` will adjust routes to complete connections.

Arguments

-allowCrossings [true | false]

(Applies only when -type automatic is specified) When true, wire crossings will be allowed when needed to complete a connection.

Default: (false) Will not cross wires during copy.

-allowViolations [true | false]

(Applies only when -type automatic is specified) When true, violations will be allowed.

Default: (false) If copied routes cause violations, opens will be left in place of those violations.

-minSpacing [true | false]

When false, it will keep the distance from the terminals.

Default: (true) Will copy wires at minSpacing distance relative to surrounding wires.

-sourceNets *s_netName* Specifies the name of the nets from which to copy the routes.

-sourceSet *d_setObj* Specifies the set of nets whose routes will be copied.

-targetNets **{***s_netName…***}**

Copies routes from the source nets to the nets in this list.

-targetSet *d_setObj* Copies routes from the source nets to the nets in the set.

-type *s_type* Chooses whether the exact topology (default) is copied or routes can be adjusted to complete the connection (automatic).

Example

The following example copies the bit0 net routes to the bit1 and bit2 nets.

```
copy_net_routes -sourceNet bit0 -targetNets {bit1 bit2}
```

Related Information

Tcl Command copy_route

Menu Command	*Edit—Copy Route*

copy_route

```
copy_route
     -set d_setObj
     -dx f_userunit
     -dy f_userunit
     [ -adjust_offset ]
     [ -numberOfCopies i_count ]
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
```

Copies routes in the given set, replicating the topology of the copied routes and assigning the correct connectivity with respect to the new locations. Multiple copies can be created by a single command.

No checking is performed by this command.

You can remove the copied routes using the split_oa_terminals command.

Arguments

`-adjust_offset`	Automatically adjusts the given offsets to place the copied routes. This permits the copied routes to be "snapped" to nearby terms. If this argument is not given and when a term cannot be found within the given offset area, the copied routes are placed where specified by `-dx` and `-dy`.
`-dx` *`f_userunit`*	Specifies the x-coordinate offset for the copied routes.
`-dy` *`f_userunit`*	Specifies the y-coordinate offset for the copied routes.
`-numberOfCopies` *`i_count`*	
	Creates multiple replicas. The first copy is offset from the original by `-dx` and `-dy`, the second copy by `-dx*2` and `-dy*2`, and so on. Default: 1
`-orient`	Specifies the orientation for the copied routes. The orientation origin is the center of the bounding box containing the set. Valid values are: `R0`, `R90`, `R180`, `R270`, `MY`, `MYR90`, `MX`, and `MXR90`. For a description of orientation values, refer to "Orientation Key" on page 208 . Default value: `R0`
`-set` *`d_setObj`*	Copies routes in the given set.

Example

The following example makes a copy of the `data0` net and places it in the design with an x-coordinate offset of 2.5 from the original net's location.

```
copy_route -set [find_net -name data0] -dx 2.5 -dy 0
```

Related Information

Tcl Command	copy_net_routes split_oa_terminals
Menu Command	*Edit—Copy Route*

create_highlight

```
create_highlight
```

Displays the Create Highlight form. Using this form, you can create new highlight sets and specify the set names and colors.

Arguments

None

Value Returned

`1`	The Create Highlight form is successfully displayed.

Related Information

Menu Commands	*Edit—Highlight—Create Highlight Set…*

create_set

```
create_set
```

Creates an empty set.

Arguments

None

Value Returned

d_setObj	Specifies the set identifier for the new empty set.
0	The empty set is not created.

Example

The following example creates a Tcl variable that contains the set identifier for an empty set.

```
set myset [create_set]
```

You can use the `$myset` variable as an argument for other commands that require a set identifier as an input.

cut

```
cut
     -region {f_xlo f_ylo f_xhi f_yhi}
     [ -set d_setObj ]
```

Breaks a wire segment into two segments.

This command is normally performed interactively in the workspace by choosing *Edit—Cut*. Segments that intersect the cut area perimeter are broken into two segments with an overlap of the minimum width. If `-set` is given, only segments in the given set are cut.

Arguments

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the lower-left (*f_xlo*, *f_ylo*) and the upper-right (*f_xhi*, *f_yhi*) bounding box coordinates for the cut area.

`-set d_setObj` Operates only on segments in the given set.

Related Information

Tcl Commands	icut move
Menu Commands	*Edit—Cut*

delete

```
delete
     [ -no_undo ]
     [ -set d_setObj ]
     [ -top_level_pin_shapes ]
     [ -unconnected_routes ]
     [ -delete_empty_pins ]
```

Deletes entire nets and routes, route segments, vias, polygons, rectangles, octagons, and instances (and their connectivity) in the selected set for the active cellview. Only objects on the top level can be deleted. Guides are created for deleted nets and routes to indicate the connectivity. Diamond-shaped vertical guides are created for deleted vias. If you do not specify the `-no_undo` argument, you can reverse the deletion using an `undo` command. You can optionally delete unconnected pins with no child shapes (`-delete_empty_pins`).

Arguments

`-delete_empty_pins`	Removes all shapes for unconnected pins (pins with no child shapes). By default, empty pins are not deleted.
`-no_undo`	Prevents you from being able to reverse this action with an `undo` command. If this option is omitted, you can restore the removed objects using the `undo` command.
`-set` *`d_setObj`*	Identifies the set to delete objects from. By default, objects in the selected set are deleted.
`-top_level_pin_shapes`	When set to `true`, selected top level pin shapes will be deleted. By default (`false`), top level pin shapes are not included.
`-unconnected_routes`	Removes entire routes in the selected set without creating guides for unconnected routes. By default, guides are created for all deleted routes.

Example

The following example deletes top-level objects from the `HL1` highlight set.

```
delete -set [get_highlight -name HL1]
```

Related Information

Tcl Commands	split_oa_terminals

Menu Commands *Edit—Delete*

delete_cellview

```
delete_cellview
     -lib s_libName
     -cell s_cellName
     -view s_viewName
     [ -delete_non_empty ]
```

Deletes cellviews.

Arguments

`-cell s_cellName`	Specifies the name of the cell for the cellview.
`-delete_non_empty`	If you did not specify a view, this argument causes cells to be removed even when they are not empty, otherwise the cell is deleted only when it has no views.
`-lib s_libName`	Specifies the name of the library.for the cellview.
`-view s_viewName`	Specifies the view to delete.

Example

The following command deletes the `layout` view in `mylib` for the `TEST` cell.

```
delete_cellview -lib mylib -cell TEST -view layout
```

delete_fill

```
delete_fill
     -type { all | { [floating] [connected] [notch] } }
     { -layer { all | {s_layerName …} }
     | [ -isolate_nets {s_netName …} {-isolate_dist f_userunit
       [ -isolate_dist_above f_userunit ] [ -isolate_dist_below
     f_userunit ] } ] }
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -connected_nets {s_netName…} ]
     [ -violators [ true | false ] ]
     [ -use_check_annotations [ true | false ] ]
```

Deletes fill shapes on specific layers and, optionally, around specific nets. Optionally deletes fill shapes of all types or of specific types.

Arguments

`-connected_nets` **{***`s_netName…`***}**

(Applies only when `-type connected` or `-type all` is given) Specifies the nets to remove shapes from on the `fill` purpose. If this argument is not given and `type` is `all` or `connected`, then all shapes on the `fill` purpose that are connected to power or ground nets are removed. The vias connecting the fill shapes to the nets are also removed.

`-isolate_dist` *`f_userunit`*

Specifies the distance to clear around each net. Must be used with the `-isolate_nets` argument.

`-isolate_dist_above` *`f_userunit`*

Specifies the distance to clear above each net. Must be used with the `-isolate_nets` and `-isolate_dist` arguments.

`-isolate_dist_below` *`f_userunit`*

Specifies the distance to clear below each net. Must be used with the `-isolate_nets` and `-isolate_dist` arguments.

`-isolate_nets` **{***`s_netName …`***}**

Specifies the nets to process. Fill shapes will be removed around the given nets on the shape layer, and the layers above and below, unless further restricted by the `-layer` argument. The distance cleared around each net is given by the `-isolate_dist` argument.

`-layer {all |` **{***`s_layerName …`***}**`}`

Restricts processing to the specified layers. By default and when `all` is specified, all routing layers are processed.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary of the area to process, given the lower left and upper right coordinates.

`-type {all |` **{** `[floating] [connected] [notch]` **}** `}`

Specifies the types of fill shapes to delete.

`all`	Removes all shapes on the `fill` and `gapFill` purpose. For connected fill, the connecting vias are also removed.
`connected`	Removes shapes on the `fill` purpose and the vias connecting the fill shapes to the nets given by the `connected_nets` argument. If the `connected_nets` argument does not exist, all shapes on the `fill` purpose that are connected to power or ground nets are removed and their connecting vias are also removed.
`floating`	Removes all unconnected shapes on the `fill` purpose.
`notch`	Removes all shapes on the `gapFill` purpose.

`-use_check_annotations [ true | false ]`

Used with `-violators` to use existing violation markers when determining what fill shapes are in violation.

`-violators [ true | false ]`

Removes any fill shapes that are in violation with any other shapes, limited by the scope given by arguments such as `-layer`, `-type`, and `-region`. Must also specify `-use_check_annotations` to use this option.

Example

The following example removes all fill shapes within a distance of 1.0 user units to the `A_32_INST/i_918` net.

```
delete_fill -isolate_nets A_32_INST/i_918 -isolate_dist 1.0 -type all
```

The following example removes all shapes on the `fill` purpose of the design that are connected to a power or ground net.

```
delete_fill -type connected -layer all
```

Related Information

Tcl Commands	create_fill create_net_fill

delete_floating_trims

```
delete_floating_trims
     [ -layer { all | {s_layerName …} } ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
```

Deletes trims that are not abutted to segments or vias.

Arguments

`-layer { all | {s_layerName …}}`

Restricts processing to the specified layers. By default, and when `all` is specified, all routing layers are processed.

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the boundary of the area to process, given the lower-left and upper-right coordinates. By default, the entire design is processed.

Examples

The following example removes all the floating trims in the design.

```
delete_floating_trims -layer all
```

The following example removes the floating trims associated with the `M1` and `M2` layers in the design.

```
delete_floating_trims -layer {M1 M2}
```

delete_pin_halo_shapes

```
delete_pin_halo_shapes
     { -all | -inst_set d_setObj | -set d_setObj }
```

Removes pin halo shapes from the terminals for all the nets in the design, selected instances, or the selected nets.

Arguments

`-all`	Removes the pin halo shapes on all terminals for all nets in the design.
`-inst_set` *`d_setObj`*	Removes the pin halo shapes on all terminals for selected instances.
`-set` *`d_setObj`*	Removes the pin halo shapes on all terminals for selected nets.

Example

The following command removes the pin halo shapes on all the terminals of instances with names beginning with `FINST`.

```
delete_pin_halo_shapes [ -inst_set [find_instance -window_id 1 -name "FINST*"]]
```

Related Information

Tcl Commands	create_pin_halo_shapes

delete_trim_fill_shapes

```
delete_trim_fill_shapes
     [ -layer { all | {s_layerName …} } ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
```

Deletes shapes that are created temporarily to prevent `minEndOfLineSpacing` violations when aligning trims.

Arguments

`-layer { all | {`*`s_layerName`*` …} }`

Restricts processing to the specified layers. By default, and when `all` is specified, all routing layers are processed.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the boundary of the area to process, given the lower-left and upper-right coordinates. By default, the entire design is processed.

Example

The following example deletes all the trim fill shapes that are inside the given region.

```
delete_trim_fill_shapes -region {100.1 100.1 150.4 18.80}
```

Related Information

Tcl Commands	extend_pins

deselect_set

```
deselect_set
```

Clears all objects from the selected set.

Arguments

None

Example

```
select_all
deselect_set
```

This example selects all the active shapes/objects and then deselects them.

filter_set

```
filter_set
     -set d_setObj
     -expr s_expression | -expr_ex s_extendedExpr
```

Returns a set containing objects from the original set that meet the criteria given by a simple expression (`-expr`) or a Tcl expression (`-expr_ex`) which compares property values for objects in the given set to target values. A number of operators are available.

Important

The syntax for simple expressions differs from the Tcl syntax. Basic syntax differences are shown in Table 3-1 on page 118.

Table 3-1 Basic Syntax Differences for filter_set Expressions

	Simple Expression (`-expr`)	Tcl Syntax (`-expr_ex`)
Property name	Name only for example, `direction`	Name must be preceded by a dollar sign ($); for example, `$direction`
String Value	String only for example, `WEST`	String must be enclosed in double quotes ("); for example, `"WEST"`

Arguments

`-expr` *`s_expr`*

Is an expression in the form:

`<propertyName> <operator> <targetValue>`

where <operator> is one of the following:

`==`	is equal to
`!=`	is not equal to
`>`	is greater than
`>=`	is greater than or equal to
`<`	is less than
`<=`	is less than or equal to
`=~`	regular expression matches pattern
`!~`	regular expression does not match pattern

The property name can be a simple property name, (for example, `direction)`, or a compound property name (for example, `net.name`). A numeric comparison is performed if the object's property value and the target value are both numbers, otherwise string comparisons are performed. For the regular expression operators (=~ and !~), the target value is a regular expression. All string comparisons are case-sensitive.

`-expr_ex` *`s_extendedExpr`*

Is an expression in Tcl syntax. Single or compound expressions can be used.

When the Tcl syntax expression is used, property names must be preceded by a dollar sign ($) and string target values must be enclosed in double quotes (“) to ensure that the expressions are parsed correctly.

`-set` *`d_setObj`*

Specifies the set of objects to operate on.

Value Returned

d_setObj	Is the set of objects that meet the expression criteria. If any object in the original set does not have the given property, an error message is issued and a NULL set is returned.

Example

Example 1—Simple expression using -expr

The following command evaluates objects in the selected set and returns a set containing all objects whose `direction` is `WEST.`

```
filter_set -set [get_selection_set] -expr {direction == WEST}
```

Note: If any object in the selected set does not have a `direction` property, the command will fail and return a NULL set.

Example 2—Simple expression using -expr_ex

The following command will return the same set of objects as Example 1—Simple expression using -expr but this command uses the `-expr_ex` syntax instead.

```
filter_set -set [get_selection_set] -expr_ex {$direction == "WEST"}
```

Example 3—Compound expression using -expr_ex

The following command returns the set of objects whose `length` is greater than 3 and whose `direction` ends with "TH".

```
filter_set -set $set1 -expr_ex {$length > 3 && [string match "*TH" $direction]}
```

find_bounds

```
find_bounds
     -set d_setObj
     [ -lpp {s_layerlpp…} ]
```

Returns a bounding box of the geometry in the given set. You can optionally restrict the geometry to shapes on the given layers and/or layer purposes.

Arguments

`-lpp` **{***s_layerlpp…***}**	Limits the geometry to the layers and/or layer purposes in the given list.
`-set` *d_setObj*	Specifies the object set to operate on.

Value Returned

f_xlo f_ylo f_xhi f_yhi	
	Are the bounding box coordinates, represented as strings.

Example

The following example gets the bounding box for objects in the selected set and displays the box as a rectangle annotation.

```
add_rectangle -rect [find_bounds -set [get_selection_set]] -color cyan -lineWidth 1
```

find_bus

```
find_bus
     [ -ignore_case [ true | false ] ]
     [ -name s_busExpr ]
     [ -no_wildcard [ true | false ] ]
```

Returns a set that contains all the nets in the buses whose names match the given -`name` argument.

Arguments

`-ignore_case [ true | false ]`

If `true`, performs a case-insensitive search.

`-name` *`s_busExpr`*

Finds all nets in the buses whose names match the expression. The expression can include special characters described in "Pattern Matching" on page 134.

If this argument is not specified, the resultant set will include the nets in all buses.

`-no_wildcard [ true | false ]`

Disables all wildcard processing.

Value Returned

`s_netName…`

Returns the set of bus nets.

find_by_area

```
find_by_area
     -region {f_xlo f_ylo f_xhi f_yhi}
     [ -single ]
     [ -set d_setObj | -window_id i_windowID ]
     [ -fully_enclosed ]
```

Finds a set of objects in a given area of a specified set or window. You can specify additional criteria including whether the return objects must be fully enclosed in the search area, and whether to return a single object.

When searching for a single object, only the first object found that matches the search criteria is returned. Space-based Router and Chip Optimizer uses the following order when searching for objects: nets (highest level), routes and vias, other shapes (lowest level).

If neither the `-set` argument nor the `-window_id` argument is specified, the scope of the search is the top cellview in the active window.

Only objects that are marked visible and active in the Layer Object Display Panel are searched.

Arguments

`-region` {*`f_xlo f_ylo f_xhi f_yhi`*}

Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates for the search area.

`-fully_enclosed`

Limits the search to objects that are fully enclosed in the search area. When not specified, all objects that are at least partially inside the search area are considered by the search routine.

`-set` *`d_setObj`*

Limits the search to instances in the specified set.

`-single`

Specifies that the search be terminated when the first object matching the criteria is found. When not specified, all matching objects are included in the output set.

`-window_id` *`i_windowID`*

Limits the search to objects in the top cellview of the specified window.

Value Returned

`d_setObj`

Specifies the Open Access set identifier for the set containing the objects matching all the specified criteria.

Example

The following example returns a set containing any active and visible objects that are at least partially inside the given area of window 1.

```
find_by_area -region {100.1 100.1 150.4 18.80} -window_id 1
```

The following example creates a set, identified by Tcl variable `aset1`, that contains the first active and visible object from set `aset` that is completely inside the given area.

```
set aset1 [find_by_area -region {2031.2 2130.5 2190.8 2156.7} -single
-fully_enclosed -set $aset]
```

Tcl set variables can be passed to other Space-based Router and Chip Optimizer commands that use the `-set` argument. For more information on set functions, refer to Chapter 2, “Set Commands.”

find_by_index

```
find_by_index
     -region {f_xlo f_ylo f_xhi f_yhi}
     -indices {i_index …}
     [ -window_id i_windowID ]
```

Creates a set of objects in the given area for the top cellview of the active window or the given window. Space-based Router and Chip Optimizer finds the list of objects that meet the criteria determined by the routing object granularity (`net`, `route` or `connected shapes`). Active individual shapes in the given area are always included in the list. The `-indices` argument gives the index for each object in the list that you want to include in the new set. This command is used by the cycle select function.

Arguments

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates for the search area.

`-indices {i_index …}`	Specifies the index numbers for the objects from the list to include in the new set.
`-window_id i_windowID`	Limits the search to objects in the top cellview of the specified window. If this argument is not given, the active window is used.

Value Returned

`d_setObj`	Returns the identifier for the new set.

Example

The following command creates a list of active objects in the given area and adds the first object from the list (with an index of `0`) to the selected set.

```
replace_set -set1 [or_sets -set1 [get_selection_set] -set2 [find_by_index -region
{ 898.13 1794.88 898..5 1795.70 } -indices {0}]] -set2 [get_selection_set]
```

find_composite_net

```
find_composite_net
     [ -group s_groupName ]
     [ -ignore_case [ true | false ] ]
     [ -no_wildcard [ true | false ] ]
     [ -regex_style [ true | false ] ]
     [ -type s_type ]
```

Returns a set containing composite nets in the active design. The search can be limited to net groups matching a given name, optionally specified by a regular expression, using wildcards, and case-sensitive, and/or limited to a specific net group type.

Arguments

`-group` *`s_groupName`*

Specifies the net group name.

`-ignore_case [ true | false ]`

When `true`, performs a case-insensitive search.

`-no_wildcard [ true | false ]`

When `true`, performs an exact match of the string, ignoring wildcards.

`-regex_style [ true | false ]`

When `true`, performs a regular expression match.

`-type` *`s_type`*

Specifies the type of net group.

net_pair	Net pairs for pair routing (see Pair Routing Commands)
net_bundle	Net group for bus routing (bus_route)
net_shield_parallel	Net group for parallel shielding
net_shield_tandem	Net group for tandem shielding
net_shield_coaxial	Net group for coaxial shielding
net_match	Net group for matched length routing
group group_of_group	Group of nets for group-level constraints
net_symmetry symmetry	Net group for symmetry routing
pwrGnd pwr_ground	Power and ground nets
crosstalk cross_talk	Group of neighbor nets.

Example

The following command returns a set containing all composite nets in the design.

```
set composites [find_composite_net]
```

The following command replaces the selected set with all composite nets for busses in the design.

```
replace_set -set1 [find_composite_net -type net_bundle ] -set2 [get_selection_set]
```

Related Information

Tcl Command	create_group

find_group

```
find_group
     [ -name s_groupExpr ]
     [ -type {net_pair | net_bundle | net_shield_parallel | net_shield_tandem
     | net_shield_coaxial| net_match | group | symmetry | group_of_group
     | pwr_ground | cross_talk} ]
     [ -ignore_case [ true | false ] ]
     [ -members [ true | false ] ]
     [ -no_wildcard [ true | false ] ]
```

Returns a set containing the groups or group member nets that match the criteria.

Arguments

`-ignore_case [ true | false ]`

If `true`, performs a case-insensitive search.

`-members [ true | false ]`

If `true`, returns group member nets in the set instead of the net groups.

Default: `false`

`-name` *`s_groupExpr`*

Finds all groups whose names match the expression. The expression can include special characters described in “Pattern Matching” on page 134.

If this argument is not specified, all groups are considered.

`-no_wildcard [ true | false ]`

Disables all wildcard processing.

`-type` *`s_groupType`*

Specifies the type of group to search for.

`cross_talk` Net groups used for specifying good neighbors and bad neighbors.

`group` Net groups used for group-level constraints

`group_of_group` Net group of groups

`net_bundle` Bus routing net group

`net_match` Net groups for matched length, capacitance and resistance

`net_pair` Net pairs used for pair routing

`net_shield_coaxial`

Net groups used for coaxial shields

`net_shield_parallel`

Net groups used for parallel shields

`net_shield_tandem`

Net groups used for tandem shields

`symmetry` Symmetry group

Value Returned

`d_setObj`	Returns the identifier for the new set.

Related Information

Tcl Commands	create_group report_group

find_inst_term

```
find_inst_term
     [ -delimiter s_char ]
     [ -ignore_case [ true | false ] ]
     [ -instance_name s_instanceName ]
     [ -instance_path s_path ]
     [ -name s_termName ]
     [ -net {s_netName…} ]
     [ -no_wildcard [ true | false ] ]
     [ -silent [ true | false ] ]
```

Returns a set containing the instance terminals that satisfy the given conditions.

Arguments

`-delimiter` *`s_char`*

Specifies the hierarchical delimiter. If not specified, the default delimiter character for the active oaNamespace is used. This option cannot be used with wildcards.

`-ignore_case [ true | false ]`

If `true`, performs a case-insensitive search.

`-instance_name` *`s_instance_name`*

Specifies the instance.

`-instance_path` *`s_path`*

Specifies the path to the instance. This option cannot be used with wildcards.

`-name` *`s_termName`*

Specifies the name of the term.

`-net` **{***`s_netName`***…}**

Searches nets in the list.

`-no_wildcard [ true | false ]`

Disables all wildcard processing.

`-silent [ true | false ]`

Suppresses informational messages. By default, these messages are output.

Value Returned

`d_setObj`

Returns the identifier for the new set containing the instance terminals.

Related Information

Tcl Commands	find_net_portion

find_instance

```
find_instance
     [ -name s_instExpr ]
     [ -set d_setObj | -window_id i_windowID ]
     [ -ignore_case ]
     [ -no_wildcard ]
     [ -silent ]
```

Searches a particular cellview or an existing set of objects for instances with names that match the given regular expression. If neither the `-set` argument nor the `-window_id` argument is specified, the scope of the search is the active window.

Arguments

`-ignore_case` Performs a case-insensitive search.

`-name` *`s_instExpr`* Finds all instances whose names match the expression. The expression can include special characters described in “Pattern Matching” on page 134.

If this argument is not specified, the resultant set will include all instances within the specified scope.

`-no_wildcard` Disables all wildcard processing.

`-set` *`d_setObj`* Limits the search to instances in the specified set.

`-silent` Suppresses informational messages, such as the number of items found. By default, these messages are output.

`-window_id` *`i_windowID`* Limits the search to instances in the specified window.

Value Returned

`d_setObj` Returns the identifier for a set containing all instances with names that matched the given expression for the given set or cellview.

Pattern Matching

Expressions can include the following special characters:

`?` Matches any single character.

`*`	Matches any sequence of zero or more characters.
[*chars***]**	Matches any single character in *chars*. If *chars* contains a sequence of the form *a*-*x*, then any character between *a* and *x* (inclusively) will match.
\[*chars***]**	Matches *chars* string enclosed in square brackets.
`{a, b,…}`	Matches any of the strings *a*, *b*, … listed within the braces.

For example, if you have nets named `Foo`, `Food`, `Foobar`, `Foozy`, `Woozy`, `Doozy`, `ooze`, `a[17]`, and `foo[12]`, then the following table indicates which nets would match the given expressions.

Expression	Matches:
`Foo*`	`Foo`, `Food`, `Foobar`, `Foozy`
`?oozy`	`Foozy`, `Woozy`, `Doozy`
`[ADW]oozy`	`Woozy`, `Doozy`
`Foo{d,bar,cat}`	`Food`, `Foobar`
`*oo*`	`Foo`, `Food`, `Foobar`, `Foozy`, `Woozy`, `Doozy`, `ooze`, `foo[12]`
`?*oo?*`	`Food`, `Foobar`, `Foozy`, `Woozy`, `Doozy`, `foo[12]`
`[a-z]oo*`	`foo[12]`
`*\[1[789]]`	`a[17]`

Example

The following example searches window `1` for any instances whose names begin with the letters `NAND`. The resultant set of instances is stored in the Tcl variable `NAND`.

```
set NAND [find_instance -window_id 1 -name "NAND*"]
```

The following example searches the set `NAND` for any instances whose names begin with the letters `NAND` and end with the number 1. The resultant set of instances is stored in the Tcl variable `NAND1`. `NAND1` is a subset of `NAND`.

```
set NAND1 [find_instance -set $NAND -name "NAND*1"]
```

Tcl set variables can be passed to other Space-based Router and Chip Optimizer commands that use the `-set` argument. For more information on set functions, refer to Chapter 2, “Set Commands.”

Related Information

Menu Commands	*Edit—Find* (*Instance*)
Example	Examples in Chapter 22, “Using Tcl”

find_instance_of

```
find_instance_of
     [ -lib s_libExpr ]
     [ -cell s_cellExpr ]
     [ -view s_viewExpr ]
     [ -set d_setObj | -window_id i_windowID ]
     [ -find_radius f_userunit {-ref_pt {f_x f_y} | -ref_inst s_instName} ]
     [ -exclude_vias ]
     [ -ignore_case ]
     [ -no_wildcard ]
     [ -silent ]
```

Searches a particular cellview or an existing set of objects for instances of master cellviews with names that match the given expressions. If neither the `-set` argument nor the `-cell_view_id` argument is specified, the scope of the search is the active cellview. You can define the area of the search using the `-find_radius` argument to search within a specific radius of an instance or a given coordinate.

Arguments

`-cell` *`s_cellExpr`*
Finds all instances whose master cell names match the expression. The expression can include special characters described in “Pattern Matching” on page 134. If this argument is not specified, then the resultant set will include instances with any master cell name.

`-exclude_vias`
Excludes vias from consideration. By default, vias are included.

`-find_radius` *`f_userunit`*
Searches for instances whose bounding box falls within the given radius (in microns) from a reference given by one of the following:

`-ref_inst` *`s_instName`*
Uses the bounding box of the given instance as the reference.

`-ref_pt` **{***`f_x f_y`***}**
Uses the given coordinate (in microns) as the reference.

`-ignore_case`
Performs a case-insensitive search.

`-lib` *`s_libExpr`*
Finds all instances whose master library names match the expression. The expression can include special characters described in “Pattern Matching” on page 134. If this argument is not specified, then resultant set will include instances with any master library name.

`-no_wildcard`
Disables all wildcard processing.

`-ref_inst` *`s_instName`*
Refer to the description for `-find_radius`.

`-ref_pt` **{***`f_x f_y`***}**
Refer to the description for `-find_radius`.

`-set` *`d_setObj`*
Limits the search to instances in the specified set.

`-silent`
Suppresses informational messages, such as the number of items found. By default, these messages are output.

`-view` *`s_viewExpr`*
Finds all instances whose master view names match the expression. The expression can include special characters described in “Pattern Matching” on page 134. If this argument is not specified, then the resultant set will include instances with any master view name.

`-window_id` *`i_windowID`*	Limits the search to instances in the specified window.

Value Returned

`d_setObj`	Returns the identifier for a set containing all instances that matched the lib/cell/view criteria for the given set or cellview.

Example

The following example searches in the active window for any instances whose master cell names begin with the letters `corner`. The resultant set of instances is stored in the Tcl variable `master`.

```
set master [find_instance_of -cell corner*]
```

Tcl set variables can be passed to other Space-based Router and Chip Optimizer commands that use the `-set` argument. For more information on set functions, refer to Chapter 2, “Set Commands.”

The following example finds all `BUFA1` cells within a 20 micron radius of the bounding box of `inst_I2`.

```
find_instance_of -cell BUFA1 -ref_inst inst_I2 -find_radius 20
```

Related Information

Tcl Commands	find_instance get_midpoint
Menu Commands	*Edit—Find* (*Instance*)
Examples	Examples in Chapter 22, “Using Tcl”

find_layer

```
find_layer
     -layer s_layerName
```

Returns the object identifier for the given layer.

Arguments

`-layer` *`s_layerName`*	Specifies the name of the layer.

Value Returned

`d_setObj`	Is the object identifier for the layer.

Example

The following commands set the `myLayer` variable to the object identifier for layer `M6`, then uses that variable to get the name of the cut layer above `M6` .

```
set myLayer [find_layer -layer M6]
inspect_prop cutLayerAbove.name $myLayer
```

find_neighbor_nets

```
find_neighbor_nets
     -net s_netName
     -halo f_userunit
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -include_ground ]
     [ -include_power ]
     [ -shapes ]
```

Returns a set containing nets or shapes surrounding the given net.

Arguments

`-halo` *`f_userunit`*	Limits the search to the area surrounding the net within the given halo.
`-include_ground`	Includes ground nets in the search. By default, ground nets are not included.
`-include_power`	Includes power nets in the search. By default, power nets are not included.
`-net` *`s_netName`*	Specifies the name of the net to search around.
`-region` {*`f_xlo f_ylo f_xhi f_yhi`*}	
	Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates for the search area.
`-shapes`	Includes individual route shapes rather than entire nets in the returned set. By default, the set includes entire nets only.

Value Returned

`d_setObj`	Specifies the set identifier for the set of nets or shapes found.

Example

The following example searches in the current artwork view area for nets within 2 microns of the `ar0[9]` net and adds the found nets to the `neighbor_net` highlight set for display.

```
set mynet [find_neighbor_nets -net ar0\[9\] -halo 2 -region [get_window_area]]
add_highlight -name neighbor_net -set $mynet -color cyan
```

Related Information

Tcl Commands	find_net

find_net

```
find_net
     [ -name s_netExpr
        [ -net_type
     {[signal][power][ground][clock][tieOff][tieHi][tieLo][scan][analog][all]} ]
      | -inst s_instName [ -term s_termName ]
      | -route_spec s_routeSpec ]
     [ -set d_setObj
     | -window_id i_windowID x
     | -include_composite_nets [ true | false ] ]
     [ -ignore_case ]
     [ -no_wildcard ]
     [ -instance_path s_path ]
     [ -delimiter s_char ]
     [ -silent ]
```

Searches a particular window or an existing set of objects for one of the following:

- Nets with names that match the given expression and, optionally, of a certain type
- The net that connects to the term of the given instance
- Nets with the given routespec

If neither the `-set` argument nor the `-window_id` argument is specified, the scope of the search is the active window.

Arguments

`-delimiter` *`s_char`*

Specifies the hierarchical delimiter. If not specified, the default delimiter character for the active oaNamespace is used. A NULL string ("") can be specified to prevent the path from being decomposed and is useful for flattened designs. This option cannot be used with wildcards.

`-ignore_case`

Performs a case-insensitive search.

`-include_composite_nets [ true | false ]`

When set to `true`, includes composite nets in the search.

Default: `false`

`-inst` *`s_instName`*

Specifies the name of the instance. Searches for the net that connects to the given term of the instance if `-term` is specified; otherwise, searches for all the nets that connect to the instance.

`-instance_path` *`s_path`*

Specifies the instance path to the sub-occurrence.

`-name` *`s_netExpr`*

Finds all nets whose names match the expression or a space-delimited list of expressions. The expressions can include special characters described in “Pattern Matching” on page 134. If this argument is not included, then all nets within the specified scope are included in the resultant set.

`-net_type`

Finds all nets of the types specified.
Valid values include: `signal`, `power`, `ground`, `clock`, `tieoff`, `tieHi`, `tieLo`, `scan`, `analog`, and `all`.

`-no_wildcard`

Disables all wildcard processing. By default, wildcards can be used.

`-route_spec` *`s_routeSpec`*

Finds nets with the specified routespec.

`-set` *`d_setObj`*

Limits the search to nets in the specified set.

`-silent`

Suppresses informational messages, such as the number of items found. By default, these messages are displayed.

`-term` *`s_termName`*

Specifies a terminal name. Used with the `-inst` argument to search for the net that connects to the terminal of the given instance.

`-window_id` *`i_windowID`*

Limits the search to nets in the specified window.

Value Returned

d_setObj	Returns the identifier for a set containing all nets whose names matched the given expression for the given set or cellview.

Examples

The following example searches for any nets whose names contain the string `RESULTS_CONV_INST_n`. The resultant set of nets is stored in the Tcl variable `net`.

```
set net [find_net -cell_view_id 1 -net *RESULTS_CONV_INST_n*]
```

Tcl set variables can be passed to other Space-based Router and Chip Optimizer commands that use the `-set` argument. For more information on set functions, refer to Chapter 2, "Set Commands."

The following example returns an unfolded shape.

```
setvar db.user_namespace def
find_net -instance_path fb_io/Fbd_data\\\[47\\\]_pad_macro -name j_ShiftDr
-delimiter "" -no_wildcard
```

The following example highlights the net connected to the `A0` term of the `B_32_INST/i_9182/i_9187/i_217` instance.

```
add_highlight -color cyan -name termA -set [find_net -inst B_32_INST/i_9182/
i_9187/i_217 -term A0]
```

Related Information

Tcl Commands	get_connected_net
Menu Commands	*Edit—Find* (*Net*)
Examples	Examples in Chapter 22, "Using Tcl"

find_net_portion

```
find_net_portion
     {-term s_termName -instance s_instName | -source d_setObj}
     -sink_set d_setObj
```

Returns a set containing the subset of shapes of a net that are fed by the source, given by the instance and term or contained in the `source` set, and that feed the sinks in the specified `sink_set` and *no other sinks* in the net.

Arguments

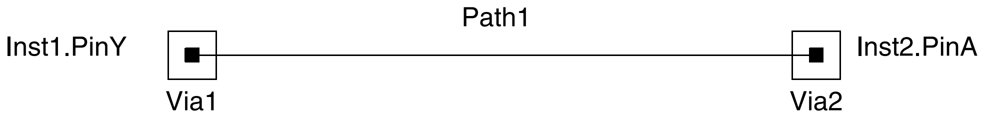
`-instance` *`s_instName`*	Specifies the name of the instance. Used with the `-term` argument to specify the source.
`-sink_set` *`d_setObj`*	Specifies the set containing sink shapes.
`-source` *`d_setObj`*	Specifies the set containing source shapes.
`-term` *`s_termName`*	Specifies a term name. Used with the `-inst` argument to specify the source.

Value Returned

`d_setObj`	Returns the identifier for a set containing all shapes in the net that are fed by the source and that feed the sinks in the specified set and no other sinks in the net.

Example

Net With One Source and One Sink

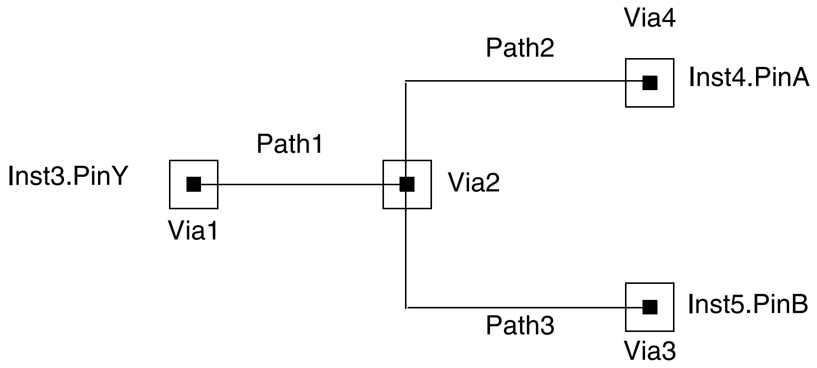


In this example, the selected set contains `Inst2.PinA`.

```
set netAp [find_net_portion -term Y -instance Inst1 -sink_set [get_selection_set]]
```

The output set contains `Via1`, `Path1`, and `Via2`.

Net With One Source and Two Sinks



In this example, the source is `Inst3.PinY,` and the sinks are `Inst4.PinA` and `Inst5.PinB`.

```
set source_set [find_inst_term -instance_name Inst3 -name Y]
set sink_setA [find_inst_term -instance_name Inst4 -name A]
set sink_setB [find_inst_term -instance_name Inst5 -name B]
set sink_setAB [or_sets -set1 $sink_setA -set2 $sink_setB]
```

The output set contains `Via1`, `Path1`, and `Via2` when both sinks are included in `sink_set`, as in the following command.

```
set np [find_net_portion -source $source_set -sink_set $sink_setAB]
```

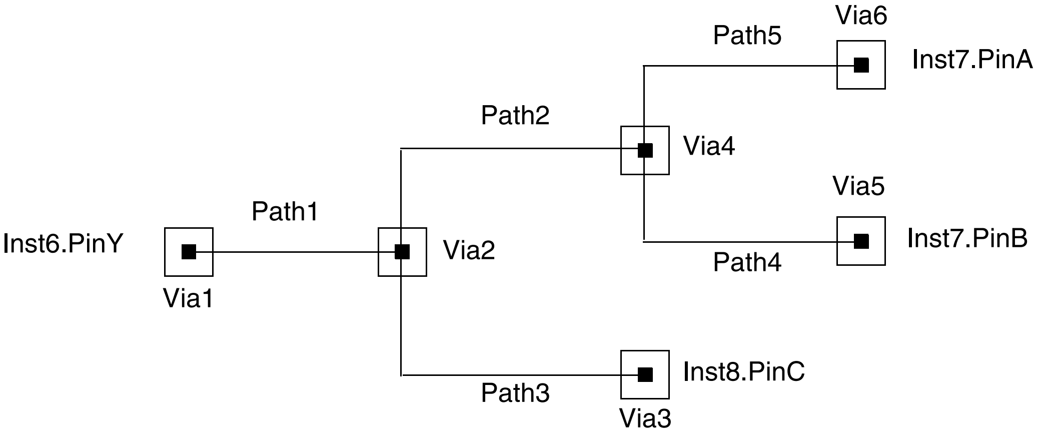
The output set contains `Path2` and `Via4` when only `Inst4.PinA` is included in `sink_set`, as in the following command.

```
set npA [find_net_portion -source $source_set -sink_set $sink_setA]
```

The output set contains `Path3` and `Via3` when only `Inst5.PinB` is included in `sink_set`, as in the following command.

```
set npB [find_net_portion -source $source_set -sink_set $sink_setB]
```

Net With One Source and Multiple Sinks



In this example, the source is `Inst6.PinY,` and the sinks are `Inst7.PinA`, `Inst7.PinB`, and `Inst8.PinC`.

```
set source_set [find_inst_term -instance_name Inst6 -name Y]
set sink_setA [find_inst_term -instance_name Inst7 -name A]
set sink_setB [find_inst_term -instance_name Inst7 -name B]
set sink_setC [find_inst_term -instance_name Inst9 -name C]
set sink_setAB [or_sets -set1 $sink_setA -set2 $sink_setB]
set sink_setAC [or_sets -set1 $sink_setA -set2 $sink_setC]
set sink_setABC [or_sets -set1 $sink_setAB -set2 $sink_setC]
```

The output set contains `Via1`, `Path1`, and `Via2` when all sinks are included in `sink_set`, as in the following command.

```
set npABC [find_net_portion -source $source_set -sink_set $sink_setABC]
```

The output set contains `Path5` and `Via6` when only `Inst7.PinA` is included in `sink_set`, as in the following command.

```
set npA [find_net_portion -source $source_set -sink_set $sink_setA]
```

The output set contains `Path2` and `Via4` when `Inst7.PinA` and `Inst7.PinB` are included in `sink_set`, as in the following command.

```
set npAB [find_net_portion -source $source_set -sink_set $sink_setAB]
```

The output set is empty when `Inst7.PinA` and `Inst8.PinC` are included in `sink_set`, as in the following command.

```
set npAC [find_net_portion -source $source_set -sink_set $sink_setAC]
```

In this case, there are no shapes that are common to `Inst7.PinA` and `Inst8.PinC` that are not common to `Inst7.PinB`.

Related Information

Tcl Commands	find_inst_term

find_scenic_nets

```
find_scenic_nets
     [ -all | -set d_setObj ]
     [ -exclude_set d_setObj ]
     [ -file s_fileName ]
     [ -max_fanout i_count ]
     [ -max_length f_userunit ]
     [ -max_ratio f_real_to_ideal ]
     [ -min_fanout i_count ]
     [ -min_length f_userunit ]
     [ -min_ratio f_real_to_ideal ]
     [ -report [ true | false ] ]
     [ -use_ideal [ true | false ] ]
     [ -use_mst [ true | false ] ]
```

(Virtuoso Routing IDE and Space-based Router only) Returns a set containing scenic nets in the entire cellview or in a given set, based on the given criteria for exact and ideal lengths.

A net is considered to be scenic if it meets all of the following conditions:

- If `-min_length` is set, net length >= `min_length`. By default, the net length is the exact length, but if `-use_ideal true` is given, the net's ideal length is checked.
- If `-max_length` is set, net length <= `max_length`. By default, the net length is the exact length, but if `-use_ideal true` is given, the net's ideal length is checked.
- If `-min_ratio` is set, exact length/ideal length >= `min_ratio`.
- If `-max_ratio` is set, exact length/ideal length <= `max_ratio`.
- If `-min_fanout` is set, fanout >= `min_fanout`.
- If `-max_fanout` is set, fanout <= `max_fanout`.

If none of the conditions is specified, an empty set is returned.

For all ideal lengths, use `-use_mst` to choose whether the length is calculated using a minimum spanning tree (default) or a Steiner tree.

The number of nets checked and the number of scenic nets found are output to the Transcript area. You can choose to have a detailed scenic net report output to the Transcript area (`-report`) or to a file (`-file`).

Arguments

-all
: (Default) Searches all nets except power nets. Cannot be used with -set.

-exclude_set *d_setObj*
: Excludes from the search the nets in this set.

-file *s_fileName*
: Outputs the scenic net details to the specified file.

-max_fanout *i_count*
: To be scenic, a net's fanout count must be less than or equal to this value.

-max_length *f_userunit*
: To be scenic, a net's length (in user units) must be less than or equal to this value. The -use_ideal argument determines whether the net's ideal or exact length is checked. By default, the exact length is checked.

-max_ratio *f__to_ideal*
: To be scenic, a net's exact length / ideal length must be less than or equal to this value. The -use_mst argument determines whether the ideal length is calculated using a minimum spanning tree or a Steiner tree.

-min_fanout *i_count*
: To be scenic, a net's fanout count must be greater than or equal to this value.

-min_length *f_userunit*
: To be scenic, a net's length (in user units) must be greater than or equal to this value. The -use_ideal argument determines whether the net's ideal or exact length is checked. By default, the exact length is checked.

-min_ratio *f__to_ideal*
: To be scenic, a net's exact length / ideal length must be greater than or equal to this value.

-report [true | false]
: Outputs the scenic net details to the Transcript area.

-set *d_setObj*
: Specifies the set of nets to operate on. Cannot be used with -all.

-use_ideal [true | false]

When set to `true`, use the ideal length for minimum and maximum length checks. The `-use_mst` argument determines whether the ideal length is calculated using a minimum spanning tree or a Steiner tree.

Default: (`false`) Use the exact length for minimum and maximum length checks.

`-use_mst [ true | false ]`

When set to `false`, use a Steiner tree to compute the ideal length.

Default: (`true`) Use the minimum spanning tree to compute the ideal length.

Returned Value

d_setObj	Set of scenic nets.
-1	A command syntax error occurred.

Example

The following example creates a set of nets in the active cellview that are at least 20 user units in length and twice as long as their ideal length using a minimum spanning tree calculation. A detailed report on the scenic nets is output to `scenic2.txt`.

```
set scNet [find_scenic_nets -all -min_length 20 -min_ratio 2 -file scenic2.txt]

Opened file scenic2.txt sucessfully for scenic net report

Scenic Net Report
------------------------------------------------------------------
NetName                             Length     ScenicRatio
------------------------------------------------------------------
MULT_32_INST/i_723/n_876                   22.44         2.08939
MULT_32_INST/i_723/n_741                   24.42         2.12903
------------------------------------------------------------------
 Visited 5694 nets, and found 2 scenic nets
sel:e8a0140
```

find_shape

```
find_shape
     [ -layer s_layerExpr ]
     [ -purpose s_purposeExpr ]
     [ -set d_setObj | -window_id i_windowID ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} | {{f_x1 f_y1}{f_x2 f_y2} … {f_xn f_yn}} ]
     [ -use_regions_on_scratch_layer i_scratchID ]
     [ -ignore_case ]
     [ -depth i_level ]
     [ -fully_enclosed [ true | false ] ]
     [ -granularity [shape | route | route_only | net] ]
     [ -no_net ]
     [ -no_wildcard ]
     [ -on_net ]
     [ -shape_types {[rectangle ] [octagon] [polygon] [routevia] [routeseg]
     [guide ] [annotation] [text]} ]
     [ -silent ]
```

Searches for shapes matching the given criteria.

Arguments

`-depth` *`i_level`*

Specifies the depth of the hierarchy to search. A value of 0 chooses the top-level only and is the default value.

`-fully_enclosed [ true | false ]`

When `true`, only finds shapes that are fully enclosed in the specified region.

`-granularity` *`s_option`*

When a routing shape is found by this command, this argument chooses the objects to include in the return set.

- `net` Chooses the routing shape's net. This is the same functionality as interactive *Routing Object Granularity—Entire Net*.
- `route` Chooses the routing shape's route, including connected pins or terms under the route.
- `route_only` Chooses the routing shape's route. This is the same functionality as interactive *Routing Object Granularity—Entire Route*.
- `shape` (Default) Chooses the shape. This is the same functionality as interactive *Routing Object Granularity—Shape or Via*.

`-ignore_case`

Performs a case-insensitive search.

`-layer` *`s_layerExpr`*

Finds all shapes existing on layers whose names match the given expression. The expression can include special characters described in "Pattern Matching" on page 134. If this argument is not specified, all shapes on all layers within the specified scope meet the criteria.

`-no_net`

Restricts search to shapes that have no nets assigned to them.

`-no_wildcard`

Disables all wildcard processing.

`-on_net`

Restricts search to shapes that are on nets.

`-purpose` *`s_purposeExpr`*

Finds all shapes existing on layer purposes whose names match the given expression. The expression can include special characters described in "Pattern Matching" on page 134. If this argument is not specified, all shapes names on all purposes within the specified scope meet the criteria.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}** | **{{***f_x*$_1$ *f_y*$_1$**}** **{***f_x*$_2$ *f_y*$_2$**}** … **{***f_x*$_n$ *f_y*$_n$**}}**

Specifies the search area using either the lower-left (*f_xlo*, *f_ylo*) and the upper-right (*f_xhi*, *f_yhi*) bounding box coordinates, or the x and y coordinate points for a polygon boundary. If you use the `-set` argument, the -region is ignored.

`-set` *d_setObj* Limits the search to shapes in the specified set.

`-shape_types` **{**[rectangle] [octagon] [polygon] [routevia] [routeseg] [guide] [annotation] [text]**}**

Specifies one or more shape types to search for. If this argument is not given, all shapes are considered.

`-silent` Suppresses informational messages, such as the number of items found. By default, these messages are output.

`-use_regions_on_scratch_layer` *i_scratchID*

Finds shapes that overlap or are fully enclosed (if `-fully_enclosed` is true) by shapes on the given scratch layer. By defining shapes on a scratch layer, you can effectively search multiple rectangular or rectilinear regions. To specify shapes on a scratch layer, refer to "geom_add_shape" on page 316.

`-window_id` *i_windowID* Limits the search to shapes in the specified window.

Value Returned

d_setObj	Returns the identifier for a set containing all shapes that meet the criteria within the specified set or cellview.

Example

The following example searches for shapes on layer `met1` with purpose `wire:detail` and saves the found shaped in the set named `met_wire_detail`.

```
set met_wire_detail [find_shape -layer met1 -purpose wire:detail -cell_view_id
[get_cell_view_id]]
```

Related Information

Menu Commands	*Edit—Find* (*Shape*)

find_shielded_nets

```
find_shielded_nets
     [ -type {net_shield_parallel | net_shield_tandem | net_shield_coaxial} ]
```

Returns a set of nets that have been identified as shielded nets, optionally limited to a shield type.

Arguments

`-type {net_shield_parallel | net_shield_tandem | net_shield_coaxial}`

Limits the nets to the specified shield type.

Value Returned

d_setObj	Is the set containing the shielded nets.

Example

The following command sets the `shieldNets` variable with a set containing nets with parallel shields.

```
set shieldNets [find_shielded_nets -type net_shield_parallel]
```

Related Information

Tcl Commands	shield_net

find_terminal

```
find_terminal
     [ -name s_termExpr ]
     [ -set d_setObj | -window i_windowID ]
     [ -ignore_case ]
     [ -silent ]
```

Returns a set containing terminals in the given set or in a window.

Arguments

`-ignore_case`	Performs a case-insensitive search. If not specified, the search is case-sensitive.
`-name` *`s_termExpr`*	Limits the search to terminals matching the given expression. If not specified, the search is for all terminals.
`-set` *`d_setObj`*	Limits the search to objects contained in the specified set. If neither `-set` nor `-window` is specified, the entire design in the active window is searched.
`-silent`	Suppresses informational messages, such as the number of items found. By default, these messages are output.
`-window_id` *`i_windowID`*	Specifies the window to use. If this is not specified, the active window is used.

Value Returned

`d_setObj`	Specifies the set identifier for the set of terminals found that match the criteria.

Example

The following example searches the entire design for terminals named t_*, adds the objects to the set identified by variable `myt_`, then replaces the selected set with the new set of objects for display. You can use the Properties form to navigate the terminals found.

```
set myt_ [find_terminal -name t_*]
replace_set -set1 $myt_ -set2 [get_selection_set]
```

Related Information

Menu Commands	*Edit—Find* (*Terminal*)

find_text

```
find_text
     [ -value s_textExpr ]
     [ -set d_setObj | -window i_windowID ]
     [ -ignore_case ]
     [ -silent ]
```

Returns a set containing text objects in the given set or in a window.

Arguments

`-ignore_case`	Performs a case-insensitive search. If not specified, the search is case-sensitive.
`-set` *`d_setObj`*	Limits the search to objects contained in the specified set. If neither `-set` nor `-window` is specified, the entire design in the active window is searched.
`-silent`	Suppresses informational messages, such as the number of items found. By default, these messages are output.
`-value` *`s_textExpr`*	Limits the search to text matching the given expression. If not specified, the search is for all text objects.
`-window_id` *`i_windowID`*	Limits the search to text in the given window. If neither `-set` nor `-window_id` is given, the search is for text in the active window.

Value Returned

`d_setObj`	Specifies the set identifier for the set of text objects found that match the criteria.

Example

The following example searches the entire design for all text objects named `lbl*`, adds the objects to the set identified by variable `mylbl`, then replaces the selected set with the new set of objects. You can use the Properties form to navigate the set of text objects that are found.

```
set mylbl [find_text -value lbl*]
replace_set -set1 $mylbl -set2 [get_selection_set]
```

Related Information

Tcl Commands	create_label
Menu Commands	*Edit—Find* (*Text*)

flatten

```
flatten
     -set d_setObj
     [ -preserve_nets [ true | false ] ]
     [ -verbose ]
```

Flattens instances in the given set.

Arguments

`-preserve_nets [ true | false ]`

If `true` and the master being flattened was originally a cut-out from the top-level, nets are preserved. If nets cannot be preserved when flattening, use extract_net_connectivity to establish connectivity.
Default: `false`

`-set` *`d_setObj`* Specifies the set containing the instances to flatten.

`-verbose [ true | false ] ]`

Outputs information about each cell extracted.
Default: `false`

ge_to_rde_selection

```
ge_to_rde_selection
     [ -nets ]
     [ -set d_setObj [ -replace ] ]
```

(Virtuoso Routing IDE only) Adds selected objects in the Virtuoso Layout Suite XL/GXL (VLS XL/GXL) Editor canvas to a set in Space-based Router and Chip Optimizer. You can optionally add nets attached to the selected Virtuoso objects instead of adding the objects themselves.

Arguments

`-nets`	If this argument is given, only nets that are attached to the selected objects in the VLS XL/GXL Editor canvas are added to the Space-based Router and Chip Optimizer set. If this argument is not given, only the selected VLS XL/ GXL Editor objects are added to the Space-based Router and Chip Optimizer set.
`-replace`	(Applies only when `-set` is given) If specified, replaces existing objects in the given Space-based Router and Chip Optimizer set with the selected Virtuoso objects or nets. By default, the selected Virtuoso objects or nets are added to the Space-based Router and Chip Optimizer set.
`-set` *`d_setobj`*	Specifies the set to which the selected objects or nets will be added. If this argument is not given, a new set is created that contains the selected objects or nets.

Value Returned

0	Command was successful.
-1	Command failed.
`d_setObj`	(When `-set` is not given) Is the set identifier for the new set containing the selected objects or nets.

Example

The following example adds the currently selected objects in the VLS XL/GXL Editor canvas to the Space-based Router and Chip Optimizer selected set.

```
ge_to_rde_selection -set [get_selection_set]
```

The following example creates a new set containing the nets attached to the objects that are currently selected in the VLS XL/GXL Editor canvas.

```
set ge_set [ge_to_rde_selection -nets]
```

Related Information

Tcl Commands	hsm_to_rde_selection

get_cell_view_id

```
get_cell_view_id
     [ -window_id i_windowID ]
```

Returns the OpenAccess (OA) identifier for the cellview in the active window or the specified window.

Arguments

`-window_id` *`i_windowID`*	Specifies the window to use. If this is not specified, the active window is used.

Value Returned

`d_oaObj`	Specifies the OA identifier for the cellview.

Example

The following example stores the OA identifier for the cellview in the active window in the Tcl variable `myView`.

```
set myView [get_cell_view_id]
```

get_connected_net

```
get_connected_net
     -inst d_ctuObj
     -term s_termName
```

Returns the object identifier for the net connected to the given term of the instance.

Arguments

`-inst` *`d_ctuObj`*	Is the object identifier for the instance.
`-term` *`s_termName`*	Specifies the name of the term.

Value Returned

`d_ctuObj`	Specifies the object identifier for the net.

Example

The following command adds to the `HL1` highlight set the net connected to the `A1` term of instance in the selected set.

```
add_object_to_set -set [get_highlight -name HL1] -object [get_connected_net \
  -term "A1" -inst [ml [get_selection_set]]]
```

Related Information

Tcl Commands	find_net

get_current_highlight

```
get_current_highlight
```

Returns the set identifier for the current highlight set.

Arguments

None

Value Returned

d_setObj	Specifies the set identifier for the current highlight set.

Example

The following example stores the set identifier for the current highlight set in the `myset` variable.

```
set myset [get_current_highlight]
```

Related Information

Tcl Commands	get_highlight get_num_highlights

get_element

```
get_element
     s_string
     s_name
```

Parses a string of the form:

"*name1*:*value1*,*name2*:*value2*,*name3*:*value3…*"

to return the value for the named element.

The shorthand version of this command, `gelt`, can also be used.

Arguments

s_name	Specifies the name.
s_string	Specifies the string to be parsed.

Value Returned

s_value	Is the value for the named element.

Example

The following command shows the bounds returns a string representing the bounds of a route segment.

```
ip bounds [get_selection_set]
"xlo: 570.42, ylo: 567.12, xhi: 589.86, yhi: 567.42"
```

Using the previous example, the following command returns the value for the `xlo` element.

```
get_element [inspect_prop bounds [get_selection_set]] xlo
570.42
```

The following command uses shorthand names for `get_element`(`gelt`) and `inspect_prop` (`ip`).

```
gelt [ip bounds [get_selection_set]] xlo
570.42
```

Related Information

Tcl Commands	inspect_prop

get_highlight

```
get_highlight
     -set_number i_index | -name s_hlName
```

Returns the set identifier for the set that matches the given name or highlight set number.

Arguments

`-name s_hlName`	Specifies the set name to search for.
`-set_number i_index`	Specifies the set number of the set to return. **Note:** You can use get_num_highlights to get the total number of highlight sets currently available. Sets are numbered `0` to `get_num_highlights-1`.

Value Returned

d_setObj	Specifies the highlight set that matches the criteria.
`-1`	Highlight set matching the criteria has not been not found.

Example

The following example stores the set identifier for the highlight set `hiset2` in the `myset` variable.

```
set myset [get_highlight -name hiset2]
```

Related Information

Tcl Command	get_num_highlights

get_inst_headers

```
get_inst_headers
```

Returns a list of the library, cell and view names for the instance masters used in the active design.

Arguments

None

Value Returned

`"`*`s_layerName/s_cellName/s_viewName`*`"…`	
	List of library, cell and view names for the instance masters used in the active design.

Example

The following example requests the list of instance masters used in the design and shows the results.

```
get_inst_headers
"myLib/AND2X1/abstract" "myLib/ssad2/abstract" "myLib/ssao211/abstract"
```

Related Information

Tcl Commands	get_term_width

get_midpoint

```
get_midpoint
     -set d_setObj
```

Returns the midpoint of the collective area represented by the bounding boxes of all instances in the given set. If the set does not contain at least one instance, an error message is issued.

Arguments

`-set` *`d_setObj`*	Specifies the set of objects to use for the calculation.

Value Returned

`f_x f_y`	Indicates the midpoint of the collective area of bounding boxes for instances in the given set.
`-1`	The midpoint was not calculated.

Example

The following example finds all cells whose master name begins with `BUFA` within a 20.6 micron radius from the collective midpoint of all cells whose master name begins with `DFFB`.

```
find_instance_of -cell BUFA* -ref_pt [get_midpoint -set [find_instance_of -cell
DFFB*]] -find_radius 20.6
```

Related Information

Tcl Commands	find_instance_of

get_num_highlights

```
get_num_highlights
```

Returns the current number of sets that are highlighted. Sets are numbered starting from 0.

Arguments

None

Value Returned

`i_count`	Specifies the current number of sets that are highlighted.
`0`	No highlight sets found.

Example

The following example gets the current number of highlight sets and saves the value in the Tcl variable `hi_count`.

```
set hi_count [get_num_highlights]
```

Related Information

Tcl Commands	get_highlight

get_occurrence

```
get_occurrence
     -lib s_libName
     -cell s_cellName
     -view s_viewName
     [ -mode {r | w | a} ]
```

Opens a cellview and creates an *objectID* for the cellview. If the cellview is already open and the occurrence is already open, the objectID is returned but the cellview is not re-opened.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*	
	Specifies the cellview to open.
`-mode {r	w	a}`	Specifies the editing mode: `r` (opens for read only), `w` (opens for write and erases all the data in the cellview), `a` (opens for append or edit).

Value Returned

`d_ctuObj`	Is the identifier for the occurrence.
`-1`	Cellview not found.

Example

The following example creates an objectID for the `small8000/small8k_routed/layout` cellview, then opens a property inspector dialog to view the occurrence properties.

```
set occ [get_occurrence -lib small8k -cell small8k_routed -view layout]
inspect -object $occ
```

Related Information

Tcl Commands	cellview_exists get_occurrence_instances get_occurrence_nets get_occurrence_terms

get_occurrence_instances

```
get_occurrence_instances
     -occurrence d_ctuObj
```

Creates a set of all instances for the given occurrence.

Arguments

`-occurrence` *`d_ctuObj`*	Specifies the identifier for the occurrence.

Value Returned

`d_setObj`	Is the identifier for the set that contains all instances for the occurrence.

Example

The following example creates a set of all instances for the `small8000/small8k_routed/layout` cellview, then opens a property inspector dialog to view the instance properties.

```
set occ [get_occurrence -lib small8k -cell small8k_routed -view layout]
set instances [get_occurrence_instances -occurrence $occ]
inspect -set $instances
```

Related Information

Tcl Commands	get_occurrence

get_occurrence_nets

```
get_occurrence_nets
     -occurrence d_ctuObj
```

Creates a set of all nets for the given occurrence.

Arguments

`-occurrence` *`d_ctuObj`*	Specifies the identifier for the occurrence.

Value Returned

`d_setObj`	Is the identifier for the set that contains all nets for the occurrence.

Example

The following example creates a set of all nets for the `small8000/small8k_routed/layout` cellview, then opens a property inspector dialog to view the net properties.

```
set occ [get_occurrence -lib small8k -cell small8k_routed -view layout]
set nets [get_occurrence_nets -occurrence $occ]
inspect -set $nets
```

Related Information

Tcl Commands	get_occurrence

get_occurrence_terms

```
get_occurrence_terms
     -occurrence d_ctuObj
```

Creates a set of all terms for the given occurrence.

Arguments

`-occurrence` *`d_ctuObj`*	Specifies the identifier for the occurrence.

Value Returned

`d_setObj`	Is the identifier for the set that contains all terms for the occurrence.

Example

The following example creates a set of all terms for the `small8000/small8k_routed/layout` cellview, then opens a property inspector dialog to view the term properties.

```
set occ [get_occurrence -lib small8k -cell small8k_routed -view layout
set terms [get_occurrence_terms -occurrence $occ
inspect -set $terms
```

Related Information

Tcl Commands	get_occurrence

get_selection_set

```
get_selection_set
```

Returns the selected set identifier for the active window.

Arguments

None

Value Returned

`d_setObj`	Is the set identifier for the selected set in the active window.
`-1`	There is no active window.

Example

```
select_all
set cur_set [get_selection_set]
```

The first line selects all active layers and objects. The second line calls `get_selection_set` which returns the set identifier for the selection set and, in this example, assigns it to the Tcl variable `cur_set`.

get_term_width

```
get_term_width
     -cell s_cellName
     -layer s_layerName
     -lib s_libName
     -term s_termName
     -noMerge [ true | false ] ]
     -verbose [ true | false ] ]
     -view s_viewName
```

Used to determine the width of a pin.

Note: For compound pins it is the widest part of the pin.

Arguments

`-cell` *`s_cellName`* Specifies the name of the cell for the cellview.

`-layer` *`s_layerName`* Specifies the name of the layer for the cellview.

`-lib` *`s_libName`* Specifies the name of the library for the cellview.

`-term` *`s_termName`* Specifies the name of the term.

`-noMerge [ true | false ] ]`

Skips merging of abutted shapes.
Default: `false`

`-verbose [ true | false ] ]`

Debugs information about each cell extracted.
Default: `false`

`-view` *`s_viewName`* Specifies the name of the view in the given library.

Example

Returns the width of pin `B` on `M1` of the device.

```
get_term_width -term B -lib cmos14_std_cells -cell CW_NAND2_X1M_A10TH -view
abstract -layer M1
```

get_via_headers

```
get_via_headers
```

Returns a list of the library, cell and view names for the via masters used in the active design.

Arguments

None

Value Returned

"*s_libName/s_cellName/s_viewName*"…	List of library, cell and view names for the via masters used in the active design.

Example

The following example requests the list of via masters that are used in the active design and shows the results.

```
get_via_headers
"mylib/V001/via" "mylib/V002/via" "mylib/V231/via"
```

Related Information

Tcl Commands	get_inst_headers

highlight

```
highlight
     [ -once | -repeat]
```

Sets the mouse command field to `highlight` and allows you to interactively highlight objects.

Arguments

`-once`	Permits only one highlight, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits consecutive highlights, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Commands	get_current_highlight
Menu Commands	*Edit—Highlight*

hp

```
hp
     s_propName
     [ d_setObj | {d_ctuObj…} ]
```

Indicates whether the specified property exists for objects in a set or objects from a list of object references. If no objects are given, the objects in the selected set are inspected.

If the property name is a dot-separated list of names, the name is treated specially, recursively processing for each element of the chain of names for property values. The following command checks for the `name` property on the grandfather of the selected object:

```
hp name [ip parent [ip parent]]
```

This command can also be given as

```
hp parent.parent.name
```

Note: In these examples, the shorthand name for inspect_prop (`ip`) is used.

Arguments

`{`*`d_ctuObj…`*`}`	Specifies a list of one or more object identifiers.
`d_setObj`	Specifies the identifier for a set of objects.
`s_propName`	Specifies the name of the property.

Value Returned

`true | false`	Indicates whether the given property exists for all of the objects.

Related Information

Tcl Commands	inspect_prop

hsm_to_rde_selection

```
hsm_to_rde_selection
     {[ -nets ] [ -instances ]}
     [ -set d_setObj [ -replace ] ]
```

(Virtuoso Routing IDE only) Adds nets and/or instances that are selected using the Virtuoso Layout Suite XL/GXL `hsmSelect` SKILL command or with the VLS XL/GXL Navigator to a Space-based Router and Chip Optimizer set.

Arguments

`-instances`	Selected instances will be added to the Space-based Router and Chip Optimizer set.
`-nets`	Selected nets will be added to the Space-based Router and Chip Optimizer set.
`-replace`	(Applies only when `-set` is given) If specified, replaces existing objects in the specified Space-based Router and Chip Optimizer set with the selected Virtuoso objects. By default, the selected Virtuoso objects are added to the specified Space-based Router and Chip Optimizer set.
`-set` *`d_setobj`*	Specifies the Space-based Router and Chip Optimizer set to which the selected Virtuoso objects will be added. If this argument is not given, a new set is created that contains the selected objects.

Value Returned

0	Command was successful.
-1	Command failed.
`d_setObj`	(When `-set` is not given) Is the set identifier for the new set containing the selected objects.

Example

The following example replaces objects in the Space-based Router and Chip Optimizer selected set with the currently selected VLS XL/GXL Navigator nets.

```
hsm_to_rde_selection -nets -set [get_selection_set] -replace
```

The following example creates a new set containing the nets attached to the currently selected VLS XL/GXL Navigator objects.

```
set hsm_set [hsm_to_rde_selection -nets -instances]
```

Related Information

Tcl Commands	ge_to_rde_selection

icopy

```
icopy
     [ -once | -repeat ]
```

Sets the mouse command field to `icopy` and allows you to interactively copy routes.

Arguments

`-once`	Permits only one copy, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits consecutive copies, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Commands	copy_route
Menu Commands	*Edit—Copy*

icut

```
icut
     [ -once | -repeat ]
```

Sets the mouse command field to `icut` and allows you to interactively choose a cut location.

Arguments

`-once`	Permits only one cut, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits consecutive cuts, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Commands	cut
Menu Commands	*Edit—Cut*

imove

```
imove
     [ -once | -repeat ]
```

Sets the mouse command field to `imove` and allows you to interactively move or slide instances, rectangles, vias, route segments, entire routes and entire nets.

Arguments

`-once`	Permits only one move, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits consecutive moves, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Commands	move
Menu Commands	*Edit—Move*

inspect

```
inspect
     [ -set d_setObj | -object d_ctuObj ]
     [ { [ -backward_page | -forward_page
      | -previous_set_item | -next_set_item | -set_item_index i_index
      | -capture
      | -raise
      | -set_link_type {"New Window" | "Current Window"}
      | -delete
      | -visit -link_index i_index
      | -set_prop -property_name s_propName -new_value s_propValue
        -link_index i_index ] } -id i_inspectorID ]
```

Opens the Properties dialog for the given object, the given set or the selected set. If no object is selected and neither `-object` nor `-set` is given, the entire design is used. This command is also used to indicate actions performed in the Properties dialog, for example, capture and setting properties.

Arguments

`-backward_page`
Scrolls back one page in the Properties browser history for the inspector given by the `-id` argument.

`-capture`
Outputs the current properties to the Transcript area for the inspector given by the `-id` argument.

`-delete`
Closes the browser for the inspector given by the `-id` argument.

`-forward_page`
Scrolls forward one page in the Properties browser history for the inspector given by the `-id` argument.

`-id` *`i_inspectorID`*
Indicates the inspector to use for the current command.

`-next_set_item`
Displays the properties for the next item for the inspector given by the `-id` argument.

`-object` *`d_ctuObj`*
Specifies the identifier for the object to inspect, or any output from inspect_getprop.

`-raise`
Opens the Properties dialog for the inspector given by the `-id` argument.

`-set` *`d_setObj`*
Specifies the identifier for the set to inspect. By default, the selected set is inspected.

`-set_item_index` *`i_index`*
Displays the properties for the given indexed item of the inspector given by the `-id` argument.

`-set_link_type` *`s_linkType`*
Determines whether a new window will be opened to display properties associated with a *linked* object in the current window.

`Current Window` The properties for the linked object will be displayed in the current window.

`New Window` The properties for the linked object will be displayed in a new window.

`-set_prop -property_name` *`s_propName`* `-new_value` *`s_propValue`* `-link_index` *`i_index`*
Sets the value of a property for the current item of the inspector given by `-id`.

`-property_name`	Is the name of the property.
`-new_value`	Is the new value for the property.
`-link_index`	Is the index of the property in the current item’s property list.

`-visit -link_index` *`i_index`*

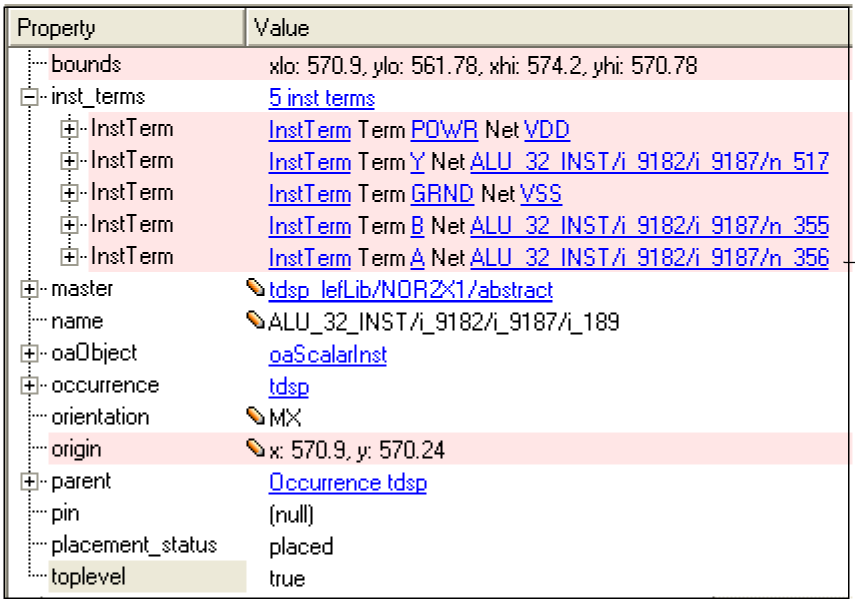
Opens the Property browser for the inspector given by `-id` to the linked object given by the *`i_index`* property for the current item.

Value Returned

`i_inspectorID`	Indicates the identifier for the property inspector when `-set`, `-object` or no argument is given.
`0`	Indicates that the command completed successfully when arguments other than `-set` or `-object` are given.
`-1`	Indicates there was a syntax error and the command was not completed.

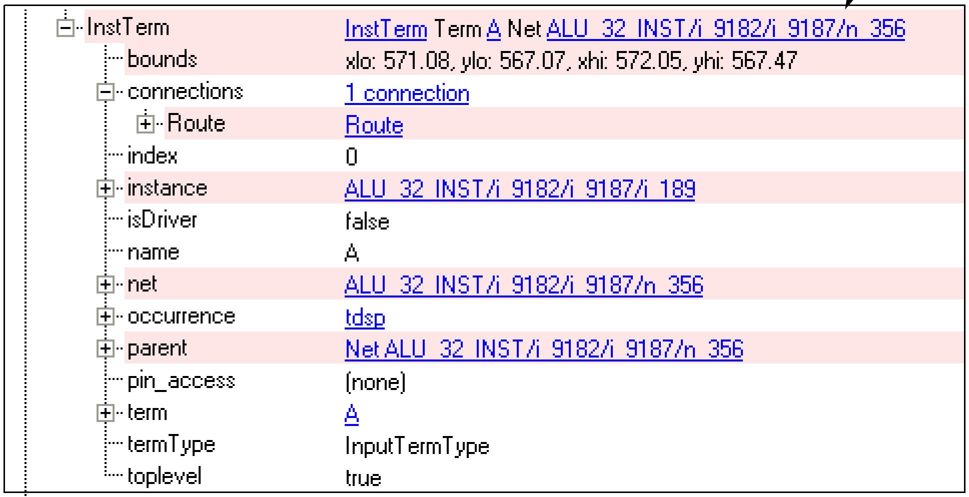
Example

The following figure shows properties for an instance in the Properties browser. The `link_index` values for the instance properties, and for the expanded view of the property with `link_index` value of `11.1`are given.



link_index	Property	Value
12	bounds	xlo: 570.9, ylo: 561.78, xhi: 574.2, yhi: 570.78
11	⊟ inst_terms	5 inst terms
11.5	⊞ InstTerm	InstTerm Term POWR Net VDD
11.4	⊞ InstTerm	InstTerm Term Y Net ALU 32 INST/i 9182/i 9187/n 517
11.3	⊞ InstTerm	InstTerm Term GRND Net VSS
11.2	⊞ InstTerm	InstTerm Term B Net ALU 32 INST/i 9182/i 9187/n 355
11.1	⊞ InstTerm	InstTerm Term A Net ALU 32 INST/i 9182/i 9187/n 356
10	⊞ master	✎ tdsp lefLib/NOR2X1/abstract
9	name	✎ ALU_32_INST/i_9182/i_9187/i_189
8	⊞ oaObject	oaScalarInst
7	⊞ occurrence	tdsp
6	orientation	✎ MX
5	origin	✎ x: 570.9, y: 570.24
4	⊞ parent	Occurrence tdsp
3	pin	(null)
2	placement_status	placed
1	toplevel	true

expand
property 11.1



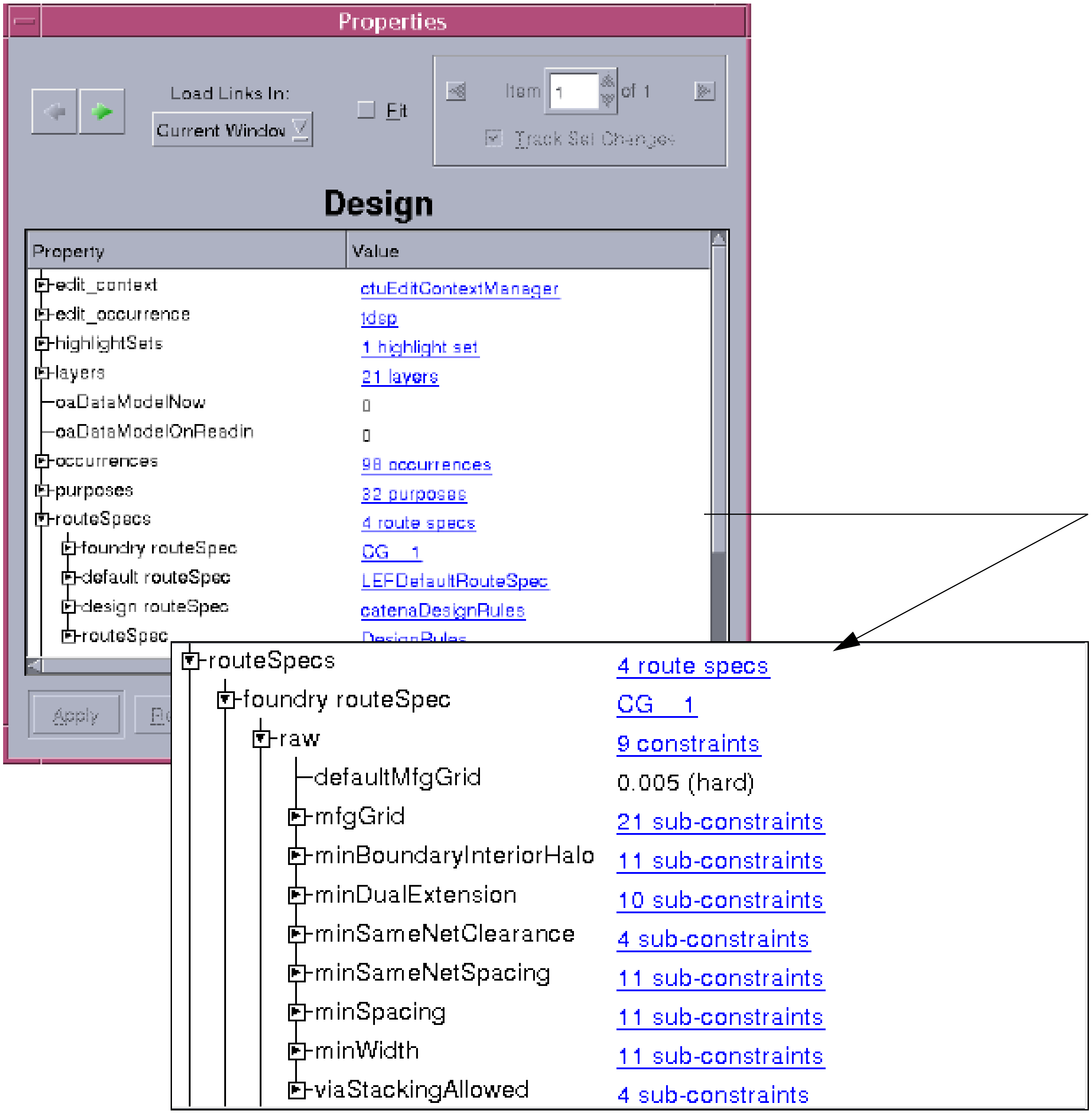
link_index	Property	Value
11.1	⊟ InstTerm	InstTerm Term A Net ALU 32 INST/i 9182/i 9187/n 356
11.1.13	bounds	xlo: 571.08, ylo: 567.07, xhi: 572.05, yhi: 567.47
11.1.12	⊟ connections	1 connection
11.1.12.1	⊞ Route	Route
11.1.11	index	0
11.1.10	⊞ instance	ALU 32 INST/i 9182/i 9187/i 189
11.1.9	isDriver	false
11.1.8	name	A
11.1.7	⊞ net	ALU 32 INST/i 9182/i 9187/n 356
11.1.6	⊞ occurrence	tdsp
11.1.5	⊞ parent	Net ALU 32 INST/i 9182/i 9187/n 356
11.1.4	pin_access	(none)
11.1.3	⊞ term	A
11.1.2	termType	InputTermType
11.1.1	toplevel	true

To change the orientation for the instance to R0 when the inspector id is 1,

```
inspect -set_prop -property_name orientation -new_value R0 -link_index 6 -id 1
```

Inspecting Routespecs

The following figure shows properties for a design, and expands the `routeSpecs` property.



Design

Property	Value
edit_context	ctuEditContextManager
edit_occurrence	1dsp
highlightSets	1 highlight set
layers	21 layers
oaDataModelNow	0
oaDataModelOnReadin	0
occurrences	98 occurrences
purposes	32 purposes
routeSpecs	4 route specs
foundry routeSpec	CG 1
default routeSpec	LEFDefaultRouteSpec
design routeSpec	catenaDesignRules
routeSpec	DesignRules

routeSpecs	4 route specs
foundry routeSpec	CG 1
raw	9 constraints
defaultMfgGrid	0.005 (hard)
mfgGrid	21 sub-constraints
minBoundaryInteriorHalo	11 sub-constraints
minDualExtension	10 sub-constraints
minSameNetClearance	4 sub-constraints
minSameNetSpacing	11 sub-constraints
minSpacing	11 sub-constraints
minWidth	11 sub-constraints
viaStackingAllowed	4 sub-constraints

To view properties for the Foundry mfgGrid route spec for this design, deselect all objects (deselect_set), then use the following command:

```
inspect -object \
[inspect_getprop -prop_name "routeSpecs.foundry routeSpec.raw.mfgGrid"]
```

Related Information

Menu Commands	*Edit—Properties*
Tcl Commands	inspect_getprop

inspect_config

```
inspect_config
     -type s_objectType
     [ -add {{[s_propName ][s_label:s_propPath]}…}]
     [ -none ]
     [ -remove {s_propName…} ]
     [ -reset ]
     [ -dump ]
```

Excludes properties from and/or adds subproperties or properties to the Properties dialog for the given object type.

Important

> This command only affects inspections of the given object type after the command is issued. Existing dialogs are not affected.

Arguments

`-add {{[s_propName ][s_label:s_propPath]}…}`

Adds properties and/or subproperties to the Properties dialog for the given object type.

To include a subproperty at the top level of display for the given object type, specify the name you want to assign to the subproperty, followed by a colon (:), then the path to the subproperty using the property names in hierarchical order from top to bottom, separated by the period (.) character. For example, `"tCap:parasitics.totalCap"`

`-dump`	Outputs a summary of the inspector configuration changes that have been made in the session for the given object type.
`-none`	Excludes all properties for the given object type.
`-remove {s_propName…}`	Excludes the given properties for the given object type.
`-reset`	Restores the default display of properties for the given object type.
`-type s_objectType`	Specifies the object type for the command.

Example

The following example excludes the top-level property from the display for Rectangle objects.

```
inspect_config -type Rectangle -remove "toplevel"
```

The following example excludes the top-level and priority properties from the display for Net objects.

```
inspect_config -type Net -remove {toplevel priority}
```

The following example shows only the layer, net and purpose properties for Rectangle objects.

```
inspect_config -type Rectangle -none -add {layer net purpose}
```

The following example restores the display to the default properties for Rectangle objects.

```
inspect_config -type Rectangle -reset
```

Displaying Subproperties at the Top Level

The following example causes the Metal6 minimum width stored built-in constraint value for the taper route spec of nets to be displayed at the top level of the Properties dialog with the label, `myM6Width`.

```
inspect_config -type Net \
-add "myM6Width:storedMultiSpecs.default.specs.taper.raw.minWidth.Metal6"
```

With hierarchy expanded in the Properties dialog, the data for this example might appear as follows in the display for a net:

```
-storedMultiSpecs             default (d:LEFDefaultRouteSpec), design, foundry
   -default                   MultiSpec 3 (d:LEFDefaultRouteSpec)
      -specs                  2 route specs
         -taper               LEFDefaultRouteSpec
            -raw              18 constraints
               -minWidth      6 sub-constraints
                  Metal6      0.3(hard)
                  Metal5      0.3(hard)
                  Metal4      0.3(hard)
                  Metal3      0.3(hard)
                  Metal2      0.3(hard)
                  Metal1      0.3(hard)
```

After the command is issued, the following appears at the top level for the net:

```
myM6Width                  0.3(hard)
```

For the same example, you could display the minimum width constraint for all metals using the following:

```
inspect_config -type Net \
-add "myWidths:storedMultiSpecs.default.specs.taper.raw.minWidth"
```

After the command is issued, the following appears at the top level for the net when myWidths is expanded:

```
-myWidths                     6 sub-constraints
                  Metal6      0.3(hard)
                  Metal5      0.3(hard)
                  Metal4      0.3(hard)
                  Metal3      0.3(hard)
                  Metal2      0.3(hard)
                  Metal1      0.3(hard)
```

inspect_getprop

```
inspect_getprop
     {-prop_name s_propName | -prop_no i_index | -name_of_prop_no i_index}
     [ -set d_setObj | -object d_ctuObj ]
     [ -item i_index ]
     [ -list_values | -count | -test_only ]
```

Returns the value of an object property. Alternatively, the name for a property can be returned. For compound properties, a list of values or the count can be returned instead of the summary value. The object may be given directly or as an item in a set. If no object or set is given, one of the following is used, in this order:

- The selected set, if not empty, and `-item` must be given if more than one item is in the set
- The entire design

Arguments

`-count` Specifies that the number of elements for the property be returned instead of the property value. Applies only to compound properties.

`-item` *`i_index`* Specifies the index of the object to inspect in the set. If there is only one object in the set, this argument is not needed. For a set of two objects, the first object is item 1, and the second is item 2. When the set includes more than one item, this value is required.

`-list_values` Specifies that the list of values be returned instead of the summary value. Applies only to compound or link properties, such as `elements`.

`-object` *`d_ctuObj`* Specifies the object to get the property for.

`-name_of_prop_no` *`i_index`*

Specifies that the name of the property given by the index be returned instead of the property value.

`-prop_name` *`s_propName`* Specifies the name of the property to return the value for. This argument will accept a dot-separated path to specify a chain of inspector links.

`-prop_no` *`i_index`* Specifies the index of the property to return the value for.

`-set` *`d_setObj`* Specifies the set to inspect.

`-test_only` Specifies that the return value for this command indicate whether the given property exists for the object (1: yes, 0:no).

Value Returned

One of the following:

`d_ctuObj` Is the value of the property.

`s_propValue` Is the value of the property.

`s_propName` (`-name_of_prop_no`) Is the name of the property.

`i_count` (`-count`) Is the number of items for the property.

`s_propElement…` (`-list_values`) Is the list of property elements.

`0 | 1` (`-test_only`) Indicates whether the property exists for the object.

Example

The following example gets the `type` property value for the first item in the selected set.

```
inspect_getprop -prop_name type -item 1 -set [get_selection_set]
```

The following example gets the `name` property value for the object given by the `net` variable.

```
inspect_getprop -prop_name name -object $net
```

Related Information

Documentation	
Menu Commands	*Edit—Properties*
Tcl Commands	inspect inspect_prop inspect_setprop

inspect_prop

```
inspect_prop
     {s_propName | list}
     [d_setObj | {d_ctuObj…} ]
```

Returns the specified property for objects in a set or objects from a list of object references. If no objects are given, the objects in the selected set are inspected.

The shorthand version of this command, `ip`, can also be used.

If the property name is a dot-separated list of names, the name is treated specially, recursively processing for each element of the chain of names. The following command returns the great-grandfather for an object:

```
    ip parent [ip parent [ip parent]]
```

This command can also be given as

```
    ip parent.parent.parent
```

Arguments

{d_ctuObj…} Specifies a list of one or more object identifiers.

d_setObj Specifies the identifier for a set of objects.

s_propName Specifies the name of the property.

`list` Returns a list of all of the property names that exist on any of the referenced objects.

Value Returned

propertyValue Is the property value. If a list of object identifiers is input, then a list of property values is returned.

Example

The following examples illustrate how the `inspect_prop` (`ip`) command is used.

Usage	Description
`ip name $net`	(`$net` contains the net object identifier) Returns `NET123`
`ip name [list $net1 $net2]`	(`$net1` and `$net2` contain object identifiers for the respective nets) Returns `[list NET123 NET456]`
`ip name`	Returns the names of the selected objects
`ip name [get_selection_set]`	Returns the name of the selected set ("`Selection Set`")
`ip type [ip parent.parent]`	For a selected route segment, returns the grandparent type "`ctuNet`" (parent is "`ctuRoute`")

For additional examples, refer to:

- Example—Finding Items in a Set
- Example—Create a Set Containing Instances Attached to Nets in a Set

Related Information

Tcl Command	hp make_list

inspect_setprop

```
inspect_setprop
     -prop_name s_propName
     -value s_propValue
     [ -set d_setObj | -object d_ctuObj ]
     [ -item i_index ]
```

Sets the value of the named existing property for an object in the given set or the selected set, or a given object.

Arguments

`-item` *`i_index`*	Specifies the index of the object within the set. If there is only one object in the set, this argument is not needed. For a set of two objects, the first object is item 1, and the second is item 2.
`-object` *`d_ctuObj`*	Specifies the object to set.
`-prop_name` *`s_propName`*	Specifies the name of the property to set the value for.
`-set` *`d_setObj`*	Specifies the set to use. If this argument is not given, the selected set is used.
`-value` *`s_propValue`*	Specifies the new value for the property.

Value Returned

`0`	The property is set.
`-1`	The property was not set due to a problem with the command such as a syntax error.

Example

The following example sets the orientation of a single selected instance.

```
inspect_setprop -prop_name orientation -value R90
```

Related Information

Menu Commands	*Edit—Properties*
Tcl Commands	inspect_getprop

make_list

```
make_list
     {d_setObj | d_ctuObjRef}
```

Constructs a Tcl list of object identifiers from the given set of objects or from the reference to a list of objects. If no argument is given, the Tcl list for the objects in the selected set is returned.

This command is useful for inspecting properties with the inspect_prop command.

The shorthand version of this command, `ml`, can also be used.

Arguments

d_ctuObjRef	Is the object reference for a list of objects, such as that returned by "`inspect_prop elements`" for a net.
d_setObj	Is the identifier for a set of objects, such as the set identifier returned by a `find_*` command.

Value Returned

d_ctuObj…	Is the Tcl list of object identifiers.

Example

The following table includes usage examples for `make_list`.

Usage	Description
`ml [ip elements $net]`	Returns a list of object ids, one for each element in the net
`ml [get_selection_set]`	Returns a list of object ids, one for each element in the set
`ml`	Returns the same as above (defaults to elements in the selection set)

Example—Finding Items in a Set

This procedure outputs the `type` property for all elements in the given set.

```
proc findItemsInSet {inputSet} {
  set instSet [create_set]
  foreach inputObject [ml $inputSet] {
    puts [ip type $inputObject]
  }
  puts "[set_count -set $inputSet] items in set."
}
```

The following example uses the `findItemsInSet` procedure and shows the output.

```
findItemsInSet [get_selection_set]
```

```
ctuRouteSegment
ctuRouteVia
ctuNet
ctuRoute
4 items in set.
```

Example—Create a Set Containing Instances Attached to Nets in a Set

This procedure returns a set of instances that are attached to nets in the given set.

```
proc findInstancesConnectedToNets {inputSet} {
  set instSet [create_set]
  foreach inputObject [ml $inputSet] {
    set inputObjectType [ip type $inputObject]
      if {[string compare $inputObjectType "ctuNet"] == 0} {
        foreach element [ml [ip elements $inputObject]] {
          set elementType [ip type $element]
          if {[string compare $elementType "ctuInstTerm"] == 0} {
            set elementInst [ip instance $element]
            add_object_to_set -object $elementInst -set $instSet
        }
      }
    } else { puts "$inputObjectType: this is not a net" }
  }
  puts "[set_count -set $instSet] instances are attached to selected nets."
  return $instSet
}
```

The following example uses the `findInstancesConnectedToNets` procedure and adds the instances that are found to the current highlight set.

```
replace_set -set1 [findInstancesConnectedToNets [get_selection_set]] -set2
[get_current_highlight]
```

Related Information

Tcl Command	hp inspect_prop

merge_poly

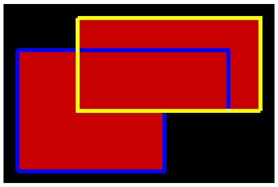
```
merge_poly
     [ -set d_setObj ]
```

Merges polygon shapes in the given set or the selected set at the top level. Only rectangles, octagons and polygons can be merged. All other shapes in the set are ignored. Shapes which overlap on the same layer purpose pair, and which belong to the same parent (for example, pin or top-level occurrence) are combined into a single geometry. If the combined geometry is rectangular, the original shapes are replaced with a rectangle shape under the same parent. If not, the original shapes are replaced with a polygon under the same parent. All-angle objects are merged to 45-degree edges when the final polygon is created.

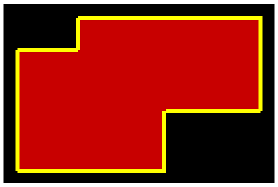
Arguments

`-set` *`d_setObj`*	Specifies the set of shapes to merge. If this argument is not included, the polygons, rectangles and octagons in the selected set are merged.

Example



Two overlapping shapes of the same layer purpose are shown, one outlined in yellow, the other in blue.



After the shapes are added to the selected set, the shapes are merged using `merge_poly`.

move

```
move
     -origin {f_x f_y}
     {-region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj}
     [ -dx f_userunit ]
     [ -dy f_userunit ]
     [ -extend_pick ]
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
```

Moves or slides instances, rectangles, vias, route segments, entire routes and entire nets. When you move objects, guides are created to indicate the disconnect between the moved objects and the existing nets. When you slide objects, connectivity is preserved by extending or adding segments.

This command is typically performed interactively in the workspace by choosing *Edit—Move* or *Edit—Slide.* The environment variable, `move.slide`, determines the operation performed.

To move objects,	`setvar move.slide false`
To slide objects,	`setvar move.slide true`

Arguments

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`	
	Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) coordinates for the bounding box target area. Picks active objects in the target area for the move.
`-dx` *`f_userunit`*	Specifies the move delta in the X direction.
`-dy` *`f_userunit`*	Specifies the move delta in the Y direction.
`-extend_pick`	Automatically picks active routes connected to picked instances and all active vias connected to picked segments.
`-orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90}`	
	Specifies the orientation to rotate the pick set, using a starting position of R0 as the current position. For a description of orientation values, refer to “Orientation Key” on page 208 .
`-origin {`*`f_x f_y`*`}`	Specifies the origin coordinates for the picked items. This point is used as the reference for rotation and move deltas.
`-set` *`d_setObj`*	Picks the objects in the specified set. If this argument is specified but no set is given, the selected set of objects is picked.

Value Returned

`0`	The move/slide was successful.
`-1`	No objects were moved.

Orientation Key

The following table describes each of the orientation values:

Value	Definition
`R0`	No rotation

Value	Definition
`R90`	Rotate counter-clockwise 90 degrees
`R180`	Rotate counter-clockwise 180 degrees
`R270`	Rotate counter-clockwise 270 degrees
`MY`	Mirror through Y axis
`MY90`	Mirror through Y axis and rotate counter-clockwise 90 degrees
`MX`	Mirror through X axis
`MX90`	Mirror through X axis and rotate counter-clockwise 90 degrees

Example

The following command moves the active objects in the selected set.

```
move -origin {2028.65 1391.65} -set -dx 2.6 -dy 1.5
```

The following command moves the active objects in the given area.

```
move -origin {2033.24 1398.75} -area {2028.14 1383.97 2033.49 1398} -dx -0.7 -dy
-1.9
```

Related Information

Tcl Commands	redo split_oa_terminals
Menu Commands	*Edit—Move*, *Edit—Slide*

redo

```
redo
```

Reapplies a routing or editing operation that was reversed by an `undo` command.You can also use a series of `redo` commands to reapply a series of operations that were reversed using `undo`.

The routing and editing operations that can be reversed with `undo` and reapplied with `redo` include the following: interactive wire editing (including adding wires and vias), interactive creation of instances and rectangles, `delete`, `cut`, and `move`.

Arguments

None

Value Returned

`0`	An operation was reapplied.
`-1`	There is no operation to reapply.

Related Information

Tcl Commands	split_oa_terminals
Menu Commands	*Edit—Redo*

remaster_instance

```
remaster_instance
     -view {s_fromView s_toView}
     {-all | -set d_setObj}
     [ -lib {s_fromLib s_toLib} ]
     [ -cell {s_fromCell s_toCell} ]
     [ -from_same_library [ true | false ] ]
     [ -verbose ]
     [ -check_terms {ignore | notify | enforce} ]
```

Binds instances matching a given criteria in the entire design or in the given set to a new master. By default, the new master must have the same number of terminals as the old master.

Arguments

`-all` Process all instances in the design that meet the lib/cell/view criteria.

`-cell {`*`s_fromCell s_toCell`*`}`

[Optional] Limits processing to instances whose master cell names match the 'from' cell. Matching instances will be changed to the corresponding 'to' master. Wildcards are permitted.

`-check_terms {ignore | notify | enforce}`

Specifies the level of checking to do on the new master from the following choices:

`ignore` Remasters without checking.

`notify` Issues a warning if the terminals for the old and new masters do not match. The warning message lists the "From" and the "To" lib/cell/view and the terminals that do not match. The instance is remastered even if the terminals do not match.

`enforce` Issues a warning if the terminals for the old and new masters do not match. The warning message lists the "From" and the "To" lib/cell/view and the terminals that do not match. The instance is not remastered if the terminals do not match.

`-from_same_library [ true | false ]`

When this argument is `true` and the `-lib` argument is omitted, then only remaster instances if the targeted view is from the same library.

Default: `true`

`-lib {`*`s_fromLib s_toLib`*`}`

[Optional] Limits processing to instances in the 'from' library. Matching instances will be changed to the corresponding 'to' master. Wildcards are permitted. Setting this argument overrides the `-from_same_library` setting. To search all libraries, use `-lib {* *}`.

`-set` *`d_setObj`* — Processes only the instances in the given set that meet the lib/cell/view criteria.

`-verbose` — Issues a message as each instance is remastered.

`-view` **{***`s_fromView s_toView`***}**

All instances matching the 'from' view are changed to the corresponding 'to' view.

Value Returned

`i_count` — Is the number of instances remastered.

Example

The following example remasters all instances in the selected set with `myView` view to the `newView` view.

```
remaster_instance -set [get_selection_set] -view {myView newView}
```

Related Information

Tcl Commands — create_instance

remove_highlight

```
remove_highlight
     {-all | -set d_setObj | -name s_hlName}
```

Deletes highlight sets.

Arguments

`-all`	Deletes all highlight sets. A new empty highlight set, `HL1`, is created.
`-name` *`s_hlName`*	Deletes the named highlight set. If this is the only highlight set for the cellview, a new empty set, `HL1`, is created.
`-set` *`d_setObj`*	Deletes the specified highlight set. If this is the only highlight set for the cellview, a new empty set, `HL1`, is created.

Value Returned

`1`	The highlights are successfully deleted.

Example

The following command removes all the highlights sets in the active cellview.

```
remove_highlight -all
```

The following command removes the highlight set named `hset1`. This is equivalent to choosing *Delete Highlight Set* from the highlight set pop-up menu in the Layer Object Display Panel.

```
remove_highlight -name "hset1"
```

Related Information

Tcl Commands	add_highlight get_highlight get_num_highlights

remove_object_from_net

```
remove_object_from_net
     -net_name s_netName
     {-object d_ctuObj | -set d_setObj}
```

Removes objects from a net. You can remove a single object (`-object`) or a set of objects (`-set`).

Arguments

`-net_name` *`s_netName`*	Specifies the name of the net.
`-object` *`d_ctuObj`*	Specifies the identifier for the object to remove.
`-set` *`d_setObj`*	Specifies the identifier for a set of objects to remove.

Example

The following example removes objects in the selected set from the `mynet` net.

```
remove_object_from_net -set [get_selection_set] -net_name mynet
```

Related Information

Tcl Command	add_object_to_net

remove_object_from_set

```
remove_object_from_set
     -object d_ctuObj
     -set d_setObj
```

Removes an object from a set.

Arguments

`-object` *`d_ctuObj`*	Specifies the object to remove.
`-set` *`d_setObj`*	Specifies a set.

Example

The following example removes an object, given by the `myobj` variable, from the selected set.

```
remove_object_from_set -object $myobj -set [get_selection_set]
```

Related Information

Tcl Command	add_object_to_set

report_set

```
report_set
     -set d_setObj
     -format {text|CSV}
     [ -file s_fileName ]
     [ -append ]
```

Creates a report for a set of objects. The report contains information about objects (instances, nets and shapes) in the set. You can choose the output file name and must specify the format for the report.

Arguments

`-append`	Appends the current report data to the file if the file already exists. If not specified and the file exists, the file is overwritten.
`-file` *`s_fileName`*	Specifies the name of the file to write the results to. If this argument is not specified, the results are saved to a temporary file. If this argument is not specified, the results are output to a file named `report_`*`yyyymmdd`*`_`*`hhnnss`*`.`*`ext`* where *`yyyy`* (year), *`mm`* (month), *`dd`* (day), *`hh`* (hour), *`nn`* (minute), and *`ss`* (second) represent the date and time the file was created, and *`ext`* is `txt` if you selected text format, or `csv` for comma-separated values format. Example: `report_20021218_113355.csv`
`-format {text|CSV}`	Specifies the output format type as text or comma-separated values. If you choose `text`, data is presented in columnar format.
`-set` *`d_setObj`*	Specifies the set to report on.

Example

The following commands find all nets whose names begin with `RESULTS_CONV_INST_n` and stores the set of objects in a Tcl variable `net`. The second command creates a text report of all objects in the set defined by the Tcl variable `net` and outputs the results to file `report.txt`.

```
set net [find_net -name ^RESULTS_CONV_INST_n]
report_set -format text -set $net -file report.txt
```

The following command creates a comma-separated values file of all objects in set `inst_high` and outputs the results to a file with a default name that includes the date and time the file was created.

```
report_set -set $inst_high -format CSV
```

Related Information

Tcl Commands	find_instance find_instance_of find_net

scheme

```
scheme
     {-create s_name [ -package s_pkgName ] [ -overwrite [ true | false ] ]
     | -copy s_name -from s_name [ -package s_pkgName ] [ -overwrite [ true |
     false ] ]
     | -rename s_name -from s_name [ -overwrite [ true | false ] ]
     | -delete s_name [ -package s_pkgName ]
     | -activate s_name [ -package s_pkgName ]
     | -comment s_name -value s_text
     | -compare s_name -against s_name [ -package s_pkgName ]}
```

Performs operations for managing schemes.

Schemes provide a method for quickly saving and restoring environment variables used for some Space-based Router and Chip Optimizer functions. Each group of environment variables is defined and manipulated by a *scheme package*. The *default* scheme is initialized by Space-based Router and Chip Optimizer, determines the available package types and the environment variables included in each package, and is used when creating new schemes.

You can create multiple schemes using the `-create` and `-copy` arguments. In addition, one scheme for each package type is the *current* or *active* scheme. This is the scheme used by the function associated with the package for its next run.

Arguments

`-activate` *`s_name`*	Specifies the name of the scheme (if -`package` is not given) or the scheme package (if -`package` is given) to make active.
`-against` *`s_name`*	Specifies the name of the scheme to compare against the scheme given by the `-compare` argument value.
`-comment` *`s_name`*	Specifies the name of the scheme to add the comment to.
`-compare` *`s_name`*	Specifies the name of the scheme to compare against the scheme given by the `-against` argument value.
`-copy` *`s_name`*	Specifies the name of the scheme to create by copying the scheme given by the -`from` argument. By default, all scheme packages are copied. If the `-package` argument is given, only the given scheme package is copied.
`-create` *`s_name`*	Specifies the name of the scheme to create by copying the `default` scheme settings. By default, all `default` scheme packages are copied. If the `-package` argument is given, only the given scheme package is copied.
`-delete` *`s_name`*	Specifies the name of the scheme to delete. **Note:** The `default` scheme cannot be deleted.
`-from` *`s_name`*	Specifies the name of the scheme to copy from (if `-copy` is given) or to rename (if `-rename` is given).
`-overwrite [ true | false ]`	
	For copy, create and rename functions, if `true`, forces the target scheme or scheme package to be overwritten, if it already exists. If not set `true` and the target scheme or scheme package already exists, a warning message is issued and no action is taken.
`-package` *`s_pkgName`*	Limits the function to the given package.
`-rename` *`s_name`*	Renames the scheme given by the -`from` argument to the given name.
`-value` *`s_text`*	Specifies the text to add to the scheme given by the `-comment` argument.

Example

The following commands are equivalent. Each command creates a new scheme, `myScheme`, from the `default` scheme.

```
scheme -create myScheme
scheme -copy myScheme -from default
```

The following command activates the `power_cell_rows` package of the `myScheme` scheme. Assuming that the `default` scheme is active, all `default` scheme packages will be active except for the `power_cell_rows` package after this command is issued.

```
scheme -activate myScheme -package power_cell_rows
```

select_all

```
select_all
```

Adds all active layers and objects to the selected set. The selected set is outlined by a yellow line in the workspace. You make layers and objects active in the Layer Object Display Panel.

Arguments

None

Related Information

Menu Commands	*Edit—Select—Select All*

select_nets_on_routes

```
select_nets_on_routes
     -in d_setObj
     -out d_setObj
```

Finds nets that the routes in the given set (`-in`) are attached to, and puts the nets into another set (`-out`).

Arguments

`-in` *`d_setObj`*	Specifies the set of routes.
`-out` *`d_setObj`*	Specifies the set in which to put the nets.

Example

The following example takes routes in the selected set and puts the nets that the routes are attached to into the `HL1` highlight set, then refreshes the artwork.

```
select_nets_on_routes -in [get_selection_set] -out [get_highlight -name HL1]
refresh
```

Related Information

Tcl Commands	select_routes_on_nets

select_routes_on_nets

```
select_routes_on_nets
     -in d_setObj
     -out d_setObj
```

Finds routes attached to the nets in the given set (`-in`), and puts the routes into another set (`-out`).

Arguments

`-in` *`d_setObj`*	Specifies the set of nets.
`-out` *`d_setObj`*	Specifies the set in which to put the routes.

Example

The following example takes nets in the selected set and puts the routes that are attached to the nets into the HL1 highlight set.

```
select_routes_on_nets -in [get_selection_set] -out [get_highlight -name HL1]
refresh
```

Related Information

Tcl Commands	select_nets_on_routes

set_count

```
set_count
     -set d_setObj
```

Returns the number of objects in the specified set.

Arguments

`-set` *`d_setObj`*	Specifies the set.

Value Returned

`i_count`	Specifies the number of objects in the set.

Example

The following example sets the `aset_count` variable with the number of objects in the selection set.

```
set aset_count [set_count -set [get_selection_set]]
```

The following example sets the `nandinst_count` variable with the number of instances with names that begin with `NAND`.

```
set nandinst_count [set_count -set [find_instance -set $NAND -name "NAND*"]]
```

Related Information

Tcl Commands	report_set

set_current_highlight

```
set_current_highlight
     -set_number i_index | -name s_hlName
```

Specifies the highlight set to assign as the current highlight set.

Arguments

`-name` *`s_hlname`*	Specifies the name of the highlight set to assign as the current highlight set.
`-set_number` *`i_index`*	Specifies the highlight set number of the set to assign as the current highlight set. **Note:** You can use get_num_highlights to get the total number of highlight sets currently available. Sets are numbered `0` to `get_num_highlights-1.`

Value Returned

`d_setObj`	Specifies the set identifier for the new current highlight set.
`0`	No sets are found that match the argument criteria, or there was a command syntax error.

Example

The following example designates the highlight set named `HL2` to be the current highlight set.

```
set_current_highlight -name "HL2"
```

Related Information

Tcl Command	get_current_highlight get_num_highlights

split_oa_terminals

```
split_oa_terminals
     -lib s_libName
     [-libDefFile <path> ]
     -cell s_cellName
     -view s_viewName
     [ -saveLib s_libName ]
     [ -saveCell s_libName ]
     [ -saveView s_libName ]
     -net s_netName
     [ -distance <number> ]
```

Iterates instance terminals of a given cell (top-cell) on a given net and examines corresponding master terminals. In case master terminal has more than one pin then the command can create additional master terminal for the pin based on distances between pins bounding boxes. Pin's bounding box is calculated as a union of bounding boxes of its figures. If distance between this pin box center and all other pins bounding boxes centers is greater than the specified input parameter, a new terminal is created. The pin is added to the new terminal and `MustJoin` relation is set between original terminal and the new one.

Note: The `dummy` net for a new terminal is created as `MustJoin` and is not allowed for two terms on the same net. Names are generated for a new net and terminal, which are unique for a block.

If the optional parameters (`-saveLib`,`-saveView`, `-saveCell`)] are not specified, the modified cells are saved in the same library with the same name and view. The top-level cell is not re-mastered. However, if `saveLib` is specified, modified cells are saved in a specified library and the top-cell is re-mastered.

Arguments

-lib *s_libName*

Specifies the input library. The library must exist in the library definitions file in the current working directory or must exist in the library definitions file specified with the -libDefFile option.

-libDefFile *<path>*

Specifies the lib.defs file to load. The path to the lib.defs file can be a full or relative path.

-cell *s_cellName*

Specifies an input cell. If the view option is not specified, the default view name layout is used.

-view *s_viewName*

Specifies the view name of the cells to examine. It is incorrect to specify the view name without specifying the cell name.

Default: layout.

-saveLib *s_saveLibName*

Specifies a name for the output library. The library must exist in the library definitions file in the current working directory or must exist in the library definitions file specified with the -libDefFile option.

-saveCell *s_saveCellName*

Specifies a name for an output cell. If the saveCell option is not specified, the original view name is used.

Note: This argument specifies only a name for a top cell. Other masters are saved with original cell names.

-saveView *s_saveViewName*

Specifies view name for all modified cells (including top-cell). If the saveView option is not specified, the original view name is used.

-net *s_netName*

Specifies an input net. InstTerms connected to this net are examined by the command.

-distance *<number>*

Specifies a distance. Pin farther away from other pins than the distance are added on the newly created terminal. Default: 0. This means if the default value is not changed, then new terminals are created for every pin.

Value Returned

`0`	When no error is encountered.

undo

```
undo
```

Reverses a routing or edit operation. You can also reverse a series of those operations using a series of `undo` commands.

You can immediately redo an operation that was reversed by an `undo`, or reverse a series of `undo` commands using multiple `redo` commands.

The routing and editing operations that can be reversed with `undo` and reapplied with `redo` include the following: interactive wire editing (including adding wires and vias), interactive creation of instances and rectangles, `delete`, `cut`, and `move`.

Arguments

None

Value Returned

`0`	An operation was reversed.
`-1`	There is no operation to reverse.

Related Information

Tcl Commands	redo
Menu Commands	*Edit—Undo*

4

View Commands

This chapter describes the View commands.

The commands are presented in alphabetical order.

create_view_context

```
create_view_context
     -region {f_xlo f_ylo f_xhi f_yhi}
     -name s_vcName
     -layer_info {{s_lpp s_visBool s_actBool i_opacity} …}
     [ -active_layers_only [ true | false ] ]
     [ -bindkey s_key ]
     [ -comment s_text ]
     [ -creation_date s_date ]
     [ -end_level i_level ]
     [ -entry_layer s_lpp ]
     [ -instance_name_display_mode {Instance | Master} ]
     [ -opacity i_opacity ]
     [ -owner s_name ]
     [ -relative_scaling [ true | false ] ]
     [ -show_all_layers_in_lods [ true | false ] ]
     [ -show_full_instance_path [ true | false ] ]
     [ -show_instance_orientation [ true | false ] ]
     [ -show_instance_origin [ true | false ] ]
     [ -start_level i_level ]
```

Creates a view context. This Tcl command is the equivalent to creating a view context in the Space-based Router and Chip Optimizer GUI by specifying a view context name in the View Contexts Browser, then clicking *Add* in the Browser.

Arguments

`-active_layers_only [ true | false ]`

Chooses whether the Global Control for setting opacity applies to active layers only.

`-bindkey` *`s_key`*

Specifies a bindkey for the view. When the bindkey is pressed, the coordinates and display options saved in the view context are restored. This allows you to quickly zoom to a view with all the original display options, except color, restored.

`-comment` *`s_text`*

Specifies comments for the view context.

`-creation_date` *`s_date`*

Specifies the creation date.

`-end_level` *`i_level`*

Specifies the last hierarchy level to display.

`-entry_layer` *`s_lpp`*

Specifies the entry layer purpose name.

`-instance_name_display_mode`

Chooses whether to display the name of the instance or the master cell. Valid values are `Instance` and `Master`.

`-layer_info {{`*`s_lpp s_visBool s_actBool i_opacity`*`}…}`

Specifies the list of layer purposes and objects, with their visibility setting (`true|false)`, active state (`true|false`) and opacity (integer from 0 through 255).

`-name` *`s_vcName`* Specifies the name for the view context.

`-opacity` *`i_opacity`*

Specifies the Global Control opacity setting. Valid values are 0 (transparent) through 255 (most opaque).

`-owner` *`s_name`* Specifies the name of the person who created the view context.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the rectangular bounding box for the view. The area is specified as a list of numbers: lower left x-coordinate, lower left y-coordinate, upper right x-coordinate, upper right y-coordinate.

`-relative_scaling [ true | false ]`

Chooses whether to maintain relative scaling when changing the Global Control opacity setting.

`-show_all_layers_in_lods [ true | false ]`

Chooses whether to show all layers in the Layer Object Display Panel, including layer purposes without shapes, or show only layer purposes that contain shapes.

`-show_full_instance_path [ true | false ]`

Chooses whether to display the full path name for instances.

`-show_instance_orientation [ true | false ]`

Chooses whether to designate the orientation when displaying instances. When used, a diagonal is drawn across a corner of the instance boundary.

`-show_instance_origin [ true | false ]`

Chooses whether to designate the origin when displaying instances. When used, a plus (+) sign on the instance boundary marks the origin.

`-start_level` *`i_level`*

Specifies the starting hierarchy level to display.

Example

The following example creates a view context named `allvis` that displays hierarchy levels 0 through 99. The opacities for layers and objects are given with the names of the visible layers and objects. The creation date and bounding box for the view are included.

```
create_view_context -active_layers_only false -region { -4870370 30 5007410 137010
} -bindkey  -comment  -creation_date "Fri Nov 14 17:05:47 2008" -end_level 99
-entry_layer metal3:wire:detail -instance_name_display_mode Instance -layer_info {
"instance:boundary true true 90""metal1:blockage true true 100" "metal1:via:pin
true true 200" "metal1:via:redundant true true 200" "metal1:via:global true true
200" "metal1:via:detail true true 200" "metal1:wire:pin true true 200"
"metal1:wire:redundant true true 200" "metal1:wire:global true true 200"
"metal1:wire:detail true true 200" } -name tiny -opacity 90 -relative_scaling true
-show_all_layers_in_lods false -show_full_instance_path false
-show_instance_orientation false -show_instance_origin false -start_level 0
```

Related Information

Tcl Commands	delete_view_context show_view_context

delete_view_context

```
delete_view_context
     -name s_vcName
```

Deletes the specified view context. This Tcl command is equivalent to deleting the view context in the Space-based Router and Chip Optimizer GUI by right-clicking on the view context name in the View Contexts Browser and choosing *Delete* in the pop-up window.

Arguments

`-name` *`s_vcName`*	Specifies the name of the view context to delete.

Example

The following example deletes the view context named `met123`. The View Contexts Browser and View Contexts Quick Navigator are automatically refreshed.

```
delete_view_context -name met123
```

Related Information

Tcl Commands	create_view_context

fit

```
fit
     {-all | -loupe | -next | -previous | -selected | -set d_setObj
     | -region {f_xlo f_ylo f_xhi f_yhi} [ -border]}
     [ -window_id i_windowID]
```

Fills the view window with the full design or a rectangular portion of it, and can be invoked at any time, including within another command.

The `fit` command handles interrupts, although it might leave a partially drawn window on the screen.

Arguments

`-all`	Requests a view of the entire design displayed in the artwork window.
`-border`	Causes `fit -region` to use settings for the environment variables, `gui.zoom_out_factor` and `gui.max_track_size_in_pixels`, when fitting the region in the window. This allows you to put context around the zoom area. If this argument is not used, the given region is fit to its maximum size in the artwork window.
`-loupe`	Fits the magnifying loupe area in the artwork window.
`-next`	Redisplays the next view on the stack.
`-previous`	Redisplays the previous view on the stack.
`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`	
	Specifies the rectangular bounding box for the fit. The area is specified as a list of numbers: lower left x-coordinate, lower left y-coordinate, upper right x-coordinate, upper right y-coordinate.
`-selected`	Fits the selected set in the artwork window.
`-set` *`d_setObj`*	Fits the specified set in the artwork window.
`-window_id` *`i_windowID`*	Specifies the identifier of the window to work in. If this option is not specified, the view in the active window is used.

Example

```
fit -all
```

Does a window fit of the entire design.

```
fit -previous
```

Goes back to the previous view.

Related Information

Tcl Commands	view_layer
Menu Command	*View—Fit*

get_drawing_order

```
get_drawing_order
     {-layers | -purposes | -highlights}
     [ -window_id i_windowID]
```

Returns the current drawing order for layers, purposes or highlights for the active window or the specified window. Only one item type (layers, purposes or highlight sets) may be requested at a time. Space-based Router and Chip Optimizer displays the requested drawing order in the Transcript area.

Arguments

`-highlights`	Requests the drawing order for highlight sets.
`-layers`	Requests the drawing order for layers.
`-purposes`	Requests the drawing order for layer purposes.
`-window_id` *`i_windowID`*	Specifies the identifier of the window to report on. If this argument is not specified, the active window is used.

Returned Value

`s_layerName`… | *`s_purposeName`*… | *`s_hlName`*…	Is the ordered list of layers, purposes or highlights represented as strings.

Example

The following example requests the drawing order for the layer purposes of the cellview in window 2 and shows the results returned by Space-based Router and Chip Optimizer.

```
get_drawing_order -purposes -cell_view_id [get_cell_view -window_id 2]
"fill" "label" "text" "drawing" "blockage" "via:pin" "via:redundant" "via:global"
"via:detail" "wire:pin" "wire:redundant" "wire:global" "wire:detail"
```

Related Information

Tcl Commands	set_drawing_order

get_layers

```
get_layers
     [ -with_shapes ]
     [ -visible true | false | dont_care ]
     [ -active true | false | dont_care ]
     [ -material [ [metal] [cut] [poly] [other] [all] ] ]
     [ -output_format layers | lpps | lpp_settings ]
     [ -routing [ true | false ] ]
     [ -colored [ true | false ] ]
     [ -window_id i_windowID ] | [ -tech_lib s_techName ]
```

Returns a list of layer names and objects in the given window or technology library that meet the criteria specified by the command arguments. If no arguments are given, the returned list includes all layers and, optionally, objects defined in the technology library for the active window. The visibility and active state for layers and objects are set in the Layer Object Display Panel of the GUI.

Arguments

`-active true | false | dont_care`

Specifies which layers and objects to include in the return list.

`dont_care` Includes all layers and objects.

`false` Includes only layers and objects that are not active.

`true` Includes only layers and objects that are active.

`-material [ [metal][cut][poly][other][all] ]`

Specifies the material types of the layers to list. The default is `all` materials.

`-output_format layers | lpps | lpp_settings`

Specifies the list output format.

`layers` Returns a list of layer names that meet the criteria. Objects are not included. This is the default.

`lpp_settings` Returns a list of layer purposes and objects that meet the criteria with their associated visibility, active state, and opacity.

`lpps` Returns a list of layer purposes and objects that meet the criteria.

`-routing [ true | false ]`

Returns the routing layer names.

`-tech_lib` *`s_techName`* Uses the given technology library. If this argument is given, the `-output_format`, `-with_shapes`, `-visible`, and `-active` arguments are ignored and the list of all layers and objects is output. The technology library must be defined in `lib.defs`.

`-visible true | false | dont_care`

Specifies which layers and objects to include in the return list.

`dont_care` Includes all layers and objects.

`false` Includes only layers and objects that are not visible.

`true` Includes only layers and objects that are visible.

`-colored`

When specified, returns the layer names that support coloring. Default is `true`.

`-window_id` *`i_windowID`* Uses the design in the given window. If this argument is not given, the active window is used.

`-with_shapes` Restricts layers to those that contain shapes. If this argument is not given, all defined layers that meet the criteria are listed.

Value Returned

{*`s_layerlpp`* [*`s_visibilityBool s_activeBool i_opacity`*]}…

	Is the list of layer/objects that meet the criteria represented as strings. If used, the -`output_format` argument determines the format of list items.

Example

The following command requests all layers that contain shapes that are active and visible in the active window.

```
get_layers -active true -visible true -with_shapes
```

The following command requests all defined layer purpose pairs in the active window.

```
get_layers -output_format lpps
```

The following command requests all defined layer purposes and objects in the `mytechLib` technology library.

```
get_layers -tech_lib mytechLib -output_format lpps
```

Related Information

Tcl Commands	set_active view_layer
Menu Command	*Window—Layer Object Panel*

get_view_contexts

```
get_view_contexts
```

Returns the list of saved view contexts that are shown in the View Contexts Browser.

Arguments

None

Value Returned

s_vcName …	Is the list of view context names represented as strings.

Example

The following is an example of information displayed in the Transcript Area in response to a `get_view_contexts` query.

```
"met1" "met2"
```

Related Information

Tcl Commands	create_view_context show_view_context

get_window_area

```
get_window_area
     [ -window_id i_windowID]
```

Requests the four bounding box coordinates (lower left x and y, and the upper right x and y) for the specified window or the current active window.

Arguments

`-window_id` *`i_windowID`*	Specifies the identifier of the window to work in. If this option is not specified, the view in the active window is used.

Value Returned

`f_xlo f_ylo f_xhi f_yhi`

Is the window area bounding coordinates, represented as strings.

Example

The following command requests the window area coordinates for window 1.

```
get_window_area -window_id 1
```

The following command requests the window area coordinates for the active window.

```
get_window_area
```

The following is an example of the returned data.

```
701.996 875.023 747.334 909.199
```

pan

```
pan
     {-up | -down | -left | -right | -ul | -ur | -dl | -dr}
     f_factor
     [ -window_id i_windowID]
```

Moves the view displayed on the screen in the direction specified, maintaining the same level of magnification. You can use `pan` at any time, including nested in another command.

When you specify a direction, the view window moves across the design in that direction, with part of the original artwork view still visible. If `pan` is interrupted, a partially drawn window might be left on the screen.

Arguments

`-dl`	Pans down and left.
`-down`	Pans down.
`-dr`	Pans down and right.
`-left`	Pans left.
`-right`	Pans right.
`-ul`	Pans up and left.
`-up`	Pans up.
`-ur`	Pans up and right.
`-window_id` *`i_windowID`*	Specifies the identifier of the window to pan in. If this option is not specified, the view in the active window is shifted.
`f_factor`	Specifies the pan factor. For example, a value of `0.5` pans the view in the desired direction, half the distance of the visible window. Default: `0.5`

Example

```
pan -left
```

Shifts the view left.

Related Information

Menu Command	*View—Pan—Up* *View—Pan—Down* *View—Pan—Left* *View—Pan—Right*

refresh

```
refresh
     [ -window_id i_windowID | -all]
     [ -pause ]
```

Redraws the entire artwork window, updating the screen to show recent changes, additions and modifications.

If `refresh` is interrupted, a partially drawn window might be left on the screen.

Arguments

`-all`	Refreshes all artwork windows.
`-pause`	Waits for the windows to be refreshed before continuing with the next command.
`-window_id` *`i_windowID`*	Specifies the identifier of the window to be refreshed.

Example

```
refresh -window_id 1
```

Redraws window 1.

Related Information

Menu Commands	*View—Refresh*

rename_layer

```
rename_layer
     -old_name s_layerName
     -new_name s_layerName
     [ -window_id i_windowID]
```

Renames a layer. The Layer Object Display Panel is automatically refreshed.

Arguments

`-new_name` *`s_layerName`*	Specifies the new name for the layer.
`-old_name` *`s_layerName`*	Specifies the layer to rename.
`-window_id` *`i_windowID`*	Renames layers in the given window.
	Default: Operates on layers in the active window.

Example

The following example renames layer `met2` to `metal2`.

```
rename_layer -old_name met2 -new_name metal2
```

rename_view_context

```
rename_view_context
     -old_name s_vcName
     -new_name s_vcName
```

Renames a view context. The View Contexts Browser is automatically refreshed.

Arguments

`-new_name` *`s_vcName`*	Specifies the new name for the view context.
`-old_name` *`s_vcName`*	Specifies the view context to rename.

Example

The following example renames the view context `met12` to `met1met2`.

```
rename_view_context -old_name "met12" -new_name "met1met2"
```

save_view

```
save_view
     [ -window_id i_windowID]
```

Saves the coordinates and magnification for the active cellview window or the given window to the view stack. You use the `fit` command with `-previous` and `-next` arguments to display saved views from the view stack.

Arguments

`-window_id` *`i_windowID`*	Saves the view in the given window. Default: Saves the view in the active window.

Related Information

Tcl Commands	fit
Menu Commands	*View—Display Next View* *View—Display Previous View*

set_active

```
set_active
     {[ -all_layers [ true | false ] ]
      [ -lpp {s_layerlpp …}]
      [ -object {s_objectName …}]}
     -active [ true | false ]
     [ -window_id i_windowID]
```

Sets the active state for layers and objects and sets the routing object granularity to control what can be selected and highlighted. The routing object granularity is displayed when you right-click in the artwork while in select or highlight mode and you choose *Routing Object Granularity*.

Arguments

`-active [ true | false ]`
Specifies whether to set the layer/object active (`true`) or inactive (`false`). Only active layers and objects can be highlighted and selected.

`-all_layers [ true | false ]`
Specifies whether all layer purposes that correspond to physical layers should be set.

`-lpp` {*`s_layerlpp`*…}
Specifies the list of layers or layer purposes and objects to set. If you specify a layer name, all lpps for the layer are set.

`-object` {*`s_objectName`*…}
Specifies the routing object granularity. Valid choices are `net`, `route` and `connected_shapes`. To enable a granularity, you must set one choice with `-active true`, and set the other two choices with `-active false`. If

Note: Active individual shapes are always accessible.

`-window_id` *`i_windowID`*
Specifies the identifier of the cellview window that contains the layers/objects to set.

Example

The following example sets `met5` layer purposes active.

```
set_active -lpp { met5 } -active true
```

The following examples set all defined layers active.

```
set_active -all_layers true -active true
```

```
set_active -lpp [get_layers] -active true
```

The following example sets the Routing Object Granularity to *Entire Net*.

```
set_active -object { "net" } -active true
set_active -object { "route" "connected_shapes" } -active false
```

Related Information

Tcl Commands	get_layers

set_drawing_order

```
set_drawing_order
     {-highlights {s_hlName …} | -layers {s_layerName …} | -purposes
     {s_purposeName …}}
     [ -window_id i_windowID]
```

Sets the order of layers, layer purposes and highlight sets. The cellview window and the Layer Object Display Panel are updated using the specified drawing order.

Note: If you do not specify all possible items in the list, the omitted items are moved to the beginning of the drawing order and are drawn first, and the named items are put at the end of the sequence and are drawn last. For listed items, the first item is drawn first and the last listed item is drawn last.

The drawing order for the cellview window is: instances, layer purposes (all purposes for the first layer are drawn, then the purposes for the second layer, and so on), term labels, guides, grids, highlights, selected set, annotations, annotation highlights, selected annotations.

Arguments

`-highlights` **{***s_hlName …***}**

Specifies the ordered list of highlight sets.

`-layers` **{***s_layerName …***}**

Specifies the ordered list of layers.

`-purposes` **{***s_purposeName …***}**

Specifies the ordered list of layer purposes.

`-window_id` *i_windowID* Specifies the identifier of the window to report on. If this argument is not specified, the active window is used.

Example

The following example causes `met3` purposes to be drawn first, `met2` next, and `met1` purposes to be drawn last.

```
set_drawing_order -layers {"met3" "met2" "met1"}
```

Related Information

Tcl Commands	get_drawing_order

set_layer_attributes

```
set_layer_attributes
     -lpp {s_layerlpp …}
     -color s_color
     [ -norepaint]
     [ -window_id i_windowID]
```

Sets the color for layer purposes and objects. Multiple objects may be specified but only one color.

Arguments

`-color` *`s_color`*

Specifies the color in hexadecimal, preceded by a pound sign (#), or as a string name of the color. A list of colors and their hexadecimal equivalents can be found on http://eies.njit.edu/~walsh/rgb.html.

`-lpp {`*`s_layerlpp…`*`}`

Specifies the list of layer purposes and objects to assign the color to. Layer purposes are represented as strings in the following format *`layer`*`:`*`layer purpose`*. For example, `"met3:Blockage"` Objects are similarly represented.

`-norepaint`

Overrides the default action to refresh the window after this command is processed. This option is useful when you are issuing commands from a Tcl script and want to delay refreshing the window until all commands have been issued.

`-window_id` *`i_windowID`*

Specifies the identifier of the window to set attributes for. If this argument is not specified, the active window is used.

Example

The following example assigns the color represented by `#ff8a5c` to all `met2` layer purposes, and the `met1` layer purposes `blockage`, `wire:pin` and `wire:detail`.

```
set_layer_attributes -lpp {"met2" "met1:wire:pin" "met1:wire:detail"}
    -color #ff8a5c
```

The following example makes instance boundaries drawn in `red`.

```
set_layer_attributes -lpp { "instance:boundary"} -color red
```

set_layer_opacity

```
set_layer_opacity
     -lpp {s_layerlpp …}
     -opacity i_opacity | -lpp_and_opacity {{s_layerlpp i_opacity} …}]
     [ -window_id i_windowID]
```

Sets the opacity for layer purposes and objects. Multiple objects may be specified with a single opacity or with an opacity for each object.

Arguments

`-lpp` {*`s_layerlpp`* …}	Specifies the list of layer names, layer purposes, and objects to assign the color to. Layer purposes are represented as strings in the following format *`layer`*`:`*`layer purpose`*. For example, `"met3:blockage"` Objects are similarly represented. If you specify a layer name, all layer purposes in the layer are set.
`-opacity` *`i_opacity`*	Specifies the opacity to assign to each layer purpose or object listed in the *`layer_list`*. Valid values are 0 (transparent) through 255 (most opaque).
`-lpp_and_opacity` {{*`s_layerlpp i_opacity`*} …}	
	Specifies a list of layer names, layer purposes, and/or objects with their opacity settings.
`-window_id` *`i_windowID`*	Specifies the identifier of the window to operate on. If this argument is not specified, the active window is used.

Example

The following example assigns an opacity of `20`0 to all met2 layer purposes, and `met1` layer purposes `blockage`, `wire:pin` and `wire:detail`.

```
set_layer_opacity -lpp { "met1:blockage" "met1:wire:pin" "met1:wire:detail"
"met2"} -opacity 90
```

The following example assigns all `met2` layer purposes an opacity of 60, and `met1:wire:detail` layer purposes an opacity of `209`.

```
set_layer_opacity -lpp_and_opacity { "met2 60" "met1:wire:detail 209" }
```

show_view_context

```
show_view_context
     -name s_vcName
     [ -layers_only | -area_only]
```

Recalls the specified view context. The view context includes the visibility setting, active state and opacity of layers and objects, start and end hierarchy levels, global control settings, instance boundary and label settings, and magnification and position of the active window in the cellview. When optional arguments are omitted, this command is equivalent to restoring a view context from the View Contexts Browser.

Arguments

`-area_only`	Restores only the magnification and position stored in the view context.
`-layers_only`	Restores only the layer display information from the view context.
`-name` *`s_vcName`*	Specifies the name for the view context.

Example

The following example recalls the view context stored in `mySavedVC`.

```
show_view_context -name mySavedVC
```

Related Information

Tcl Commands	create_view_context get_view_contexts

view_layer

```
view_layer
     {-all layers [ true | false ] | -lpp {s_layerlpp …}}
     -visible [ true | false ]
     [ -norepaint ]
     [ -window_id i_windowID ]
```

Sets the visibility of a layer or object. The window is refreshed unless the `-norepaint` is specified.

Arguments

`-all_layers [ true | false ]`

Sets the visibility for all layer purposes that correspond to physical layers.

`-lpp` {*`s_layerlpp…`*}

Specifies the list of layer purposes and objects to assign the color to. Layer purposes are represented as strings in the following format *`layer`*`:`*`layer purpose`*. For example, `"met3:Blockage"` Object types are similarly represented.

`-norepaint`

Overrides the default action to refresh the window after this command is processed. This option is useful when you are issuing commands from a Tcl script and want to delay refreshing the window until all commands have been issued.

`-visible [ true | false ]`

Specifies the visibility of the items in the *`layer_list`*. If no setting is specified, items in the list are made visible.

`-window_id` *`i_windowID`*

Specifies the identifier of the window to refresh. If this option is not used, the active window is refreshed.

Example

The following example hides the `blockage` and `wire:pin` layer purposes of `met3`.

```
view_layer -visible false -lpp { "met3:blockage" "met3:wire:pin" }
```

The following example makes `guides` and the `wire:detail` layer purpose of `met2` visible.

```
view_layer -visible -lpp { guides met2:wire:detail }
```

The following example makes all layer purposes in the active artwork visible.

```
view_layer -visible true -all_layers
```

zoom

```
zoom
     [ -in | -out ]
     [ f_factor ]
     [ -window_id i_windowID ]
```

Changes the width of the displayed data and the center of the active artwork window or a given window. You can run `zoom` at any time, including within other commands. If you run `zoom` more than once, the size of the view changes by the zoom factor each time.

The `zoom` command handles interrupts, and might leave a partially drawn window on the screen.

Arguments

`-in`	Shows less of the design but more detail. This is the default zoom direction.
`-out`	Shows less detail but more of the design.
`f_factor`	A positive real number, greater than 1,designating the rescaling of detail relative to the existing view. For example, using `-in`, detail size is increased by *`zoom_factor`* each time the command is used. Default: 2
`-window_id` *`i_windowID`*	Zooms in the given window. If this option is not used, the active window is used.

Example

```
zoom -out 3
```

This command shows less detail by a factor of 3.

Related Information

Menu Commands	*View—Zoom In by 2* *View—Zoom Out by 2*

5

Collaborate Commands

This chapter describes the Collaborate commands.

The commands are presented in alphabetical order.

add_arrow

```
add_arrow
     -color s_color
     -p1 {x1 y1}
     -p2 {x2 y2}
     [ -end1 arrow ]
     [ -end2 arrow ]
     [ -lineWidth width ]
```

The `add_arrow` command adds an arrow annotation to the active window.

Arguments

`-color` *s_color*	Specifies the color for the arrow as the color name (such as `blue`) or the hexadecimal representation of the color preceded by a pound sign (#). A list of colors and their hexadecimal equivalents can be found on http://eies.njit.edu/~walsh/rgb.html.
`-end1 arrow`	Specifies whether the first endpoint has an arrowhead. If this argument is omitted, no arrowhead is added.
`-end2 arrow`	Specifies whether the second endpoint has an arrowhead. If this argument is omitted, no arrowhead is added.
`-lineWidth` *i_width*	Specifies the width in pixels of the line and arrowheads (if specified). Default: 1
`-p1` **{***f_x f_y***}**	Specifies the x- and y-coordinates for the first endpoint, as two space-delimited real numbers, enclosed in braces.
`-p2` **{***f_x f_y***}**	Specifies the x- and y-coordinates for the second endpoint, as two space-delimited real numbers, enclosed in braces.

Example

The following example adds an arrow annotation, colored red, with an arrowhead at the second endpoint.

```
add_arrow -color red -lineWidth 2 -p1 {132.1 170} -p2 {134.5 171.2} -end2 arrow
```

Related Information

Tcl Command	arrow

add_dimension

```
add_dimension
     -arrowLine {f_x f_y}
     -color s_color
     [ -font s_font ]
     -orient s_orientation
     -p1 {f_x f_y}
     -p2 {f_x f_y}
```

The `add_dimension` command adds a dimension annotation to the active window.

Arguments

`-arrowLine {`*`f_x f_y`*`}`

Specifies the coordinates for the first arrow endpoint as two space-delimited real numbers, enclosed in braces.

`-color` *`s_color`* Specifies the color for the annotation as the color name (such as `blue`) or the hexadecimal representation of the color preceded by a pound sign (#). A list of colors and their hexadecimal equivalents can be found on http://eies.njit.edu/~walsh/rgb.html.

`-font` *`s_font`* Specifies the font type as `Arial`, `Courier`, or `Times Roman`.

`-orient` *`s_orientation`*

Specifies the orientation of the measurement as one of the following:

- `horiz` indicates the measurement is on the x-axis.
- `vert` indicates the measurement is on the y-axis.
- `allangle` indicates the measurement is between two angles on a diagonal.
- `area` indicates that an area is measured.

`-p1 {`*`f_x f_y`*`}` Specifies the x- and y-coordinates for the first endpoint as two space-delimited real numbers, enclosed in braces

`-p2 {`*`f_x f_y`*`}` Specifies the x- and y-coordinates for the second endpoint as two space-delimited real numbers, enclosed in braces

Related Information

Tcl Command	dimension

add_rectangle

```
add_rectangle
     -color s_color
     -rect {f_xlo f_ylo f_xhi f_yhi}
     [ -fillColor s_color ]
     [ -lineWidth i_pixel ]
     [ -opacity i_opacity ]
     [ -text s_text ]
```

Adds a rectangle annotation to the active window.

Arguments

`-color` *s_color*

Specifies the color for the rectangle as a character string (such as `blue`) or in the hexadecimal representation of the color preceded by a pound sign (#). A list of colors and their hexadecimal equivalents can be found on http://eies.njit.edu/~walsh/rgb.html.

`-fillColor` *s_color*

Specifies the color for the interior of the rectangle as a character string (such as `blue`) or in the hexadecimal representation of the color preceded by a pound sign (#).

Default: No fill

`-lineWidth` *i_pixel*

Specifies the width (in pixels) of the line.

Default: 1

`-opacity` *i_opacity*

Specifies the opacity for rectangle fill. Valid values are 0 (transparent) through 255 (opaque). If `-fillColor` is not given, the `-color` setting is used for the fill.

Default: 255 (opaque) when `-fillColor` is given.

`-rect` **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the bounding box coordinates for the rectangle.

`-text` *s_text*

Specifies text to fit in the rectangle. If this argument is given, `-fillColor` and -`opacity` are ignored.

Default: No text is included.

Example

The following example adds a rectangle annotation, colored blue, with enclosed text of `vm1`.

```
add_rectangle -color blue -lineWidth 2 -rect {120.1 132.3 123.6 134.1} -text vm1
```

Related Information

Tcl Command	add_arrow

add_text

```
add_text
     -balloonColor s_color
     -font s_font
     -halign s_hPos
     -tailLoc {f_x f_y}
     -text s_text
     -textColor s_color
     -textLoc {f_x f_y}
     -valign s_vPos
```

The `add_text` command adds a text annotation to the active window.

Arguments

`-ballooncolor` *`s_color`*

Specifies the color for the balloon as the color name (such as `blue`) or the hexadecimal representation of the color preceded by a pound sign (#). A list of colors and their hexadecimal equivalents can be found on http://eies.njit.edu/~walsh/rgb.html.

`-font` *`s_font`* Specifies the font type as `Arial`, `Courier`, or `Times Roman`.

`-halign` *`s_hPos`* Specifies the horizontal alignment for the text as `left`, `right`, or `hcenter.`

`-tailLoc` **{***`f_x f_y`***}** Specifies the x- and y-coordinates for the end of the balloon tail as two space-delimited real numbers enclosed in braces.

`-text` *`s_text`* Specifies the text.

`-textcolor` *`s_color`* Specifies the color for the text as the color name (such as `blue`) or the hexadecimal representation of the color preceded by a pound sign (#).

`-textLoc` **{***`f_x f_y`***}** Specifies the x- and y-coordinates for the end of the balloon tail as two space-delimited real numbers enclosed in braces.

`-valign` *`s_vPos`* Specifies the vertical alignment for the text as `top`, `bottom`, or `vcenter`.

Example

The following example adds a text annotation.

```
add_text -balloonColor Yellow -font Arial -halign left -tailLoc { 1374.54 1993.79
} -text "look here" -textColor #55ffff -textLoc { 1411.46 2167.85 } -valign bottom
```

Related Information

Tcl Command	add_arrow add_rectangle

arrow

```
arrow
     [ -once | -repeat ]
```

Sets the mouse command field to `arrow` and lets you interactively draw arrow annotations in the artwork using the mouse. This command is invoked when you choose *Collaborate—Arrow Mode* from the menubar or the arrow icon in the toolbar.

If neither `-once` nor `-repeat` is given, the `cmd.repeating` and `cmd.arrow.repeating` environment variables, described in Table 21-1 on page 1165, determine whether the command is automatically repeated.

Arguments

`-once`	Permits only one arrow annotation to be added, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits multiple arrow annotations to be added, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Command	add_arrow

dimension

```
dimension
     [ -once | -repeat ]
```

Sets the mouse command field to `dimension` and lets you interactively draw dimension annotations in the artwork using the mouse. This command is invoked when you choose *Collaborate—Dimension Mode* from the menubar or the dimension icon in the toolbar.

If neither `-once` nor `-repeat` is given, the `cmd.repeating` and `cmd.dimension.repeating` environment variables, described in Table 21-1 on page 1165, determine whether the command is automatically repeated.

Arguments

`-once`	Permits only one dimension annotation to be added, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits multiple dimension annotations to be added, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Command	add_dimension

map_annotations

```
map_annotations
     -rule s_ruleName
     -lpp s_lpp
     [ -annotation_limit i_count ]
     [ -keep_original [ true | false ] ]
```

Maps a specific rule marker to a layer purpose. This is particularly useful when loading rule markers for which no layers have been defined.

- When mapped to a specific layer purpose using this command, the annotations that represent the imported rule markers will be displayed in the artwork on the given layer purpose.
- All mapped annotations will appear in the Violations Browser, under *Annotation Mapper* and grouped by their rule name.
- The original unmapped annotations will be removed, unless you specify `-keep_original`.

Arguments

`-annotation_limit i_count`

Processes up to the specified number of annotations. Specify a value of `-1` to process an unlimited number of annotations.

Default: 1000

`-keep_original [ true | false ]`

When `true`, prevents the original unmapped annotations from being removed.

Default: `false`

`-lpp s_lpp` Specifies the layer purpose to which the specified rule marker will be mapped.

`-rule s_ruleName` Specifies the name of the rule to which the specified layer purpose will be mapped.

Example

The following command maps the `yrcmA_m02` rule to the `M02:annotation:violation` layer purpose.

```
map_annotations -rule yrcmA_m02 -lpp M02:annotation:violation
```

Related Information

Tcl Command	read_calibre_errors

read_calibre_errors

```
read_calibre_errors
     -file s_fileName
     [ -max_shape_count i_count ]
```

Reads a Mentor Graphics® Calibre® physical verification tool ASCII error file and converts the violation markers to annotations. You can view the annotations in list form in the Violations page of the Annotation Browser, then use the Browser to locate the annotations in the workspace.

Arguments

`-file` *`s_fileName`*	Specifies the name of the Calibre file to read.
`-max_shape_count` *`i_count`*	Specifies the maximum number of shape annotations to load for each rule. Default: 1000

Related Information

GUI	Annotation Browser *Load* with Violations listed

read_litho_errors

```
read_litho_errors
     -file s_fileName
     [ -annotation_limit i_count ]
     [ -layers {s_layerName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -severity {i_severityLevel…} ]
     [ -types { [LINEEND] [SPACING] [WIDTH] [ENCLOSURE [ABOVE|BELOW] ] [SPACEEND]
     [GATE_CD] [ s_errorTypeName ] }  ]
```

(Space-based Router only) Loads a Litho error file and creates annotations for the error markers in the error file. The error marker annotations are added to the `annotation:violation` purpose for the layer and appear on the Violations page of the Annotation Browser under *Lithography Errors*, grouped by error type, layer, and severity.

The **error types** are given in the Litho error file as follows:

- **SPACING** is a fault where two wires may be unacceptably close or merge during manufacturing, also known as *bridging*.
- **ENCLOSURE** is a fault where the contact metal of a via may not be sufficiently covered by metal to form an acceptable electrical connection during manufacturing. Enclosure error types can be qualified as **ABOVE** or **BELOW**, which specifies whether the enclosure error refers to the via contacting this layer from above or below, respectively.
- **WIDTH** is a fault where a wire may become unacceptably narrow during manufacturing. also known as *necking*.
- **LINEEND** is a fault where the wire is unacceptably shorter in length than the drawn dimension.
- **SPACEEND** is a fault where metal separating two slots narrows unacceptably.
- **GATE_CD** is a fault where the critical dimension of a device gate varies unacceptably across process variation.
- Other error types can be specified in the Litho error file. These are new error type prototypes. The semantics of the error file will uniquely identify these names.

The Litho error file can contain suggestions, or hints, for how an error can be corrected.

Arguments

`-annotation_limit` *`i_count`*

Specifies the maximum number of errors that this command can load. Specify `-annotation_limit -1` to choose no limit.

Default: Up to 1000 errors of each type will be loaded.

`-file` *`s_fileName`* Specifies the name of the Litho error file to read.

`-layers` **{***`s_layerName…`***}** Creates only annotations for the given layers.

Default: Errors for all layers are loaded.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Creates annotations only for errors that are partially or fully within the given region.

Default: Errors in the entire design are considered.

`-severity` **{***`i_severityLevel…`***}**

Creates annotations only for errors with the given severities.

Default: All severities are loaded.

`-types {[LINEEND] [SPACING] [WIDTH] [ENCLOSURE [ABOVE|BELOW]] [SPACEND] [GATE_CD] [s_errorTypeName ]}`

Creates annotations only for errors of the given types.

Default: All error types are loaded.

Example

The following example reads lithography markers from `my_litho_file`.

```
read_litho_errors -file my_litho_file
```

Related Information

Tcl Command	fix_litho_errors

read_niagara_errors

```
read_niagara_errors
     -file s_fileName
```

Reads a Design Framework II (DFII) error file that was converted from an IBM® Niagara file and creates Violation annotations for the error markers. You can view the annotations in list form in the Violations page of the Annotation Browser, then use the Browser to locate the annotations in the workspace.

Arguments

`-file` *`s_fileName`*	Specifies the name of the error file to read.

rectangle

```
rectangle
     [ -once | -repeat ]
```

Sets the mouse command field to `rectangle` and lets you interactively draw rectangle annotations in the artwork using the mouse. This command is invoked when you choose *Collaborate—Rectangle Mode* from the menubar or the rectangle icon in the toolbar.

If neither `-once` nor `-repeat` is given, the `cmd.repeating` and `cmd.dimension.repeating` environment variables, described in Table 21-1 on page 1165, determine whether the command is automatically repeated.

Arguments

`-once`	Permits only one rectangle annotation to be added, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits multiple rectangle annotations to be added, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Command	add_rectangle

remove_annotations

```
remove_annotations
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -type {all | user | violation | optimization ]
```

Removes annotations from the active cellview or from a specific area of the active artwork window.

Arguments

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the bounding box coordinates for the area to remove annotations from. If this argument is not included, annotations in the entire active design are removed.

`-type {all | user | violation | optimization}`

Determines the types of annotations to remove.

`all`	Removes all annotation types from the Annotation Browser and the related markers from the active window. This is the default.
`optimization`	Removes all annotations in the Optimizations tab of the Annotation Browser and the related markers from the active window.
`user`	Removes all annotations in the User tab of the Annotation Browser and the related markers from the active window.
`violation`	Removes all annotations in the Violations tab of the Annotation Browser and the related markers from the active window.

Example

The following example removes only annotations in the viewing area of the active artwork window.

```
remove_annotations -region [get_window_area]
```

Related Information

Menu Command	*Collaborate—Clear Annotations* *Verify—Shapes (General tab)*

send_annotations

```
send_annotations
     -from s_email
     -to s_email
     [ -msg s_text ]
     [ -subject s_text ]
     [ -violations ]
```

Mails Space-based Router and Chip Optimizer annotations for the active window to a recipient as an E-mail attachment.

Arguments

`-from` *s_email*	Specifies the E-mail address of the sender.
`-msg` *s_text*	Specifies the message to include in the E-mail text.
`-subject` *s_text*	Specifies the subject line for the E-mail message.
`-to` *s_email*	Specifies the E-mail address of the recipient.
`-violations`	Writes Space-based Router and Chip Optimizer violations to the XML file. By default, user annotations are written.

Value Returned

`"The mail file was sent"`

The E-mail is successfully sent.

`"There are no annotations attached to the current document"`

The E-mail was not sent.

Example

The following example sends an E-mail with an XML file attachment of the annotations for the active window.

```
send_annotations -from me@mycompany.com -to you@mycompany.com -subject "opens"
-msg "review now"
```

Related Information

Tcl Command	send_jpeg
Menu Commands	*Collaborate—Send Annotations*

send_jpeg

```
send_jpeg
     -to s_email
     -from s_email
     [ -subject s_text ]
     [ -msg s_text ]
     [ -png | -jpeg ]
     [ -imageQuality i_ratio ]
```

Creates a JPEG graphics file for the active window. The image quality (i.e., compression ratio) can optionally be set.

Important

This command cannot be used in batch/non-graphics mode.

Arguments

`-from` *`s_email`*	Specifies the E-mail address of the sender.
`-imageQuality` *`i_ratio`*	Specifies the image quality/compression ratio. The lower the number, the lower the quality (and the higher the compression). Zero (0) specifies maximum compression (which seriously degrades quality), 100 specifies the maximum quality, -1 specifies the default ratio (currently set to 100). Default: -1
`-jpeg | -png`	Specifies the format for the graphics file as JPEG or PNG. Default: JPEG
`-msg` *`s_text`*	Specifies the message to include in the E-mail text.
`-subject` *`s_text`*	Specifies the subject line for the E-mail message.
`-to` *`s_email`*	Specifies the E-mail address of the recipient.

Value Returned

`"The image was sent"`	The E-mail is successfully sent.

Example

```
send_jpeg -from me@cadence.com -subject newchip -to eval@mygrp.com
```

This sends an E-mail to `eval@mygrp.com` from `me@cadence.com` with a subject of `newchip` and a JPEG file of the active window attached.

Related Information

Menu Commands	*Collaborate—Send Window Image* *File—Export—JPEG...*

text

```
text
     [ -once | -repeat ]
```

Sets the mouse command field to `text` and lets you interactively draw text annotations in the artwork using the mouse. This command is invoked when you choose *Collaborate—Text Mode* from the menubar or the text icon in the toolbar.

If neither `-once` nor `-repeat` is given, the `cmd.repeating` and `cmd.dimension.repeating` environment variables, described in Table 21-1 on page 1165, determine whether the command is automatically repeated.

Arguments

`-once`	Permits only one text annotation to be added, then the mouse command mode automatically reverts to `select` mode.
`-repeat`	Permits multiple text annotations to be added, until the mode is canceled or another interactive mode is enabled.

Related Information

Tcl Command	add_text

write_calibre_errors

```
write_calibre_errors
     -file s_fileName
```

Creates a Calibre format ASCII error file from the existing annotated DRC violations.

Arguments

`-file s_fileName`	Specifies the name of the file.

Example

The following example creates a Calibre error file named `drc_errors.db` containing the violations represented by existing DRC violation annotations.

Related Information

Tcl Command	read_calibre_errors

6

Create Commands

The Create commands add instances, nets, polygons, pins, pin halos, rectangles, and generated shapes from boolean and sizing operations to your design.

The commands are presented in alphabetic order:

- geom_or on page 337
- geom_remove_scratch_layers on page 341
- geom_scratch_layers_in_use on page 342
- geom_size on page 343
- geom_tiles_on_scratch_layer on page 348
- geom_xor on page 349

create_blockage

```
create_blockage
     { -set d_setObj [ -layer {s_layerName…} ]
     | -region {f_xlo f_ylo f_xhi f_yhi}
        {-layer {s_layerName …} | -placement [ true | false ]}
     | -inst {s_instName …} -layer {s_layerName …} [ -use_prBoundary
     [ true | false ] ]}
     [ -halo {f_bloatEdges | {f_left f_bottom f_right f_top}} ]
     [ -output_set d_setObj ]
```

Adds blockage shapes on the `blockage` purpose or the `placement_blockage` layer of the active design. You specify one of the following:

- A set

  Blockage shapes are added on the same layers and with the same footprint as the nets, routes, route segments and route vias in the set. You can optionally specify the layers to include.

- A region given by coordinates and the layers to add blockage shapes to that cover the region
- One or more instances and the layers to add blockage shapes to that cover the bounding box of the given instances

Arguments

`-halo {`*`f_bloatEdges`*` | `**`{`***`f_left f_bottom f_right f_top`***`}`**`}`

Specifies the halo for added blockage shapes. Positive values bloat amounts outward. Negative values shrink amounts inward.

If one argument value is given, it is applied to all blockage edges.

If four argument values are given, they apply to the left, bottom, right, and top, respectively.

`-inst `**`{`***`s_instName`*** **…}** Specifies one or more instances to create blockage shapes on each given layer of the instance's bounding box.

`-layer `**`{`***`s_layerName`*** **…}** Specifies one or more layers to create blockage shapes on.
If the `-set` argument is given and `-layer` is not included, blockage shapes are created on the layers given by the nets, routes, route segments and route vias in the set.

`-output_set `*`d_setObj`* Adds the new blockage shapes to the given set.

`-placement [ true | false ]`

(Applies only when `-region` is given) When `true`, adds a blockage shape for the entire region on the `placement_blockage` layer.

`-region `**`{`***`f_xlo f_ylo f_xhi f_yhi`*****`}`**

Specifies the area to add blockage shapes to, given the lower-left coordinate and the upper-right coordinate.

`-set `*`d_setObj`* Specifies a set. Blockage shapes are added on the same layers and with the same footprint as the nets, routes, route segments and route vias in the set. You can optionally specify the layers to include using `-layer`.

`-use_prBoundary [ true | false ]`

(Applies only when `-inst` is given) If set to `true`, each instance's prBoundary is used as the blockage region with the halo adjustments, if specified. If set to `false`, the instance's bounds are used as the initial blockage region.

Default: `false` for backwards compatibility

Value Returned

i_count	Is the total number of blockage shapes created.

Example

The following command creates two blockage shapes on `met1` and two on `met2` in the area given by the bounding boxes of the two instances, `I_8306` and `I_8307`. A value of 4 is returned for the four blockage shapes created.

```
create_blockage -inst {I_8306 I_8307} -layer {met1 met2}
```

Related Information

Tcl Commands	create_instance delete
Menu Commands	*Create—Blockage*

create_fill_cell

```
create_fill_cell
     -types {all|floating|connected|notch|OPC}
     [ -layers {all | {s_layerName…}} ]
     {[ -lib s_libName ]
      [ -cell s_cellName ]
      [ -view s_viewName ]}
     [ -replace [ true | false ] ]
     [ -instance_name s_instanceName ]
```

Creates a new cellview of fill shapes of all or specified types, from all or specific layers. Can optionally create an instance of the new cellview, remove the original fill shapes from the active view, and replace the removed fill shapes with the new instance.

Arguments

`-cell` *`s_cellName`*

Names the cell for the new cellview. By default, the active cell name is used. At least one of `-lib`, `-cell`, or `-view` must be given.

`-instance_name` *`s_instanceName`*

Names the instance of the new cellview that is created by the `-replace` argument.

`-layers {all|`**{***`s_layerName…`***}**`}`

Restricts processing to the given layers.
Default: `all`

`-lib` *`s_libName`*

Names the library for the new cellview. By default, the active library name is used. At least one of `-lib`, `-cell`, or `-view` must be given.

`-replace [ true | false ]`

If `true`, creates an instance of the cellview that is created by this command and replaces the original fill shapes in the active design with the instance. The name of the new instance is given by the `-instance_name` argument. If `-instance_name` is not given, a name is assigned that is derived from the `lib`, `cell`, and `view` arguments.
Default: `false`

`-types {all|floating|connected|notch|OPC}`

Selects the types of fill shapes to put in the new cell.

`all`	All types of fill shapes
`connected`	Fill shapes connected to power/ ground fill shapes (`fill` purpose)
`floating`	Unconnected fill shapes (`fill` purpose, not assigned to a net)
`notch`	Fill shapes used to fill notches (`gapFill` purpose)
`OPC`	Unconnected fill shapes (`opcFill` purpose)

`-view` *`s_viewName`*

Names the view for the new cellview. By default, the active view name is used. At least one of `-lib`, `-cell`, or `-view` must be given.

Example

The following example creates a new cellview for all types of fill shapes on layer `Metal2`. The new cellview is added to the active library and cell, with the view name `fill_Metal2`.

```
create_fill_cell -types all -layers Metal2 -view fill_Metal2
```

The following example creates a new cellview for notch fill shapes on all layers and creates an instance named `notchFillA`, containing the notch fill shapes. The notch fill shapes in the active view are removed and replaced by the new instance. The new cellview is `mylib/mycell/notch_fill`.

```
create_fill_cell -types notch -layers all -lib mylib -cell mycell -view notch_fill
-replace -instance_name notchFillA
```

Related Information

Tcl Commands	create_fill create_pg_fill fill_notch flatten

create_instance

```
create_instance
     -lib s_libName
     -cell s_cellName
     -view s_viewName
     [ -location {f_x f_y} ]
     [ -name instanceName ]
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
     [ -placement_status {none | unplaced | suggested | placed | locked | firm |
     cover}]
     [ -connections {s_termName:s_netName …} ]
     [ -no_connect ]
```

Adds an instance of a cellview to the design. If no arguments are given, the Create Instance form appears, allowing you to interactively specify and place the instance.

If the instance is a subdesign, instance terminals are created only when the `db.create_inst_terms_in_subdesign` environment variable is set to `true.`

Arguments

`-cell` *`s_cellName`* Specifies the cell name for the cellview to add.

`-connections` **{***`s_termName`*`:`*`s_netName`* **…}**

Creates instance terminals for the instance and connects them to the specified nets. If *`netName`* is not given, the instance terminal is connected to the default net. The colon is required.

`-lib` *`s_libName`* Specifies the library name for the cellview to add.

`-location` **{***`f_x f_y`***}** Specifies the placement location for the origin of the instance. If this option is not specified, Space-based Router and Chip Optimizer operates in interactive mode. The instance appears in the artwork window at the cursor. Drag the instance in the artwork, then click to place it.

`-name` *`s_instName`* Specifies the name to assign to the new instance.

Note: This name must be unique from all other instances in the design. If this argument is omitted, a default name is assigned, in the format `I_`*`xxxx`* where *`xxxx`* is a unique number.

`-no_connect` Prevents the instance from being automatically connected to any nets.

`-orient` *`s_value`* Specifies the orientation for the instance.

Valid values are: `R0`, `R90`, `R180`, `R270`, `MY`, `MYR90`, `MX`, and `MXR90`. For a description of orientation values, refer to “Orientation Key” on page 208 .

Default: `R0`

`-placement_status` Specifies the OpenAccess placement status.

Valid values are: `none`, `unplaced`, `suggested`, `placed`, `locked`, `firm`, `cover`.

Default value: `firm`

`-view` *`s_viewName`* Specifies the view name for the cellview to add.

Example

The following example creates an instance.

```
create_instance -lib mylib -cell ssad -view abstract -location {3200.00 1323.20}
```

Related Information

Tcl Commands	delete move
Menu Commands	*Create—Instance*

create_label

```
create_label
     -layer s_layerName
     -origin {f_x f_y}
     -label s_text
     [ -height f_userunit ]
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
     [ -justification {lowerLeft | centerLeft | upperLeft | lowerCenter |
     centerCenter | upperCenter | lowerRight | centerRight | upperRight} ]
     [ -font {stick | euroStyle | gothic | math | roman | script | fixed | swedish
     | milSpec} ]
```

Adds a label to the active design on the `text` purpose of the given layer.

Arguments

`-font`	Specifies the font style for the label. The default is `stick`.
`-height` *`f_userunit`*	Specifies the label height in user units. The default is 1.0.
`-justification`	Places the label, by specifying the relative location of the origin with respect to the text. For example, `upperLeft` places the text so that the origin is in the upper left corner of the text.
`-label` *`s_text`*	Specifies the label text. Enclose the text in double quotes if it includes embedded spaces.
`-layer` *`s_layerName`*	Specifies the name of the layer to place the label on.
`-orient`	Specifies the orientation for the text. For a description of orientation values, refer to "Orientation Key" on page 208 . The default is `R0`.
`-origin {`*`f_x f_y`*`}`	Specifies the coordinates for the point of origin.

Example

The following command creates a label, `VSS`, on the `text` purpose of the `Metal2` layer.

```
create_label -origin { 474.633 566.3} -layer Metal2 -label VSS
```

Value Returned

d_ctuObj	Is object identifier for the label.

Related Information

Tcl Commands	find_text

create_net

```
create_net
     -name s_netName
```

Adds a logical net to the design. This is helpful for ECO use, to add buffers.

Arguments

`-name` *`s_netName`*	Specifies a unique name for the logical net to add.

Example

The following command creates a logical net named `mynet`.

```
create_net -name mynet
```

Related Information

Tcl Commands	assign_term create_instance unassign_term

create_pin

```
create_pin
     -layer s_layerName
     -net_name s_netName
     -term_name s_termName
     -width f_userunit
     -height f_userunit
     -origin {f_x f_y}
     [ -create_term ]
```

Creates a pin and rectangle, and optionally a term.

Arguments

`-create_term`	Creates the specified term name if it does not already exist. By default this is not done.
`-height` *`f_userunit`*	Specifies the height of the pin rectangle. This value must be equal to or greater than the minimum width rule for the layer.
`-layer` *`s_layerName`*	Specifies the name of the layer to put the pin rectangle on.
`-net_name` *`s_netName`*	Specifies the name of the net to put the pin on.
`-origin` **{***`f_x f_y`***}**	Specifies the center position of the pin rectangle.
`-term_name` *`s_termName`*	Specifies the term name on the net to put the pin on.
`-width` *`f_userunit`*	Specifies the width of the pin rectangle. This value must be greater than or equal to the minimum width rule for the layer.

Example

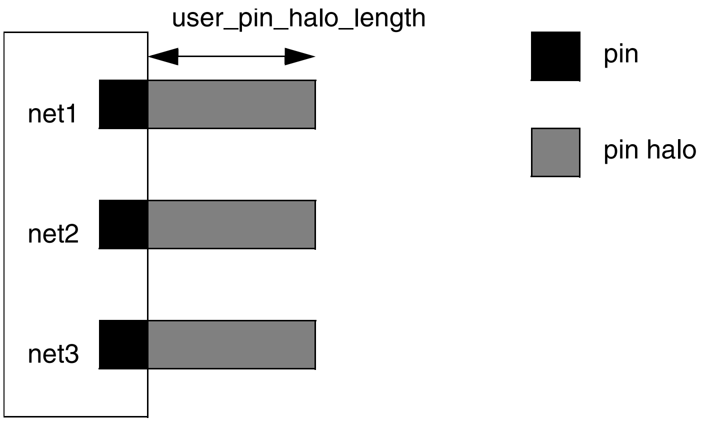
The following command creates a pin rectangle on `Metal1` with a term_name `CC` on `netA`.

```
create_pin -layer Metal1 -net_name netA -term_name CC -width .35 -height .4
-create_term
```

create_pin_halo_shapes

```
create_pin_halo_shapes
     [ -all ]
     [ -include_power_gnd_nets ]
     [ -inst_set d_instSetObj ]
     [ -multi_pin_terminals ]
     [ -on_all_valid_layers ]
     [ -set d_setObj ]
     [ -user_pin_halo_length f_length ]
```

Creates pin halo shapes on the terminals for all the nets in the design, for the selected nets, or on the specified instances. Power and ground nets are excluded from processing unless you include the `-include_power_gnd_nets` argument. The pin halo creates an area around a pin that is reserved for the net assigned to the pin. This prevents different-net routing from blocking access to the pin. Use delete_floating_trims to remove pin halo shapes after routing.



Arguments

`-all`
Creates pin halo shapes on all terminals for all the nets in the design.

`-include_power_gnd_nets`
Creates pin halo shapes on power and ground nets. By default, power and ground nets are excluded from processing.

`-inst_set` *`d_instSetObj`*
Creates pin halo shapes on terminals of the selected instances.

`-multi_pin_terminals`
Creates pin halo shapes for all pins of a multi-pin terminal. By default, a pin halo is created for only one pin of a multi-pin terminal.

`-on_all_valid_layers`
Creates pin halo shapes on all valid routing layers. By default, pin halo shapes are created only on the pin layer, and the layers above and below the pin layer.

`-set` *`d_instSetObj`*
Creates pin halo shapes on all the terminals of the nets in the selected set.

`-user_pin_halo_length` *`f_userunit`*
Length, in user units, of the pin halo shapes to be created. The default is 0.0.

Example

The following command creates pin halo shapes on instance terminals of instances with names beginning with `FINST`.

```
create_pin_halo_shapes(-inst_set(find_instance -window_id 1 -name "FINST*"))
```

Related Information

Tcl Commands	delete_floating_trims

create_polygon_shape

```
create_polygon_shape
     -lpp s_lpp
     -points {f_x1 f_y1 f_x2 f_y2… f_xn f_yn [ f_x1 f_y1 ]}
```

Creates a polygon shape on the given layer purpose at the given coordinates.

Arguments

`-lpp` *`s_lpp`*	Specifies the layer purpose to add the polygon shape to.
`-points` **{** *`f_x1 f_y1 f_x2 f_y2`* … *`f_xn f_yn`* `[` *`f_x1 f_y1`* `]` **}**	Specifies the boundary points for the polygon. The polygon will be closed even if you do not include the starting point (*`f_x1 f_y1`*) at the end of the list.

Example

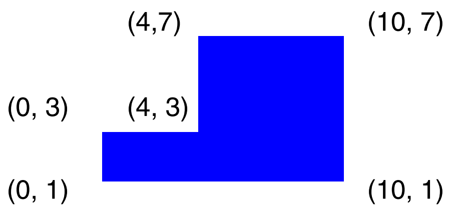
The following command creates the polygon shape shown in Figure 6-1 on the `Metal1:wire:pin` purpose.

```
create_polygon_shape -points {0 1 10 1 10 7 4 7 4 3 0 3 0 1} -lpp Metal1:wire:pin
```

The following equivalent command does not include the starting point at the end of the list.

```
create_polygon_shape -points {0 1 10 1 10 7 4 7 4 3 0 3} -lpp Metal1:wire:pin
```

Figure 6-1 Polygon Shape



Related Information

Tcl Commands	create_rect

create_rect

```
create_rect
     [ -sync ]
```

Activates `create_rect` as the current interactive mode. You are prompted to click in the artwork to mark the first point, then the opposite corner. The new rectangle is added to the current Entry Layer.

Arguments

`-sync`	Copies the new rectangle to the OpenAccess database.

Related Information

Tcl Commands	delete move
Menu Commands	*Create—Rectangle*

create_rect_shape

```
create_rect_shape
     -lpp s_lpp
     -region {f_xlo f_ylo f_xhi f_yhi}
```

Creates a rectangle shape on the given layer purpose at the given coordinates. Use the create_rect command to interactively create rectangle shapes.

Arguments

`-lpp` *`s_lpp`* Specifies the layer purpose to add the rectangle shape to.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area for the rectangle shape, given the lower left and upper right coordinates.

Related Information

Tcl Commands	create_rect
Menu Commands	*Create—Rectangle*

create_routing_cell

```
create_routing_cell
     {[ -lib s_libName ]
      [ -cell s_cellName ]
      [ -view s_viewName ]}
     [ -instance_name s_instanceName ]
     [ -no_pins [ true | false ] ]
     [ -replace [ true | false ] ]
     [ -ignore_fixed [ true | false ] ]
     [ -ignore_locked [ true | false ] ]
     [ -set d_setObj ]
```

Creates a new cellview containing all routes and route shapes. Pin and term shapes are not included. Pins are created from all route shapes on any route that connects to a term or InstTerm in the original occurrence (for example, `wire:detail` becomes `wire:pin`, `via:detail` becomes `via:pin`). If you choose to replace the original routing with an instance of the routing cell (-`replace`), instTerms will be created for all of the terms on the routing cell and those instTerms will be put on the appropriate nets.

Arguments

`-cell s_cellName` Names the cell for the new cellview. By default, the active cell name is used. At least one of `-lib`, -`cell`, or -`view` must be given.

`-ignore_fixed [ true | false ]`

When `true`, fixed routes will not be created in the new cellview. By default, fixed routes are included.

`-ignore_locked [ true | false ]`

When `true`, locked routes will not be created in the new cellview. By default, locked routes are included.

`-instance_name s_instanceName`

Names the instance of the new cellview that is created by the `-replace` argument.

`-lib s_libName` Names the library for the new cellview. By default, the active library name is used. At least one of `-lib`, -`cell`, or -`view` must be given.

`-no_pins [ true | false ] ]`

When `true`, prevents pins and terminals from being created on the new cell.

Default: `false`

`-replace [ true | false ] ]`

If `true`, creates an instance of the cellview that is created by this command and replaces the original routing shapes in the active design with the instance. The name of the new instance is given by the -`instance_name` argument. If -`instance_name` is not given, a name is assigned that is derived from the `lib`, `cell`, and `view` arguments.
Default: `false`

`-set d_setObj` Limits processing to route and via objects in the given set.

`-view s_viewName` Names the view for the new cellview. By default, the active view name is used. At least one of `-lib`, -`cell`, or -`view` must be given.

Example

The following example creates a new cellview containing all of the routing in the active design. The new cellview is added to the active library and cell, with the view name `routingA`.

```
create_routing_cell -view routingA
```

The following example creates a new cellview containing all of the routing in the active design and creates an instance of the new cellview named `routingInst`. The new cellview is `mylib/mycell/routingOnly`.

```
create_routing_cell -lib mylib -cell mycell -view routingOnly -replace
-instance_name routingInst
```

Related Information

Tcl Commands	flatten

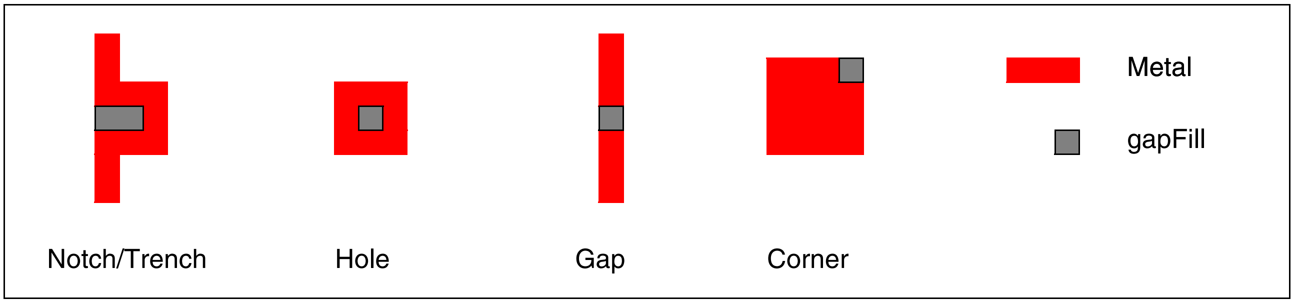
fill_notch

```
fill_notch
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -lpp {s_layerlpp …} ]
     [ -output_lpp {s_layerlpp…} ]
     [ -width f_userunit ]
     [ -trench_depth f_maxTrenchDepth ]
     [ -trench_width f_maxTrenchWidth ]
     [ -use_notch_spacing_rule [ true | false ] ]
     [ -fill_notch_set d_setObj ]
     [ -set d_setObj ]
     [ -annotate [ all | none | filled | unfilled ]
     [ -annotation_limit i_count ]
     [ -clear_annotations [ all | none | filled | unfilled ]
     [ -check_mode { hard | soft } ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -fill_type { all | {[notch] [corner] [gap] [trench]}} ]
     [ -ignore_active_route_status [ true | false ] ]
     [ -prop_name s_propertyName -prop_value s_propertyValue ]
     [ -trim_corners [ true | false ] ]
     [ -top_level_only [ true | false ] ]
     [ -use_fill_purpose [ true | false ] ]
```

Adds metal fill to notches, holes, gaps, and trenches that are smaller than a minimum required size, or to corners that do not meet the oa`MinEdgeAdjacentLength` or `minEdgeMaxCount` rules. By default, the shapes are added to the gap`Fill` purpose for the metal layer.

- A notch is a minimum spacing violation between two edges with a common adjacent side.
- A trench is a shallow notch, a user-defined width and depth violation between two edges with a common adjacent side where the notch width is greater than the notch depth.
- A hole is a minimum enclosure violation.
- A gap is a minimum spacing violation between two shapes on the same net and layer.

- A corner is an `oaMinEdgeAdjacentLength` violation between adjacent edges. If the `oaMinEdgeAdjacentLength` rule is not set, then corners can be `minEdgeMaxCount` violations.



Arguments

`-all` Searches the entire design for notches to fill.

`-annotate [ all | none | filled | unfilled ]`

Specifies the annotations to create with this command. By default, annotations are not created.

`all`	Creates annotations for notches, trenches, gaps, holes, and corners that are filled or left unfilled by this command. This is the default when the `-annotate` argument is given.
`none`	Prevents the creation of any annotation by this command.
`filled`	Creates annotations for notches, trenches, gaps, holes, and corners that are filled by this command.
`unfilled`	Creates annotations for notches, trenches, gaps, holes, and corners that are left unfilled by this command.

`-annotation_limit` *`i_count`*

Limits the total number of annotations that can be created by this command. A value of -1 specifies no limit. Default is 1000.

`-check_mode { soft | hard }`

Controls the lookup of constraints or rules used for checking.

`hard`	Checks using hard constraints only.
`soft`	Checks soft (preferred) rules first, then will check against the hard rule if the soft rule is not found. This is the default.

`-clear_annotations [ all | none | filled | unfilled ]`

Specifies the fill notch annotations to remove before running this command.

`all`	Removes all annotations previously created by this command. This is the default when the `-clear_annotations` argument is given.
`none`	Prevents annotations from being removed.
`filled`	Removes annotations for notches, trenches, gaps, holes, and corners that were previously filled by this command.
`unfilled`	Removes annotations for notches, trenches, gaps, holes, and corners that were previously left unfilled by this command.

`-exclude_net` **{***`s_netName`*…**}**

Specifies the names of nets to exclude from processing. By default, all nets are considered.

`-exclude_type` **{** `[power] [ground] [clock]` **}**

Specifies the net types to exclude from processing. By default, all net types are considered.

`-fill_notch_set` *`d_setObj`*

Adds the new fill shapes to the given set. You can use this to add annotations to a highlight set. For example, this argument,

```
-fill_notch_set [get_highlight -name HL1]
```

adds the new fill shapes to the `HL1` highlight set.

`-fill_type { all | {[notch] [corner] [gap] [trench] } }`

Specifies the type of space to fill or `all` types.

Note: Currently, gaps, notches and holes are filled when you choose `gap`.

`-ignore_active_route_status [ true | false ]`

When set to `true`, notches and gaps belonging to fixed routes are also filled.

Default: `(false)` Notches and gaps on fixed routes are not filled.

`-lpp {`*`s_layerlpp`*` …}` Specifies the layers and/or layer purposes to check. By default, all routing layers are checked.

`-output_lpp {`*`s_layerlpp`*` …}`

Creates fill shapes on a specific layer or purpose. By default, fill shapes are created on the `gapFill` purpose of the input layer.

`-prop_name` *`s_propertyName`*

Attaches the given property to the added fill shapes. Must be specified with `-prop_value`. The property can be viewed using the Properties Browser or queried using inspect_getprop or inspect_prop.

`-prop_value` *`s_propertyValue`*

Assigns the value to the property given by `-prop_name`. The property can be viewed using the Properties Browser or queried using inspect_getprop or inspect_prop.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the boundary of the area to process, given the lower left and upper right coordinates. If not specified, the entire design is processed.

`-set` *`d_setObj`* Adds fill shapes to the specified set.

`-top_level_only [ true | false ]`

Specifies that only top-level shapes be checked against all levels. Default is `false`.

`-trench_depth` *`f_maxTrenchDepth`*

Specifies the maximum depth of a trench to be filled. If this argument is not given, trench filling will not occur.

`-trench_width` *`f_maxTrenchWidth`*

Specifies the maximum width of a trench to be filled. If this argument is not given, trench filling will not occur.

`-trim_corners [ true | false ]`

Trims corners when diagonals are present.

`-use_fill_purpose [ true | false ]`

Specifies that shapes be added to the `fill` purpose, instead of the `gapFill` purpose. Default is `false`.

`-use_notch_spacing_rule [ true | false ]`

Uses the `oaMinNotchSpacing` constraint to determine whether a notch needs to be filled. By default and when `false`, notches are filled if they violate the minimum spacing rule.

`-width` *`f_userunit`*

Specifies the minimum width spacing. Fills notches, holes and gaps that are smaller than this width. By default, the minimum same net spacing from the technology file is used.

Value Returned

`i_count`

Is the total number of added shapes.

Example

The following example finds all `Metal2` notches and holes that are smaller than 0.35 um wide and fills them with shapes on `Metal2:gapFill`.

```
fill_notch -all -lpp Metal2 -width .35 -fill_type notch
```

The following example fills corners by first setting the minimum edge lengths for adjacent edges, then issuing the `fill_notch` command to fill corners that do not meet the requirements within a region on `Metal2`.

```
set_constraint_parameter -name adjacentLength -Value 0.26
set_layer_constraint -layer Metal2 -constraint oaMinEdgeAdjacentLength -Value 0.2
fill_notch -lpp Metal2 -fill_type corner -region [get_window_area]
```

Related Information

Tcl Command

check_minarea

geom_add_shape

```
geom_add_shape
     -scratch_layer i_scratchID
     {[ -rect { f_xlo f_ylo f_xhi f_yhi } ]
      [ -rects {[ { f_xlo f_ylo f_xhi f_yhi }… ]
               [ { f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 … f_xn f_yn }… ]}
```

Adds one or more rectangle or polygon shapes to the given scratch layer. These shapes can be:

- Used directly with Boolean operations
- Used to indicate regions to operate on by commands such as find_shape and proute_create_via_array.

Arguments

`-rect {f_xlo f_ylo f_xhi f_yhi}`

Specifies the location for the rectangle shape, given the lower left and upper right coordinates.

`-rects {{f_xlo f_ylo f_xhi f_yhi}… {f_x1 f_y1 … f_xn f_yn}…}`

Specifies one or more rectangle shapes (given by the lower left and upper right coordinates) and/or one or more polygonal shapes (given by at least **four** x-y coordinate pairs in rectilinear sequence and the first x-y pair must be the same as the last x-y pair).

`-scratch_layer i_scratchID`

Adds the shapes to the scratch layer identified by the positive non-zero integer.

Example

The following command adds a rectangle and a polygon shape to scratch layer `2`.

```
geom_add_shape -rects {{20 20 30 30} {0 1 10 1 10 7 4 7 4 3 0 3 0 1}} -scratch_layer 2
```

The following command, using `-rect`, adds a rectangle to scratch layer `2`.

```
geom_add_shape -rects {20 20 30 30} -scratch_layer 2
```

The following command, using `-rects`, adds a rectangle to scratch layer `2`.

```
geom_add_shape -rects {{20 20 30 30}} -scratch_layer 2
```

Related Information

Tcl Commands	geom_and geom_and_not geom_not geom_size geom_or geom_remove_scratch_layers geom_scratch_layers_in_use geom_xor

geom_and

```
geom_and
     {{-lpp1 {s_layerlpp …}
       [ -lpp1_levels {i_startLevel [i_endLevel]} ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names {s_netName…} | -lpp1_net_set d_setObj ]}
     | -input1_scratch_layer i_scratchID}
     {{-lpp2 {s_layerlpp …}
       [ -lpp2_levels {i_startLevel [i_endLevel]} ]
       [ -set2 d_setObj ]
       [ -lpp2_net_names {s_netName…} | -lpp12_net_set d_setObj ]}
     | -input2_scratch_layer i_scratchID}
     {{-output_lpp s_lpp [ -output_net_name s_netName ][ -output_set d_setObj ]}
     | -output_scratch_layer i_scratchID}
     [ -no_sync ]
     [ -polygons ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -route_topology {core_ring | block_ring | stripe | cell_row_strap} ]
```

Generates new shapes where shapes from one input group overlap shapes from another input group. Input groups can be a scratch layer, or one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, a specific net. You can operate on a specific region or the entire design.

Arguments

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-input2_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp2` shapes.

`-lpp1` **{***`s_layerlpp`* **…}** Specifies the first LPPs. For details on the LPP notation, refer to "Specifying Input Layer Purposes" on page 321.

`-lpp1_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`***…}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`* Limits `lpp1` shapes to the nets in the set.

`-lpp2` **{***`s_layerlpp`* **…}** Specifies the second LPPs. For details on the LPP notation, refer to "Specifying Input Layer Purposes" on page 321.

`-lpp2_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp2` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp2_net_names` **{***`s_netName`***…}**

Limits `lpp2` shapes to the named nets.

`-lpp2_net_set` *`d_setObj`*

Limits `lpp2` shapes to the nets in the set.

`-no_sync` By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`*

Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to the scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer or` `-input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`*

Adds generated shapes to the given set.

`-polygons`

Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`*

Limits `lpp1` shapes to shapes in the set.

`-set2` *`d_setObj`*

Limits `lpp2` shapes to shapes in the set.

Value Returned

i_count	Is the total number of shapes created.

Specifying Input Layer Purposes

You can use the following notation to specify one or more layers and/or layer purposes (LPPs) for this command:

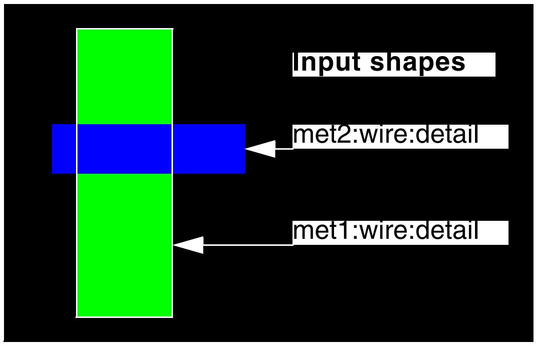
Notation	Description	Example
layer	Specifies all LPPs for the *layer*.	`met1` or "`met1`"
layer:purpose	Specifies a specific LPP.	`met1:wire:detail` `"met1:wire:detail"`

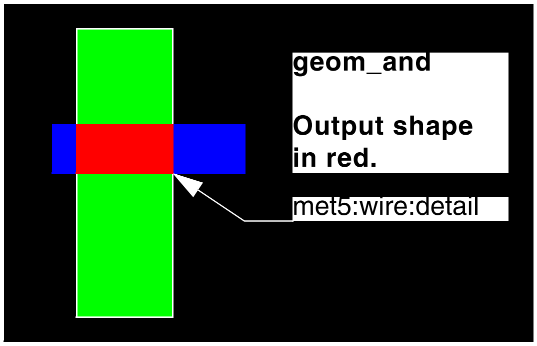
layer purpose	Specifies a specific LPP.	`met1 wire:detail`
		`"met1" "wire:detail"`

Example

The following example generates new shapes on the `met5:wire:detail` layer purpose that are common to the `met2` and `met1 wire:detail` layer purposes.

```
geom_and -lpp1 { met2:wire:detail } -lpp2 { met1:wire:detail } -region { 2486.75
2850.75 2635.22 2946.52 } -output_lpp { met5:wire:detail }
```





Related Information

Tcl Commands	geom_and_not geom_extent geom_not geom_or geom_size geom_xor
Menu Commands	*Create—Derived Layers*

geom_and_not

```
geom_and_not
     {{-lpp1 {s_layerlpp …}
       [ -lpp1_levels {i_startLevel [i_endLevel]} ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names {s_netName…} | -lpp1_net_set d_setObj ]}
     | -input1_scratch_layer i_scratchID}
     {{-lpp2 {s_layerlpp …}
       [ -lpp2_levels {i_startLevel [i_endLevel]} ]
       [ -set2 d_setObj ]
       [ -lpp2_net_names {s_netName…} | -lpp12_net_set d_setObj ]}
     | -input2_scratch_layer i_scratchID}
     {{-output_lpp s_lpp [ -output_net_name s_netName ][ -output_set d_setObj ]}
     | -output_scratch_layer i_scratchID}
     [ -no_sync ]
     [ -polygons ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -route_topology {core_ring | block_ring | stripe | cell_row_strap} ]
     [ -size1 f_userunit ]
     [ -size2 f_userunit ]
     [ -trim_corners ]
```

Generates new shapes where shapes from one input group do not overlap shapes from another input group. Input groups can be a scratch layer, or one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. You can optionally use sized shapes from one or both of the input groups. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, a specific net. You can choose to operate on a specific region or the entire design.

Arguments

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-input2_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp2` shapes.

`-lpp1` **{***`s_layerlpp`* **…}**

Specifies the first LPPs. For details on the LPP notation, refer to "Specifying Input Layer Purposes" on page 321.

`-lpp1_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`***…}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`*

Limits `lpp1` shapes to the nets in the set.

`-lpp2` **{***`s_layerlpp`* **…}**

Specifies the second LPPs. For details on the LPP notation, refer to "Specifying Input Layer Purposes" on page 321.

`-lpp2_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp2` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp2_net_names` **{***`s_netName`***…}**

Limits `lpp2` shapes to the named nets.

`-lpp2_net_set` *`d_setObj`*

Limits `lpp2` shapes to the nets in the set.

`-no_sync`

By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`*

Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer` or `-input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`*

Adds generated shapes to the given set.

`-polygons`

Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`* Limits `lpp1` shapes to shapes in the set.

`-set2` *`d_setObj`* Limits `lpp2` shapes to shapes in the set.

`-size1` *`f_userunit`* Specifies the non-negative sizing amount in user units for shapes on the first layer purpose. By default, no sizing is done.

`-size2` *`f_userunit`* Specifies the non-negative sizing amount in user units for shapes on the second layer purpose. By default, no sizing is done.

`-trim_corners` Specifies that corners be trimmed when shapes are resized. By default, corners are preserved when resized.

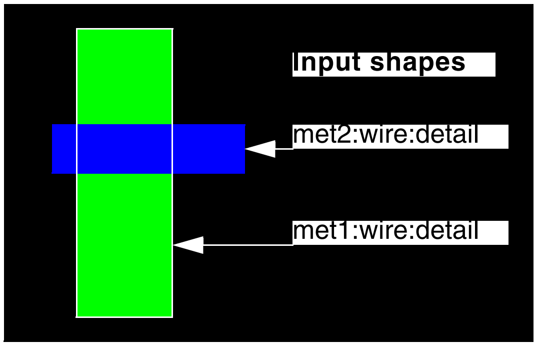
Value Returned

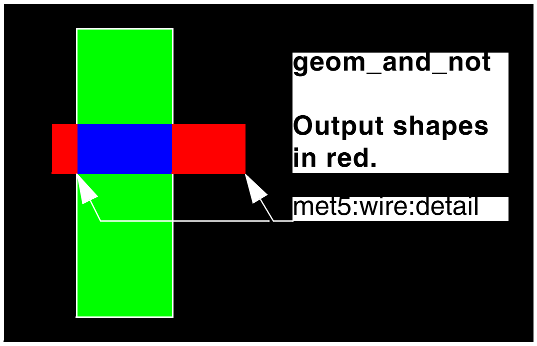
`i_count` Is the total number of shapes created.

Example

The following example generates new shapes on the `met5:wire:detail` layer purpose that exist on `met2` and do not overlap shapes on `met1 wire:detail` layer purposes.

```
geom_and_not -lpp1 { met2:wire:detail } -lpp2 { met1:wire:detail } -region {
2486.75 2850.75 2635.22 2946.52 } -output_lpp { met5:wire:detail }
```





Related Information

Tcl Commands	geom_and geom_extent geom_not geom_or geom_size geom_xor
Menu Commands	*Create—Derived Layers*

geom_create_scratch_layer

```
geom_create_scratch_layer
```

Creates a scratch layer that can be used with Boolean operations.

Arguments

None

Value Returned

`i_scratchID`	Specifies the scratch layer identifier for the new, empty scratch layer.

Example

The following command creates a new scratch layer, then adds a rectangular shape to the scratch layer.

```
set scratchLayerID [geom_create_scratch_layer]
geom_add_shape -scratch_layer $scratchLayerID -rect {10 10 20 20}
```

Related Information

Tcl Commands	geom_add_shape geom_and geom_and_not geom_extent geom_not geom_size geom_or geom_remove_scratch_layers geom_scratch_layers_in_use geom_xor

geom_extent

```
geom_extent
     { {-lpp1 { s_layerlpp … }
       [ -lpp1_levels { i_startLevel [i_endLevel] } ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names { s_netName… } | -lpp1_net_set d_setObj ] }
     | -input1_scratch_layer i_scratchID }
     { {-output_lpp s_lpp [ -output_net_name s_netName ] [ -output_set d_setObj
     ] }
     | -output_scratch_layer i_scratchID }
     [ -no_sync ]
     [ -region { f_xlo f_ylo f_xhi f_yhi } ]
     [ -route_topology { core_ring | block_ring | stripe | cell_row_strap } ]
```

Generates new shapes that extend to the bounds of two or more intersecting shapes on a scratch layer, or on one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, a specific net. You can operate on a specific region or the entire design.

Arguments

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-lpp1` **{***`s_layerlpp`* **…}**

Specifies the first LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

`-lpp1_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`***…}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`*

Limits `lpp1` shapes to the nets in the set.

`-no_sync`

By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`*

Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer or -input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`*

Adds generated shapes to the given set.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`* Limits `lpp1` shapes to shapes in the set.

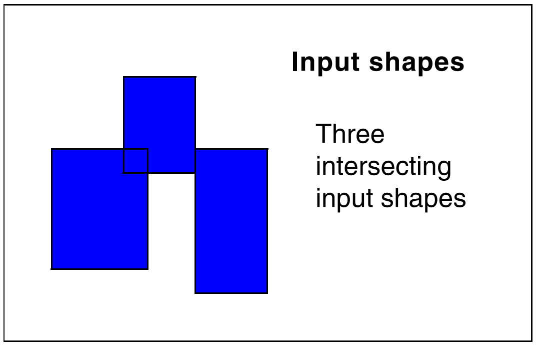
Value Returned

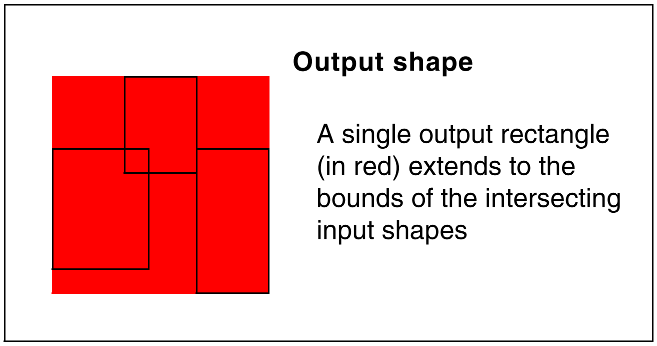
`i_count` Is the total number of shapes created.

Example

The following example creates new shapes that extend to the bounds of two or more intersecting shapes on `met1` layer purposes for the entire design and outputs the new shapes to scratch layer 1.

```
geom_extend -lpp1 met1 -output_scratch_layer 1
```




Output shape
A single output rectangle (in red) extends to the bounds of the intersecting input shapes

Related Information

Tcl Commands	geom_and geom_and_not geom_not geom_or geom_size geom_xor
Menu Commands	*Create—Derived Layers*

geom_not

```
geom_not
     { {-lpp1 { s_layerlpp … }
       [ -lpp1_levels { i_startLevel [i_endLevel] } ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names { s_netName… } | -lpp1_net_set d_setObj ]}
     | -input1_scratch_layer i_scratchID}
     { { -output_lpp s_lpp [ -output_net_name s_netName ] [ -output_set
     d_setObj ]}
     | -output_scratch_layer i_scratchID}
     [ -no_sync ]
     [ -polygons ]
     [ -region { f_xlo f_ylo f_xhi f_yhi } ]
     [ -route_topology { core_ring | block_ring | stripe | cell_row_strap } ]
```

Generates new shapes by inverting the shapes on a scratch layer, or on one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, a specific net. You can operate on a specific region or the entire design.

Arguments

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-lpp1` **{***`s_layerlpp`* …**}** Specifies the first LPPs. For details on the LPP notation, refer to "Specifying Input Layer Purposes" on page 321.

`-lpp1_levels` **{***`i_startLevel`* [*`i_endLevel`*]**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`*…**}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`* Limits `lpp1` shapes to the nets in the set.

`-no_sync` By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`* Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer or` `-input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`* Adds generated shapes to the given set.

`-polygons`

Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`*

Limits `lpp1` shapes to shapes in the set.

Value Returned

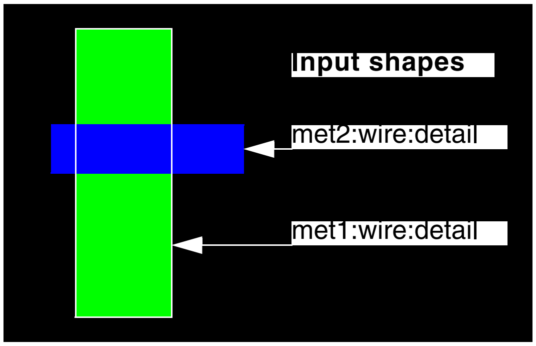
`i_count`

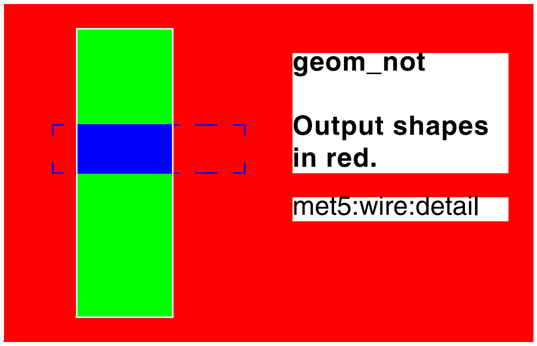
Is the total number of shapes created.

Example

The following example creates new shapes that are the inverse of existing shapes on the `met1` layer purposes and `met2:wire:detail` for the given region.

```
geom_not -lpp1 {met1 met2:wire:detail} -region 227869 2372749 230399 234515}
-output_lpp {"met5:wire:detail"}
```





Related Information

Tcl Commands	geom_and geom_and_not geom_extent geom_or geom_size geom_xor
Menu Commands	*Create—Derived Layers*

geom_or

```
geom_or
     { {-lpp1 { s_layerlpp … }
       [ -lpp1_levels { i_startLevel [i_endLevel] } ]
       [ -set1 d_setObj]
       [ -lpp1_net_names { s_netName… } | -lpp1_net_set d_setObj ] }
     | -input1_scratch_layer i_scratchID }
     { {-lpp2 { s_layerlpp … }
       [ -lpp2_levels { i_startLevel [i_endLevel] } ]
       [ -set2 d_setObj ]
       [ -lpp2_net_names { s_netName… } | -lpp12_net_set d_setObj ] }
     | -input2_scratch_layer i_scratchID}
     { {-output_lpp s_lpp [ -output_net_name s_netName ] [ -output_set
     d_setObj ] }
     | -output_scratch_layer i_scratchID }
     [ -no_sync ]
     [ -polygons ]
     [ -region { f_xlo f_ylo f_xhi f_yhi } ]
     [ -route_topology { core_ring | block_ring | stripe | cell_row_strap } ]
```

Generates new shapes where shapes from either of two input groups exist. Input groups can be a scratch layer, or one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, to a specific nett. You can operate on a specific region or the entire design.

Arguments

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-input2_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp2` shapes.

`-lpp1` **{***`s_layerlpp`* …**}** Specifies the first LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

`-lpp1_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`*…**}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`*

Limits `lpp1` shapes to the nets in the set.

`-lpp2` **{***`s_layerlpp`* …**}** Specifies the second LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

`-lpp2_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp2` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp2_net_names` **{***`s_netName`*…**}**

Limits `lpp2` shapes to the named nets.

`-lpp2_net_set` *`d_setObj`* Limits `lpp2` shapes to the given set of nets.

`-no_sync` By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`* Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer or` `-input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`* Adds generated shapes to the given set.

`-polygons` Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`* Limits `lpp1` shapes to shapes in the set.

`-set2` *`d_setObj`* Limits `lpp2` shapes to shapes in the set.

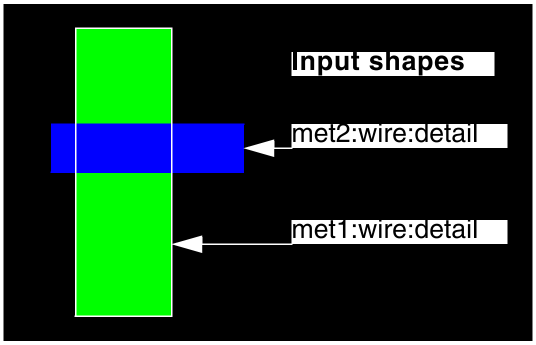
Value Returned

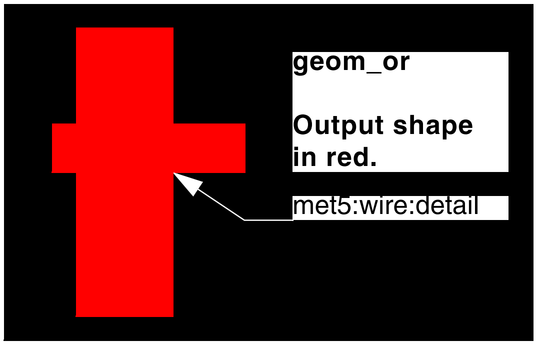
i_count	Is the total number of shapes created.

Example

The following example merges all shapes on the `met1` and `met2` and outputs them to the `met5:wire:detail` layer purpose.

```
geom_or -lpp1 {met1 met2} -output_lpp {met5:wire:detail}
```





Related Information

Tcl Commands	geom_and geom_and_not geom_extent geom_not geom_size geom_xor
Menu Commands	*Create—Derived Layers*

geom_remove_scratch_layers

```
geom_remove_scratch_layers
     { -layers { i_scratchID… } | -all [ true | false ] }
```

Removes one or more scratch layers. Scratch layers can be created to temporarily store the results of a Boolean layer operation (`geom_* -output_scratch_layer`), can be used as inputs for all Boolean operations, and are automatically removed when the current design is closed.

Arguments

`-all [ true | false ]`	If set to `true`, all existing scratch layers are removed.
`-layers {i_scratchID…}`	Removes the scratch layers corresponding to the positive integers in the list.

Example

The following example removes all scratch layers.

```
geom_remove_scratch_layers -all
```

Related Information

Tcl Commands	geom_add_shape geom_and geom_and_not geom_extent geom_not geom_size geom_or geom_scratch_layers_in_use geom_xor

geom_scratch_layers_in_use

```
geom_scratch_layers_in_use
     -report_tiles [ true | false ] ]
```

Returns a list of scratch layers that are currently in use. Scratch layers can be used by all Boolean operations (`geom_*)` and the `geom_add_shape`.

Arguments

`-report_tiles [ true | false ]`	
	When `true`, reports the number of tiles (rectangles or octagons) for each scratch layer. A single rectilinear (non-rectangular) shape consists of more than one tile. Default: `false`

Value Returned

`i_scratchID` …	Tcl list of scratch layers in use, represented by positive non-zero integers.

Example

The following example requests the list of scratch layers in use and shows the output to the Transcript area, indicating that scratch layers 6 and 3 are in use.

```
geom_scratch_layers_in_use
6 3
```

Related Information

Tcl Commands	geom_add_shape geom_and geom_and_not geom_extent geom_not geom_size geom_or geom_remove_scratch_layers geom_tiles_on_scratch_layer geom_xor

geom_size

```
geom_size
     -size f_userunit
     { {-lpp1 { s_layerlpp … }
       [ -lpp1_levels { i_startLevel [i_endLevel] } ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names { s_netName… } | -lpp1_net_set d_setObj ]}
     | -input1_scratch_layer i_scratchID}
     { {-output_lpp s_lpp [ -output_net_name s_netName ] [ -output_set d_setObj
     ] }
     | -output_scratch_layer i_scratchID }
     [ -fix_edges_to_region [ true | false ] ]
     [ -no_sync ]
     [ -polygons ]
     [ -region { f_xlo f_ylo f_xhi f_yhi } ]
     [ -route_topology { core_ring | block_ring | stripe | cell_row_strap } ]
     [ [ -size_east f_userunit ] [ -size_north f_userunit ]
       [ -size_south f_userunit ] [ -size_west f_userunit ] ]
     | [ -trim_corners ]
```

Generates new shapes by expanding or shrinking shapes on a scratch layer, or on one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, to a specific net. You can operate on a specific region or the entire design.

Arguments

`-fix_edges_to_region [ true | false ]`

If `true` and `-region` is given, then any shape that is on a region edge is not sized with respect to that edge. By default, this argument is `false` and only applies when `-region` is given.

`-input1_scratch_layer` *`i_scratchID`*

Uses a scratch layer from a previous Boolean operation for `lpp1` shapes.

`-lpp1` **{***`s_layerlpp`* **…}**

Specifies the first LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

`-lpp1_levels` **{***`i_startLevel`* `[`*`i_endLevel`*`]`**}**

Limits `lpp1` shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is `0`.
Default: all levels

`-lpp1_net_names` **{***`s_netName`***…}**

Limits `lpp1` shapes to the named nets.

`-lpp1_net_set` *`d_setObj`*

Limits `lpp1` shapes to the nets in the set.

`-no_sync`

By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

`-output_lpp` *`s_lpp`*

Specifies the output LPP for the generated shapes.

`-output_net_name` *`s_netName`*

Adds shapes to the named net. When this argument is given, the `-no_sync` argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

`-output_scratch_layer` *`i_scratchID`*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the `-input1_scratch_layer or` `-input2_scratch_layer` arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

`-output_set` *`d_setObj`*

Adds generated shapes to the given set.

`-polygons`

Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

`-route_topology` *`s_topologyName`*

(Applies only if `-output_net_name` is given with the name of an existing power or ground net) Sets the `routeTopology` property for the output shapes as follows:

`block_ring`	`BlockRingRouteTopology`
`cell_row_strap`	`StandardCellWireRouteTopology`
`core_ring`	`RingRouteTopology`
`stripe`	`StripeRouteTopology` (this is the default)

`-set1` *`d_setObj`*

Limits `lpp1` shapes to shapes in the set.

`-size` *`f_userunit`*

Specifies the sizing amount (in user units) as a real number. This value can be negative or positive. Positive values increase object sizes; negative values decrease object sizes.

`-size_east` *`f_userunit`*

Specifies the sizing amount (in user units) as a real number for the right (east) edge only. This value can be negative or positive. Positive values increase object sizes; negative values decrease object sizes. Cannot be used with -`trim_corners.`

`-size_north` *`f_userunit`* Specifies the sizing amount (in user units) as a real number for the top (north) edge only. This value can be negative or positive. Positive values increase object sizes; negative values decrease object sizes. Cannot be used with `-trim_corners`.

`-size_south` *`f_userunit`* Specifies the sizing amount (in user units) as a real number for the bottom (south) edge only. This value can be negative or positive. Positive values increase object sizes; negative values decrease object sizes. Cannot be used with `-trim_corners`.

`-size_west` *`f_userunit`* Specifies the sizing amount (in user units) as a real number for the right (west) edge only. This value can be negative or positive. Positive values increase object sizes; negative values decrease object sizes. Cannot be used with `-trim_corners`.

`-trim_corners` Specifies that corners be extended by the `size` argument value, effectively trimming corners. By default, edges are moved out by the `size` argument value, preserving corners.

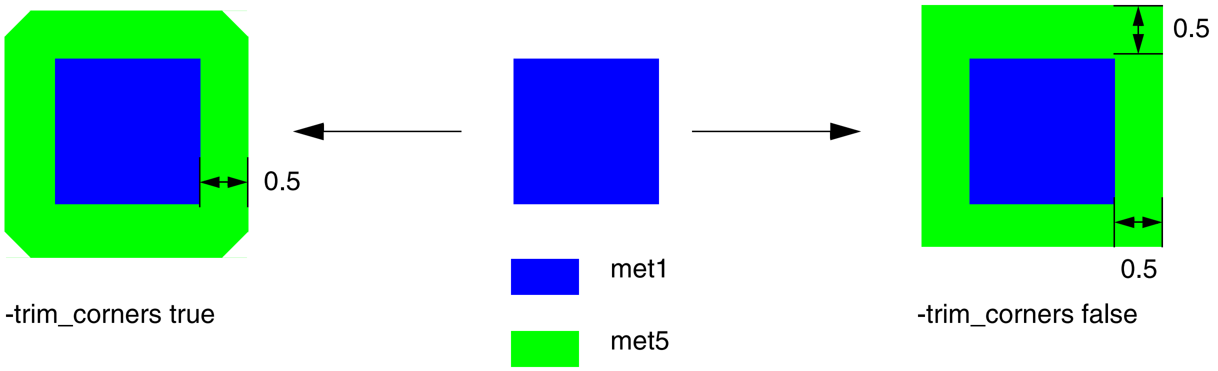
Value Returned

`i_count` Is the total number of shapes created.

Example

The following example creates new shapes on `met5:wire:detail` by expanding `met1` shapes by 0.5 user units.

```
geom_size -lpp1 { met1 } -output_lpp { met5:wire:detail } -size .5
```

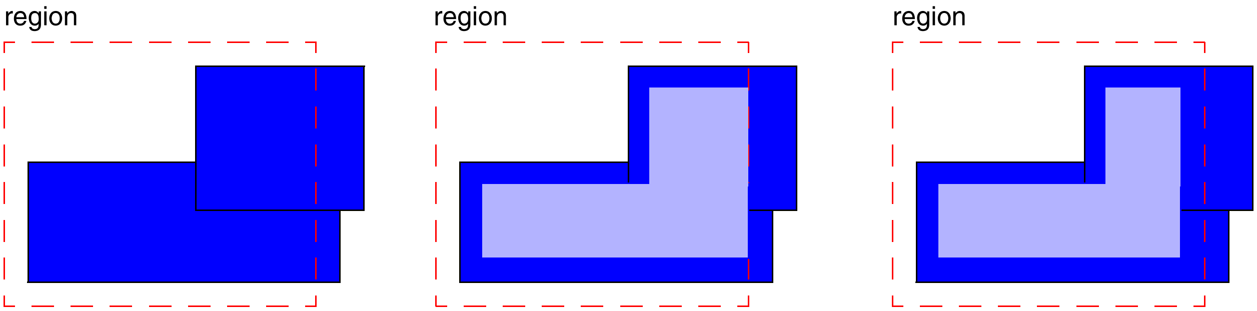


-trim_corners true

-trim_corners false

The following command shrinks shapes in a region while maintaining edges on the region boundaries.

```
geom_siZe -region [get_window_area] -size -0.2 -fix_edges_to_region
```



Before Size operation

Shrink in a region with
-fix_edges_to_region true

Shrink in a region with
-fix_edges_to_region false

Related Information

Tcl Commands	geom_and geom_and_not geom_extent geom_not geom_or geom_xor
Menu Commands	*Create—Derived Layers*

geom_tiles_on_scratch_layer

```
geom_tiles_on_scratch_layer
     -scratch_layer i_scratchID
```

Returns the number of tiles on the given scratch layer.

Arguments

`-scratch_layer` *`i_scratchID`*

Specifies the scratch layer identified by the positive integer.

Value Returned

`i_count`	Is the number of tiles on the given scratch layer.
`-1`	The scratch layer is not in use or was not specified properly.

Example

The following example sets `numTilesOn3` with the number of tiles on scratch layer 3.

```
set numTilesOn3 [geom_tiles_on_scratch_layer -scratch_layer 3]
```

Related Information

Tcl Commands	geom_scratch_layers_in_use

geom_xor

```
geom_xor
     { {-lpp1 { s_layerlpp … }
       [ -lpp1_levels {i_startLevel [i_endLevel] } ]
       [ -set1 d_setObj ]
       [ -lpp1_net_names { s_netName… } | -lpp1_net_set d_setObj ] }
     | -input1_scratch_layer i_scratchID }
     { {-lpp2 { s_layerlpp … }
       [ -lpp2_levels {i_startLevel [i_endLevel] } ]
       [ -set2 d_setObj ]
       [ -lpp2_net_names { s_netName… } | -lpp12_net_set d_setObj ] }
     | -input2_scratch_layer i_scratchID }
     { { -output_lpp s_lpp [ -output_net_name s_netName ] [ -output_set
     d_setObj ] }
     | -output_scratch_layer i_scratchID }
     [ -no_sync ]
     [ -polygons ]
     [ -region { f_xlo f_ylo f_xhi f_yhi } ]
     [ -route_topology { core_ring | block_ring | stripe | cell_row_strap } ]
```

Generates new polygons wherever the shapes of one input group do not overlap shapes of another input group. Input groups can be a scratch layer, or one or more layer purposes, optionally limited to specific levels for nets in a set or a list, or shapes in a set. The generated shapes are added to a scratch layer, or to a layer purpose and, optionally, to a specific net. You can operate on a specific region or the entire design.

Arguments

-input1_scratch_layer *i_scratchID*

Uses a scratch layer from a previous Boolean operation for lpp1 shapes.

-input2_scratch_layer *i_scratchID*

Uses a scratch layer from a previous Boolean operation for lpp2 shapes.

-lpp1 **{***s_layerlpp…***}**

Specifies the first LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

-lpp1_levels **{***i_startLevel* [*i_endLevel*]**}**

Limits lpp1 shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is 0.
Default: all levels

-lpp1_net_names **{***s_netName…***}**

Limits lpp1 shapes to the named nets.

-lpp1_net_set *d_setObj*

Limits lpp1 shapes to the nets in the set.

-lpp2 **{***s_layerlpp…***}**

Specifies the second LPPs. For details on the LPP notation, refer to “Specifying Input Layer Purposes” on page 321.

-lpp2_levels **{***i_startLevel* [*i_endLevel*]**}**

Limits lpp2 shapes to those on the specified levels. If start and end levels are given, all levels in between are included. The top level is 0.
Default: all levels

-lpp2_net_names **{***s_netName…***}**

Limits lpp2 shapes to the named nets.

-lpp2_net_set *d_setObj*

Limits lpp2 shapes to the nets in the set.

-no_sync

By default, the OpenAccess database is immediately updated with the results of this operation. If this argument is given, the results will not be saved until the database is saved (write_db).

-output_lpp *s_lpp*

Specifies the output LPP for the generated shapes.

-output_net_name *s_netName*

Adds shapes to the named net. When this argument is given, the -no_sync argument is ignored. Shapes that are created by this command will be saved with the named net when the database is saved.

-output_scratch_layer *i_scratchID*

Adds generated shapes to a scratch layer identified by the positive integer. This scratch layer can be used as an input layer in subsequent Boolean operations using the -input1_scratch_layer or -input2_scratch_layer arguments for the current design. Scratch layers cannot be displayed but can be removed using geom_remove_scratch_layers.

-output_set *d_setObj* Adds generated shapes to the given set.

-polygons Generates output shapes as polygons, rather than individual rectangles. By default, individual rectangles are generated.

-region **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the area of the operation given the lower-left coordinate and the upper-right coordinate.

-route_topology *s_topologyName*

(Applies only if -output_net_name is given with the name of an existing power or ground net) Sets the routeTopology property for the output shapes as follows:

block_ring	BlockRingRouteTopology
cell_row_strap	StandardCellWireRouteTopology
core_ring	RingRouteTopology
stripe	StripeRouteTopology (this is the default)

-set1 *d_setObj* Limits lpp1 shapes to shapes in the set.

-set2 *d_setObj* Limits lpp2 shapes to shapes in the set.

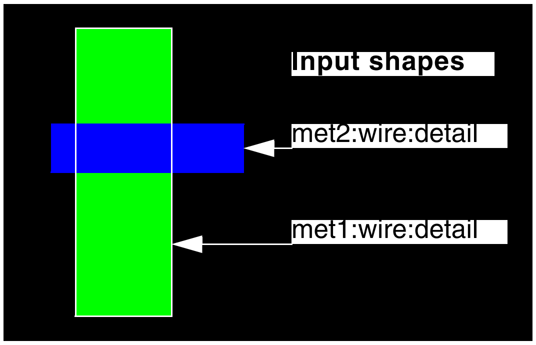
Value Returned

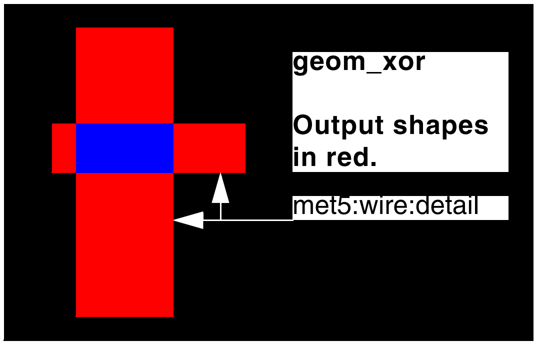
`i_count`	Is the total number of shapes created.

Example

The following example generates new shapes on the `met5:wire:detail` layer purpose from shapes on either the `met1` or `met2` layers that do not overlap.

```
geom_xor -lpp1 {met1 met2} -output_lpp {met5:wire:detail}
```





Related Information

Tcl Commands	geom_and geom_and_not geom_extent geom_not geom_or geom_size
Menu Commands	*Create—Derived Layers*

Design Configuration Commands

You use the Design Configuration commands to get, set, and change parameters for your design, and prepare data from third party tools.

The commands are presented in alphabetic order:

assign_term

```
assign_term
     -inst { s_instName | d_ctuObj }
     -net s_netName
     -term s_termName
```

Assigns a terminal of an instance to a net.

Arguments

`-inst { s_instName | d_ctuObj }`	
	Specifies the name of the instance or the object identifier for the instance.
`-net s_netName`	Specifies the name of the net.
`-term s_termName`	Specifies the name of the term.

Example

The following example assigns the `Q` term of instance `I1` to the `clkA` net.

```
assign_term -inst I1 -term Q -net clkA
```

Related Information

Tcl Commands	unassign_term

clean_nets

```
clean_nets
     [ -all | -net {s_netName…} | -set d_setObj ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
```

Prepares third party data by establishing centerline connectivity, and removing loops, dangles, and redundant route elements without changing the footprint of the metal. Works on the entire design, specified nets, or nets in the given set.

Arguments

`-all`	Cleans all nets in the design. This is the default.
`-exclude_net {s_netName…}`	Excludes the given nets from processing.
`-exclude_set d_setObj`	Excludes nets in the given set from processing.
`-exclude_type {[power][ground][clock]}`	Excludes one or more given types of nets from processing.
`-net {s_netName…}`	Cleans nets in the list.
`-set d_setObj`	Cleans nets in the given set.

Example

This command establishes centerline connectivity for the entire design and cleans nets without changing the footprint of the metal.

```
clean_nets
```

Related Information

Tcl Commands	pattern_route repair_net update_net_connectivity

collapse_shape_term

```
collapse_shape_term
     {-net s_netName | -net_id d_netObj | -set d_setObj}
```

Moves individual term shapes to a single term parent for a named net, a net with a given object identifier, or a set containing term shapes or nets.

This command can be used to create a single terminal for newly created power and ground nets, similar to the function provided by `db.one_shape_term_per_net` when existing power and ground nets are loaded using `read_db`.

Arguments

`-net` *`s_netName`*	Moves term shapes on the named net to a single term parent.	
`-net_id` *`d_netObj`*		
	Moves term shapes on the identified net to a single, primary term parent.	
`-set` *`d_setObj`*	The set may contain the following:	
	term shapes	Collapses term shapes for the same net to a single term parent. Any term shapes for the same net that are not in the set will not be included.
	nets	Collapses term shapes on each net to a single term parent.

Examples

The following example creates a single terminal from all VDD term shapes.

```
collapse_shape_term -net VDD
```

convert_routing_shapes

```
convert_routing_shapes
     -set d_setObj
     {-to_regular [ true | false ]} | {-to_special [ true | false ] [-to_trunk [
     true | false ]]}
     [ -persistent [ true | false ] ]
     [ -verbose [ true | false ] ]
```

Converts special (*shape term*) shapes to regular *route* shapes to permit editing, and converts regular route shapes that were created in the current session to special shape terms. You can convert individual shapes (segments and vias), entire routes, or nets. The converted shapes will automatically revert to their original state before saving, unless you make them persistent.

Special nets from DEF are loaded as shape terms, which are not editable and do not participate in most routing-related operations. Special net route segments and vias are typically on the `wire:pin` and `via:pin` purposes, respectively. Regular route segments and vias are on the `wire:detail` and `via:detail` purposes, respectively. When routing is converted, the purposes are changed correspondingly. If the original purpose of a shape is not one of these purposes, its purpose will not be changed. For example, the purpose for shapes on the `blockage` purpose will not be changed.

Converting Special to Regular

Since pre-existing special nets are not editable, if you need to make any changes to them, such as adding tie-offs to power connections, you must convert them to regular route shapes to make them editable. To keep the regular route shapes when saving, you must make the conversion persistent, otherwise, the converted routing shapes will automatically revert to shape terms before saving.

Converting Regular to Special

Only new routing created during the current session can be converted to special shape terms. Routing data read in from an OpenAccess database cannot be converted. In addition, Space-based Router and Chip Optimizer cannot automatically revert from special to regular in *One Shape Term Per Net* mode. To convert from regular to special, then revert back to regular, you must use one of the following methods:

- With *Multiple Shape Terms Per Net* mode (`db.one_shape_term_per_net` is `false`), load the design, convert to special, then auto revert back to regular on save.

- With *One Shape Term Per Net* mode (`db.one_shape_term_per_net` is `true`), load the design, convert to special, then manually convert back to regular persistently before saving.

Arguments

`-persistent [ true | false ]`

When `true`, the conversion of shapes will persist when saved.

Default: Conversion will automatically revert before saving, except when converting from regular to special in One Shape Term per Net mode.

`-set` *`d_setObj`* Converts shapes, routes and nets in the given set. If the set is not given, the selected set is used.

`-to_regular [ true | false ]`

Converts shape terms to route shapes.

`-to_special [ true | false ]`

Converts route shapes created in the session to special shape terms.

`-to_trunk [ true | false ]`

Marks special (*shape term*) shapes as trunks.

`-verbose [ true | false ]`

When `true`, outputs to the transcript area information on each shape, route, and net that is converted, plus run times.

Default: `false`

Example

The following commands change `netA`, that was read in from DEF as a special net, from shape terms to editable regular routes.

```
set NET_A [find_net -name netA -ignore_case true -no_wildcard false ]
convert_routing_shapes -set $NET_A -to_regular true
```

create_derived_vias

```
create_derived_vias
     [ -cut_layers {[existing] [{s_cutLayerName…}]} ]
     [ -replace [ true | false | derivedVias | stdVias | customVias | all ] ]
     [ -include_via_variants [ true | false ] |
      -discard_via_variants [ true | false ] ]
     [ -single_cut [ true | false ] ]
     [ -non_pref [ true | false ] ]
     [ -use_taper_rule [ true | false ] ]
     [ -top_of_stack [ true | false ] ]
     [ -asymmetric_tos [ true | false ] ]
     [ -align_tos [ true | false ] ]
     [ -max_cuts [ true | false ] ]
     [ -min_num_cuts [ true | false ] ]
     [ -align_extensions [ true | false ] ]
     [ -large_cut_space [ true | false ] ]
     [ -std_via_def [ true | false ] ]
     [ [ -force_std_via_def [ true | false ] | -max_std_via_def [ true | false ] ]
      [ -keep_extensions [ true | false ] ] ]
     [ -use_extension_entry {i_tableColIndex…} ]
     [ -offset_origin [ true | false ] | -offset_origin_center_cut
     [ true | false ] ]
     [ -pre_clear [ true | false ] ]
     [ [ -cut_class {f_x f_y} [ -cut_class_num_cuts i_numCut] ]
     | [ -cut_class_names {s_cutClassName…} [ -cut_class_rotate [ true| false ] ]
     ]
     [ -cut_columns i_colCount ]
     [ -cut_rows i_rowCount ]
     [ -group s_groupName [ -taper_spec s_taperSpecName ]
      [ -count [ true | false ] ] ]
     [ -hardness {hard | soft} ]
     [ -report [ true | false ] ]
     [ -report_all [ true | false ] ]
     [ -right_way {top | bottom | both | none} ]
     [ -silent [ true | false ] ]
     [ -warn [ true | false ] ]
     [ -use_cut_class [ true | false ] ]
     [ -right_way [ top | bottom | both | none ] ]
     [ -use_allowed_cut_classes [ true | false ] ]
```

Creates vias that are design rule compliant for the standard vias specified in the `validRoutingVias` list for each routespec that is used in the current cellview. The derived vias are also referred to as *parameterized* vias.

Most of the Boolean arguments determine which via types will be created and default to `true`. No command arguments are necessary to create these via variations. To eliminate some variations, specify one or more arguments with a `false` setting, and/or use

`-use_extension_entry`. To create vias from cutClass settings, you must specify a cut class argument (`-cut_class` or `-cut_class_names`).

The types of vias that are created are dependent on whether certain constraints are set or can be derived from the default oaStdViaDef parameters. These constraints are listed in Table 7-1 on page 362.

Table 7-1 Constraints for create_derived_vias

`minWidth`
`minSpacing`
`minSameNetSpacing`
`oaMinViaSpacing`
`minLargeViaArraySpacing`
`minLargeViaArrayCutSpacing`
`minLargeViaArrayWidth`
`minAdjacentViaSpacing`
`minNeighborViaSpacing`
`minParallelViaSpacing`
`minParallelWithinViaSpacing`
`minExtension`
`minDualExtension`
`minEdgeLength`
`minEdgeAdjacentLength`
`minEdgeMaxCount`
`validRoutingLayers`
`validRoutingVias`
`minArea`
`minRectArea`
`minAreaEdgeLength`
`minNumCut`
`oaPreferredRoutingDirection`
`preferredExtensionDirection`
`inlineViaPreferred`
`preferredViaOrigin`
`cutClass`

The constraints specify cut dimensions, spacings, and metal enclosures. They also specify metal edge and area for top-of-stack vias, and preferred routing directions. Only vias that are design rule compliant will be created. If there are dual extensions, a *preferred* via is created with the greater extent in the preferred routing direction of the corresponding metal layer. By default, a *non-preferred* via is also created with the greater extension oriented wrong-way.

The new vias are added to the `extendedValidRoutingVias` constraint, making them available for use by the router and other Space-based Router and Chip Optimizer functions. The new vias are given the name of the existing standard via appended with `_derived_x` where `x` is a unique number. If the `extendedValidRoutingVias` constraint did not previously exist, it is set by this command and will include the new vias. The `validRoutingVias` constraint values can also be included in the `extendedValidRoutingVias` constraint, depending on the `-replace` and -`hardness` argument settings.

Note: Although you cannot directly set `extendedValidRoutingVias`, you can get the current value using:

```
get_constraint -constraint extendedValidRoutingVias -group s_groupName
```

You can also unset `extendedValidRoutingVias` using:

```
unset_constraint -constraint extendedValidRoutingVias -group s_groupName
```

If a standard via definition did not exist for a layer pair, then the technology database will be updated with the newly created standard via definition (oaStdViaDef). All new derived vias (oaStdVia) are saved to the design database. If you load a design that was saved with derived vias, you must re-issue the `create_derived_vias` command to set the `extendedValidRoutingVias` constraint and enable the derived vias.

Arguments

`-align_extensions [ true | false ]`

Creates vias with aligned top and bottom metal extensions.

Default: `true`

`-align_tos [ true | false ]`

Creates top-of-stack vias with aligned top and bottom metal extensions. Refer to Figure 7-1 on page 372 for an example.

Default: `false`

`-asymmetric_tos [ true | false ]`

Creates top-of-stack vias with asymmetric metal extensions. Refer to Figure 7-1 on page 372 for an example.

Default: `true`

`-count [ true | false ]`

Reports the number of extended valid vias per cut layer for a given constraint group, then quits. No vias are created.

Default: `false`

`-cut_class` **{***`f_x f_y`***}**

Creates vias with the specified directional cut class dimensions {*`f_x f_y`*}.

`-cut_class_names` **{***`s_cutClassName…`***}**

Creates vias with dimensions from the `cutClass` constraint with the given names. By default, both cut orientations (x, y) and (y, x) are created.

`-cut_class_num_cuts` *`i_numCut`*

Specifies the equivalent number of cuts for the given directional cut class. This value is ignored if the given cut class dimensions match an existing cutClass constraint.

Default: 1

`-cut_class_rotate [ true | false ]`

(Applies only with `-cut_class_names`) When `true`, creates both cut orientations (x, y) and (y, x) for non-square cut classes. Set this to `false` to create only the {x, y} orientation given by the cutClass names.

Default: `true`

`-cut_columns` *`i_colCount`*

Specifies the number of cut columns.

Default: 1

`-cut_layers {[existing][`**{***`s_cutLayerName…`***}**`]}`

When only the list is given, creates vias only for the cut layers in the list.
When only `existing` is given, creates vias for the same cut layers as the existing standard vias in `validRoutingVias`.
When `existing` is given with a list of cut layers, creates vias only for the cut layers in the list that also have existing standard vias in `validRoutingVias`.

Default: Creates vias for all cut layers

`-cut_rows` *`i_rowCount`* Specifies the number of cut rows.

Default: 1

`-discard_via_variants [ true | false ]`

When processing `validRoutingVias`, discard (`true`) or keep (`false`/default) existing via variants.

`-force_std_via_def [ true | false ]`

If `true`, derives vias using parameter values from the applicable oaStdViaDef instead of using constraint values for the cut layer. No vias will be created using via parameters that violate existing via rules.

Default: (`false`) oaStdViaDef settings are only used when required constraints are not defined.

`-group` *`s_groupName`* Specifies the name of the constraint group, or rule spec, for which to create the derived vias, based on the rules for the group.

Default: All constraint groups of type `net`, `route`, and `default`

`-hardness {hard | soft}`

Determines whether the resulting `extendedValidRoutingVias` with the new derived vias is a hard or soft (preferred) constraint and whether hard or soft `validRoutingVias` are considered when creating the `extendedValidRoutingVias`.

Default: Both hard and soft `extendedValidRoutingVias` constraints are created or updated to include the new vias.

`-include_via_variants [ true | false ]`

When processing `validRoutingVias`, derive vias from via variants in `validRoutingVias` (`true`), or keep the existing via variants in `validRoutingVias` (`false`/ default).

`-keep_extensions [ true | false ]`

(Applies only with `-force_std_via_def` or `-max_std_via_def`) If `true`, keep extension orientation of the applicable oaStdViaDef.

Default: `false`

`-large_cut_space [ true | false ]`

Creates multi-cut vias with large cut spacing.

Default: `true`

`-max_cuts [ true | false ]`

Creates vias with the maximum number of cuts for the widths of each layer pair.

Default: `true`

`-max_std_via_def [ true | false ]`

When `true`, use the maximum of via parameters from the applicable oaStdViaDef and existing via rules.

Default: `false`

`-min_num_cuts [ true | false ]`

When `true` **and** -`cut_rows` and -`cut_columns` are not given (or are set to 1), creates multi-cut vias based on the `minNumCut` rules.

Default: `true`

`-non_pref [ true | false ]`

When `true`, creates additional vias for the non-preferred (wrong-way) routing direction. When `false`, wrong-way extension variations are not created.

Default: `true`

`-offset_origin [ true | false ]`

(Applies only to `1xN` and `Nx1` cut vias with `N`>1) When `true`, the via origin is offset such that wires connect at the edge of via metal. When `false`, the via origin is placed at the center of the cuts. Refer to Figure 7-2 on page 372 to see a graphical representation of each choice.

Default: `false`

`-offset_origin_center_cut [ true | false ]`

(Applies only to `1xN` and `Nx1` cut vias with `N`>1) When `true`, the via origin is offset such that wire centers connect at the center of an end cut. If `false`, the via origin is placed at the center of all cuts. Refer to Figure 7-2 on page 372 to see a graphical representation of each choice.

Default: `false`

`-pre_clear [ true | false ]`

If `true`, clears all derived vias before creating new vias for the applicable constraint groups.

Default: `false`

`-replace [ true|false | derivedVias | stdVias | customVias | all ]`

Chooses whether the new derived vias will replace existing vias or will be added to the existing derived vias.

`all` Replaces all existing vias with the new derived vias.

`customVias` Replaces custom vias with the new derived vias.

`derivedVias` Replaces existing derived vias with the new derived vias. Has the same function as `true`.

`false` Adds new derived vias to the existing derived vias. This is the default.

`stdVias` Replaces standard vias, including existing derived vias, with the new derived vias.

`true` Replaces existing derived vias with the new derived vias. Has the same function as `derivedVias`.

`-report [ true | false ]`

Reports the via parameters for existing derived vias. No vias are created when this argument is given.

Default: `false`

`-report_all [ true | false ]`

If `true`, reports detail information for all existing standard and custom vias, then quits. No vias are created.

Default: `false`

`-right_way` *`s_orient`*

Specifies the orientation for cut pattern and extensions when creating vias. Refer to Figure 7-3 on page 373 for graphical examples of the available choices.

`top`	Creates vias with cut pattern and extensions oriented only in the preferred routing direction of the top metal layer.
`bottom`	Creates vias with cut pattern and extensions oriented only in the preferred routing direction of the bottom metal layer.
`both`	Creates vias with cut pattern and extensions oriented only in the preferred routing direction of each metal layer.
`none`	Creates vias with all possible cut patterns and extension orientations. This is the default.

`-silent [ true | false ]`

If `true`, most output messages are suppressed.

Default: `false`

`-single_cut [ true | false ]`

Creates basic single-cut vias if `-cut_rows` and `-cut_columns` are not specified.

Default: `true`

`-std_via_def [ true | false ]`

If `minWidth`, `minSpacing`, or one of `minExtension` or `minDualExtension` is not defined for the cut layer, this option allows these constraints to be derived from the default oaStdViaDef. If set `false` and `minSpacing` and `minWidth` constraints are not specified for the cut layer, no derived vias are created.

Default: `true`

`-taper_spec` *`s_taperSpecName`*

(Applies only with `-group`) Specifies the name of the constraint group, or rule spec, to use as a taper rule. Allows you to temporarily substitute/override a group's taper rule when creating vias for taper rules.

`-top_of_stack [ true | false ]`

Creates top-of-stack vias with symmetric extensions. Refer to Figure 7-1 on page 372 for an example.

Default: `true`

`-use_extension_entry` **{***`i_tableColIndex`*…**}**

Limits the creation of derived vias to the metal extensions from a OneDDualArrayTblValue whose column index is in this list. By default, all entries in the table are used. Entry values are non-negative integers from 0 to *N*-1, where *N* is the number of extension pair entries in the minDualExtension table.

For example,

```
-OneDDualArrayTblValue {0 2 0.05 0.20 0.10 0.10}
```

specifies two extension pairs {0.05 0.20} and {0.10 0.10}.

If you use `-use_extension_entry {0}`, then only {0.05 0.20} extensions will be used to create the derived vias.

`-use_taper_rule [ true | false ]`

Creates vias for taper rules.

Default: `true`

`-warn [ true | false ]` Specifies whether warnings are output.

Default: `true`

`-use_cut_class [ true | false ]`

Creates vias with dimensions from the cut class constraints for the smallest number of equivalent cuts.

Default: `true`

`-right_way [ top | bottom | both | none ]`

Creates vias with cut pattern and extensions oriented only for right-way metal on top, bottom, both, or none.

Default: `none`

`-use_allowed_cut_classes [ true | false ]`

Creates vias using the allowed cut classes in the `allowedCutClass` constraint.

Default: value of `db.cdv_use_allowed_cut_classes`, if defined, else `false`.

Example

The following example sets the `minWidth` (which is not typically set for vias in the LEF) and `minDualExtension` constraints for creating parameterized `Via1` vias. In this case, the `minSpacing` rule for `Via1` was set in the LEF.

```
set_layerpair_constraint -constraint minDualExtension -DualValue { 0.00 0.05} \
-layer1 Metal1 -layer2 Via1
set_layer_constraint -layer Via1 -constraint minWidth -hardness hard -Value 0.1
```

`dump_ctu_constraints -constraint` *`s_name`* can be used to check these settings, or `get_layerpair_constraint` and `get_layer_constraint` for the individual settings.

With these values, Space-based Router and Chip Optimizer creates parameterized vias when the following command is issued:

```
create_derived_vias
```

Space-based Router and Chip Optimizer creates derived vias based on the available or derived constraints and outputs messages to the Transcript area to indicate the vias that are generated. For example,

```
Creating preferred via (offset=0,0) (rows: 1 cols: 1 cutX: 600 cutY: 600 spcX: 600
spcY: 600) - Metal1 ext: N:200 S:200 E:200 W:200 <==> Metal2 ext: N:200 S:200 E:200
W:200
```

where

- **`offset`** is the origin offset (x,y)
- **`rows`** is the number of cut rows
- **`cols`** is the number of cut columns
- **`cutX`** is the horizontal cut dimension
- **`cutY`** is the vertical cut dimension
- **`spcX`** is the horizontal spacing between adjacent cuts
- **`spcY`** is the vertical spacing between adjacent cuts
- **ext** are the extensions for the given metal layer in the (N)orth, (S)outh, (E)ast and (W)est directions

Use the following command to output the list of valid routing vias, including derived vias:

```
get_constraint -constraint extendedValidRoutingVias
```

Additional special 4-cut vias for wide nets (assigned to the `WIDE` constraint group) can be created by ignoring the `minNumCut` rule using the following commands:

```
create_derived_vias -group WIDE -cut_rows 1 -cut_columns 4
create_derived_vias -group WIDE -cut_rows 4 -cut_columns 1
create_derived_vias -group WIDE -cut_rows 2 -cut_columns 2
```

Derived Vias from MinDualExtension OneDDualArrayTblValue

If both metal layers for a via have `OneDDualArrayTblValue` `minDualExtension` extension values, then the extension pairs of one metal layer will only be matched with the corresponding table column for the other metal layer.

In the following example, both M2 and M3 have three extension pairs for a width of 0.0 and three extension pairs for a width of 0.7.

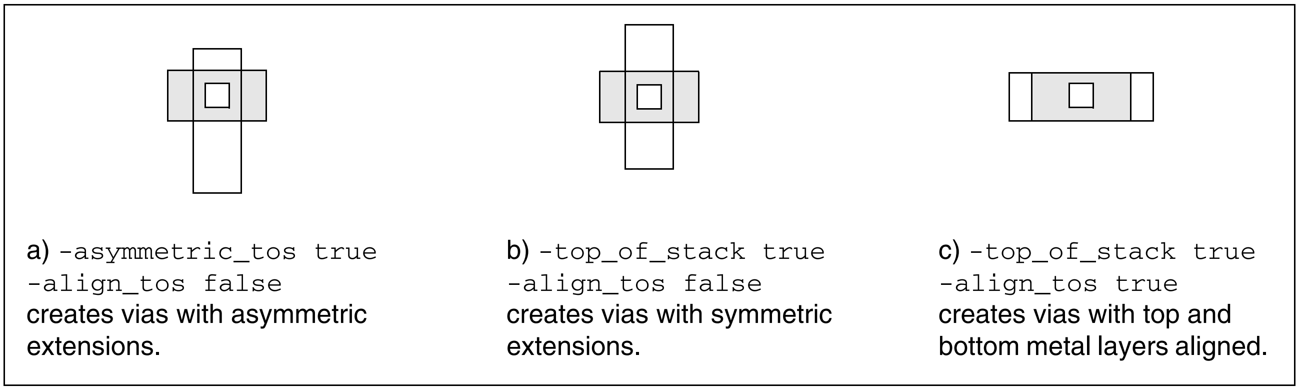
```
set_layerpair_constraint -constraint minDualExtension -layer1 M2 -layer2 VIA2
-OneDDualArrayTblValue {0.0 3 0.01 0.10 0.01 0.16 0.10 0.10 0.7 3 0.02 0.20 0.02
0.32 0.20 0.20}
set_layerpair_constraint -constraint minDualExtension -layer1 M3 -layer2 VIA2
-OneDDualArrayTblValue {0.0 3 0.02 0.10 0.18 0.18 0.10 0.10 0.7 3 0.04 0.20 0.36
0.36 0.20 0.20}
```

The `VIA2` derived vias will consist of three sets of vias for each given width, no additional pair combinations will be created. It is assumed that the first extension pair of M2 corresponds to the first extension pair on M3. The first extension pair on M2 will not be combined with the second extension pair on M3 to form a via.

For width 0.0,

```
Extensions in Set 0 = M2 (0.01 0.10) - M3(0.02 0.10)
Extensions in Set 1 = M2 (0.01 0.16) - M3(0.18 0.18)
Extensions in Set 2 = M2 (0.01 0.10) - M3(0.10 0.10)
```

Figure 7-1 Derived Top-of-Stack Vias Examples

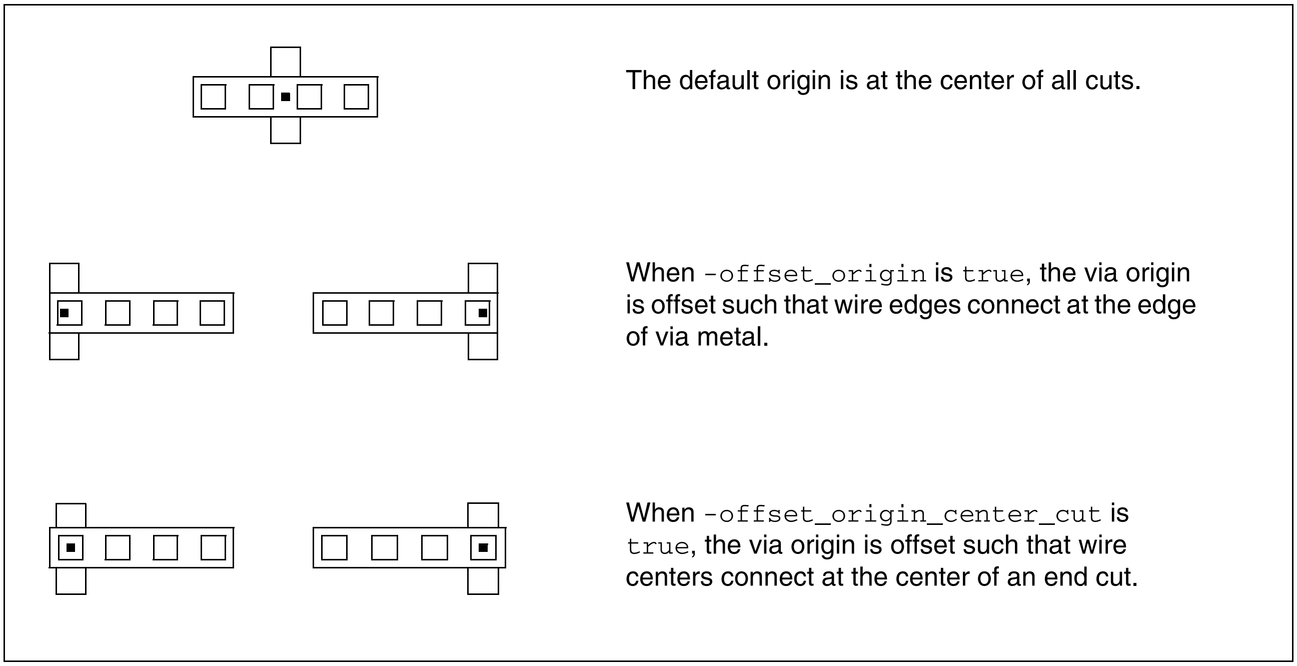


a) `-asymmetric_tos true -align_tos false` creates vias with asymmetric extensions.

b) `-top_of_stack true -align_tos false` creates vias with symmetric extensions.

c) `-top_of_stack true -align_tos true` creates vias with top and bottom metal layers aligned.

Figure 7-2 Origin Variations for create_derived_vias

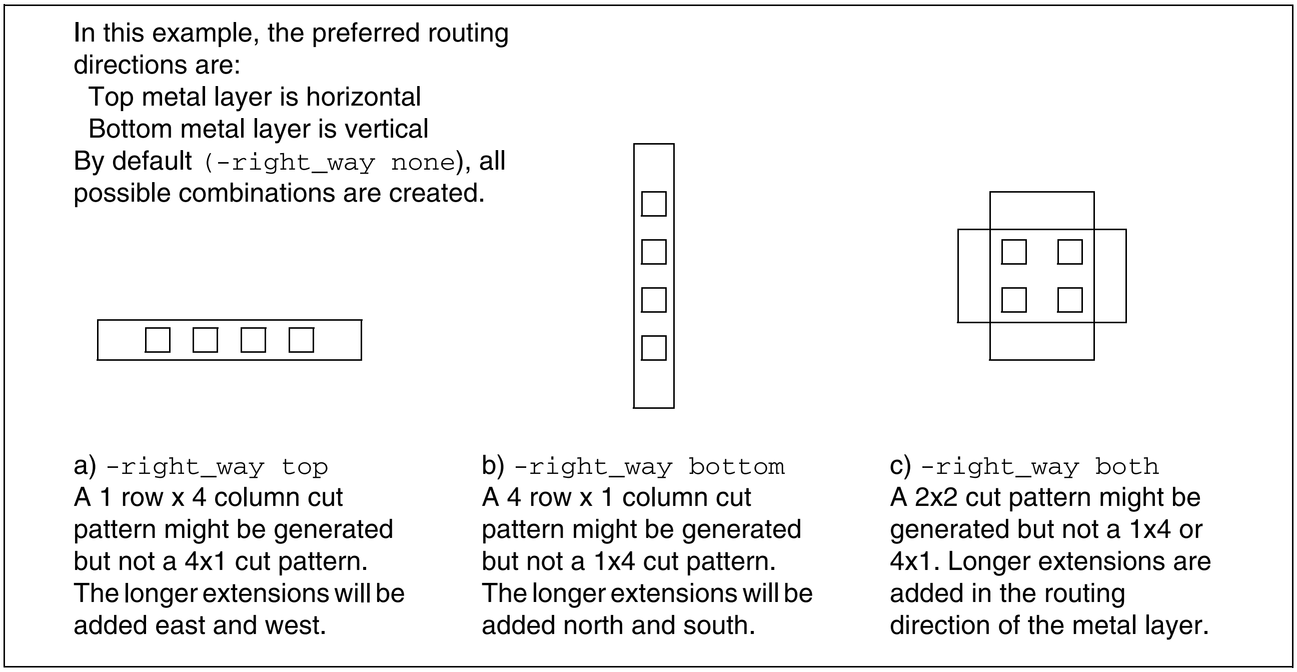


The default origin is at the center of all cuts.

When `-offset_origin` is `true`, the via origin is offset such that wire edges connect at the edge of via metal.

When `-offset_origin_center_cut` is `true`, the via origin is offset such that wire centers connect at the center of an end cut.

Figure 7-3 Right-way Variations for create_derived_vias



In this example, the preferred routing directions are:
  Top metal layer is horizontal
  Bottom metal layer is vertical
By default (`-right_way none`), all possible combinations are created.

a) `-right_way top`
A 1 row x 4 column cut pattern might be generated but not a 4x1 cut pattern. The longer extensions will be added east and west.

b) `-right_way bottom`
A 4 row x 1 column cut pattern might be generated but not a 1x4 cut pattern. The longer extensions will be added north and south.

c) `-right_way both`
A 2x2 cut pattern might be generated but not a 1x4 or 4x1. Longer extensions are added in the routing direction of the metal layer.

Related Information

Tcl Command	create_via_variant dump_ctu_constraints get_constraint get_layer_constraint get_layerpair_constraint set_layer_constraint set_layerpair_constraint

create_no_pref_dir_region

```
create_no_pref_dir_region
```

(Virtuoso Routing IDE and Space-based Router only) Identifies regions in the design that contain a bank of tightly spaced pins on layers whose preferred routing direction is opposite to the preferred direction for the most efficient routing to or from the pins. Each *no preferred direction* region is marked with a rectangular boundary and the router is permitted to wrong-way route as needed within those regions.

To remove all *no preferred direction* regions, use `delete_no_pref_dir_region`.

Arguments

None

Related Information

Tcl Commands	delete_no_pref_dir_region

create_pr_boundary

```
create_pr_boundary
     -region {f_xlo f_ylo f_xhi f_yhi}
```

Sets or replaces the PR boundary for the active design. The PR boundary is viewable in the workspace and its visibility is controlled by the `area_boundary:PR boundary` entry in the Object section of the Layer Object Display Panel.

Arguments

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the new PR boundary points.

Example

The following example sets the area in the workspace to the new PR boundary.

```
create_pr_boundary -region [get_window_area]
```

create_preferred_direction_region

```
create_preferred_direction_region
     -name s_regionName
     -group s_groupName
     {-region {f_xlo f_ylo f_xhi f_yhi} |
      -points {f_x1 f_y1 … f_xn f_yn} ]
```

Creates and names a rectangular or polygonal preferred direction region. This is used to override the global settings for the preferred routing direction in the region. The region boundary is viewable in the workspace and its visibility is controlled by the `area_boundary:pref dir area` entry in the Object section of the Layer Object Display Panel.

Arguments

`-group` *`s_groupName`*

Specifies the name of the constraint group to assign this region to. When the `oaPreferredRoutingDirection` constraint is included in the constraint group, the preferred routing directions for layers given by the constraint will override the global settings in the region given by this command.

`-name` *`s_regionName`*

Specifies the name for the preferred direction region.

`-points` **{***`f_x1 f_y1…f_xn f_yn`***}**

Specifies the boundary points for a polygonal area.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for a rectangular area.

Example

The following commands sets layers with preferred horizontal directions, layers with preferred vertical directions and the region that these constraints should apply to.

```
create_constraint_group -name grpA -type route
set_constraint -constraint oaPreferredRoutingDirection \
 -LayerArrayValue {M1 M3 M5 M6} -group grpA -StringAsIntValue horzPrefROutingDir
set_constraint -constraint oaPreferredRoutingDirection \
 -LayerArrayValue (M2 M4} -group grpA -StringAsIntValue vertPrefRoutingDir
create_preferred_direction_region -name regionA -group grpA -region {200 200 400
450}
```

Related Information

Tcl Command	delete_preferred_direction_region set_constraint

create_soft_fence

```
create_soft_fence
     -name s_regionName
     {-region {f_xlo f_ylo f_xhi f_yhi}
     | -points {f_x1 f_y1 … f_xn f_yn}}
```

Creates a soft fence for an area (rectangle or polygon).

General rules for routing with fences are as follows:

- If all terminals of a net are outside of fence areas, all wiring for the net must be outside of fence areas.
- If all terminals of a net are inside of a fence area, all wiring for the net must be inside of the fence area.
- If terminals of a net are inside and outside of a fence area, then the wiring for the net can cross the fence to make the connection.

The soft fence boundary is viewable in the workspace and its visibility is controlled by the `area_boundary:soft fence` entry in the Object section of the Layer Object Display Panel.

Arguments

`-name` *`s_regionName`* Specifies the name for the fence.

`-points {`*`f_x1 f_y1…f_xn f_yn`*`}`

Specifies the boundary points for a polygonal fence.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the boundary points for a rectangular fence.

Example

The following example creates a rectangular fence that covers the current workspace area.

```
create_soft_fence -name fence1 -region [get_window_area]
```

Related Information

Tcl Command	check_route_quality delete_soft_fence

delete_no_pref_dir_region

```
delete_no_pref_dir_region
```

(Virtuoso Routing IDE and Space-based Router only) Removes all *no preferred direction* regions from the design.

Arguments

None

Related Information

Tcl Commands	create_no_pref_dir_region

delete_preferred_direction_region

```
delete_preferred_direction_region
     {-name s_regionName | -all}
```

Removes the named preferred direction region or all preferred direction regions.

Arguments

`-all`	Removes all existing preferred direction regions in the current cellview.
`-name` *`s_regionName`*	Specifies the name for the preferred direction region.

Example

The following removes the `regionA` preferred direction region.

```
delete_preferred_direction_region -name regionA
```

Related Information

Tcl Command	create_preferred_direction_region

delete_soft_fence

```
delete_soft_fence
     {-name s_regionName | -all}
```

Removes an existing fence or all soft fences.

Arguments

`-all`	Removes all soft fences.
`-name` *`s_regionName`*	Specifies the name of the fence.

Example

The following example deletes a fence named `fence1`.

```
delete_soft_fence -name fence1
```

Related Information

Tcl Command	check_route_quality create_soft_fence

get_current_design

```
get_current_design
```

Returns the current CTU design object.

Arguments

None

Example

```
set design [get_current_design]
```

It returns the current CTU design object.

extract_net_connectivity

```
extract_net_connectivity
     {-all
     | -set d_setObj
     | -window_id i_windowID
     | -cells {s_libCellView…}}
     [ -apply_pin_style_to_top_cell [ true | false ] ]
     [ -clock_names {s_netName…}]
     [ -extract_poly [ true | false ] ]
     [ -flatten_all_lpps [ true | false ] ]
     [ -flatten_blockages [ true | false ] ]
     [ -flatten_to_depth {1|2}]
     [ -ground_names {s_netName…}]
     [ -ignore_cells {s_libCellView…}]
     [ -layer_precedence {s_layerName…}]
     [ -net_name_file s_lnnFileName ]
     [ -net_name_prefix s_prefix]]
     [ -net_name_property s_property]
     [ -pin_style {labeled_shapes | connected_shapes | whole_net
     | whole_net_all_layers}]
     [ -power_names {s_netName…}]
     [ -resolve_shorts [ true | false ] ]
     [ -routify [ true | false ] ]
     [ -save ]
     [ -save_lib s_libName ]
     [ -save_view s_viewName | -view s_viewName ]
     [ -stop_level i_level]
     [ -text_layer {shape_layer | s_layerName}]
     [ -text_purpose s_lpp]
     [ -use_layout_text [ true | false ] ]
     [ -verbose ]
```

Extracts connectivity from a layout without connectivity or with partial connectivity, such as data imported from GDSII Stream format. Adding connectivity to the data will allow it to be routed and/or optimized.

Operates on objects with the following characteristics:

- The object must be a rectangle, polygon, route wire, or route via
- Rectangles, polygons and route wires must be on a layer of material type metal, poly, or cut, and must be on one of the following purposes: wire:detail, via:detail, wire:pin, via:pin, or a user-defined purpose that is mapped to one of these purposes.

These objects are referred to as *interconnect objects*. In addition, the extractor can use text objects (`-use_layout_text true`), which must have purpose `text`.

Any interconnect object that is not part of a net is assigned to a net by the extractor by recursively identifying any other objects that are physically connected to that object. If any object in this connected set is already part of a net, then all unassigned objects in the set are assigned to that net. If none of the objects in the connected set is already assigned to a net, then a new net will be created.

Nets created by the extractor can be assigned a name by the extractor from one of the following sources:

- The value of a property attached to an object in the net (`-net_name_property`)
- The value of a text object in the layout that labels one of the objects in the net (-`use_layout_text`)
- A user-given name prefix with a number appended to ensure unique naming (-`net_name_prefix`)

Arguments

`-all` Extracts all of the nets in the cellview.

`-apply_pin_style_to_top_cell [ true | false ]`

Specifies whether the `-pin_style` setting should be applied to the top cell.

For example, if you are routing the top cell and you want all of the relevant shapes at level one of its hierarchy to be possible routing targets, specify `-pin_style whole_net`. In addition, if you do not want all of the top cell shapes to be made into pin shapes, specify `-apply_pin_style_to_top_cell false`.

Default: `true`

`-cells {`*`s_libCellView…`*`}`

Performs extraction on cells in the given list. Cells are specified with the format *`lib/cell/view`*. Wildcards are accepted. Cells can be excluded using `-ignore_cells`. You do not need to load a top cell to use this argument.

For example, `-cells {stdLib/*/layout}` will extract all cells in the `stdLib` library with `layout` view.

`-clock_names {`*`s_netName…`*`}`

Recognizes names in the list as clock nets in the design. If the extractor creates a net with one of these names, the signal type of the net will be set to "`clock`".

`-extract_poly [ true | false ]`

If `true`, extracts shapes on the poly layer.

Default: `true`

`-flatten_all_lpps`

Applies only when `-flatten_to_depth` is given. If `true`, shapes on all layer purposes are copied up the hierarchy during the flattening process. If `false`, only *interconnect objects* are copied. In some cases, specifying false can result in reduced memory usage without affecting the extraction or subsequent routing and optimization results (for example, if subcells contain well, implant or non-mask layers that are not related to routing). Default: `true`

`-flatten_blockages [ true | false ]`

Applies only when -`flatten_to_depth` is given. If `true`, blockage shapes are copied during flattening. If `false`, blockage shapes are not copied. Default: `true`

`-flatten_to_depth {1|2}`

Removes hierarchy from the design. If set to 1, layout data from all of the instances are copied into the extracted cell and the instances are removed from the extracted cell. If set to `2`, all hierarchy is removed from the immediate subcells (level 1) of the extracted cell. By default, no flattening occurs.

`-ground_names {`*`s_netName…`*`}`

Recognizes names in the list as ground nets in the design. If the extractor creates a net with one of these names, the signal type of the net will be set to "`ground`".

`-ignore_cells {`*`s_libCellView…`*`}`

Ignores layout in the given cells. Cells are specified with the format *`lib/cell/view`*. Wildcards are accepted. To include cells, use `-cells`.

`-layer_precedence {`*`s_layerName…`*`}`

Specifies an ordered list of layers which must be valid technology layer names. Used with `-text_layer` to specify the layer search order for any interconnect object whose boundary contains the text object's origin. The first interconnect shape found will have its net named using the text value. This argument must **not** be used with `-text_layer shape_layer`.

`-net_name_file` *`s_lnnFileName`*

Specifies the name of the Mentor Graphics® Calibre® Layout Netlist Names (LNN) file to use for net name mapping.

For more information on using this argument, refer to "Using -net_name_property with Calibre Layout Netlist Names Files" on page 390.

`-net_name_prefix` *`s_prefix`*

Specifies a prefix string for names of nets created by the extractor that are not named with a label from text or property. The extractor will create a unique name for each net by appending a number at the end of the prefix.

For example, if you specify a prefix of "`abc`", the extractor will assign names such as "`abc1`" and "`abc234`". The default prefix string is "`extract_n_`".

`-net_name_property` *`s_property`*

Names extracted nets using the value of this property attached to at least one of the net shapes.

For more information on using this argument, refer to "Using -net_name_property with Calibre Layout Netlist Names Files" on page 390.

`-pin_style` *`s_string`*

Controls which shapes become pin shapes.

`connected_shapes`	Marked shapes and those that are recursively connected become pin shapes.
`labeled_shapes`	Only shapes marked by text or property become pin shapes. This is the default.

`whole_net` All of the shapes in the net become pin shapes.

`whole_net_all_layers`

All of the shapes in the net, regardless of layer, become pin shapes.

`-power_names {`*`s_netName…`*`}`

Recognizes names in the list as power nets in the design. If the extractor creates a net with one of these names, the signal type of the net will be set to "`power`".

`-resolve_shorts [ true | false ]`

If `true`, merges nets that are shorted.

Default: `false`

`-routify [ true | false ]`

When `true`, organizes the route segments and vias into *routes* and creates the necessary route objects in preparation for routing.

Default: `false`

`-save` Saves cell views after extraction, overwriting current views. By default, no save occurs, except when `-save`, `-save_lib`, `-save_view` or `-view` is given.

`-save_lib` *`s_libName`* Saves the layout with the extracted nets to the given library. By default, no save occurs, except when `-save`, `-save_lib`, `-save_view` or `-view` is given.

`-save_view` *`s_viewName`* Saves the layout with the extracted nets to the given view. Has the same function as `-view`. Both arguments cannot be given in the same command. By default, no save occurs, except when `-save`, `-save_lib`, `-save_view` or `-view` is given.

`-set` *`d_setObj`* Extracts only the nets that own shapes in the given set.

`-stop_level` *`i_level`* Ignores all layout in hierarchy levels below the given level. By default, all levels are extracted.

`-text_layer {shape_layer |` *`s_layerName`*`}`

Specifies the layers on which the extractor looks for text objects to name nets. Valid only with `-use_layout_text true`.

`shape_layer`	The extractor looks on all routing layers for text. For any text object found, the extractor searches the text object's layer for any interconnect object whose boundary contains the text origin. If such a shape is found, that shape's net will be named with the text value.
`s_layerName`	Only the given layer will be searched for text objects. For any text object found, all routing layers are searched for an interconnect object whose boundary contains the text origin. The search order of the layers can be specified using `-layer_precedence`.

`-text_purpose` *`s_lpp`*

(Applies only when `-use_layout_text` and/or `-text_layer` is given) Specifies the name of the layer purpose to use to search for text objects in the layout.

Default: `text`

`-use_layout_text [ true | false ]`

Names extracted nets using text objects in the layout.

For a text object to name a net, the following criteria must be satisfied:

- The text object must be on an appropriate layer, given by `-text_layer` and be on the `text` purpose.
- The text object must lie in or on the boundary of a shape (rectangle, polygon, or wire segment) in the net.

Nets not named with text or properties are named using a prefix and number (see `-net_name_prefix)`.

`-verbose [ true | false ]`

	If set to `true`, outputs information about each extracted cell to the Transcript area. Default: `false`
`-view` *`s_viewName`*	Saves layout with extracted nets to the given view. By default, no save occurs, except when `-save`, `-save_lib`, `-save_view` or `-view` is given.
`-window_id` *`i_windowID`*	Specifies the identifier for the window to work in. If this option is not specified, the view in the active window is used.

Example

The following command extracts all nets in the active cellview. The subcells of the active cellview are flattened. For nets with terminals in the extracted cells, all of the shapes in the nets will be added to a terminal pin. This allows the router the most freedom for finding connections.

```
extract_net_connectivity -all -flatten_to_depth 2 -pin_style whole_net
```

Using -net_name_property with Calibre Layout Netlist Names Files

When GDSII data from Mentor Graphics Calibre is translated to OpenAccess, properties are attached to the layout objects whose value is a net name or number.

For example, the OpenAccess property is named `"STREAM PROPERTY #5"`, where the number 5 depends on which property number was used in Calibre for this purpose. The property value may be a name or number, indicating the net that the object belongs to. If you specify `-net_name_property "STREAM PROPERTY #5"`, the extractor will name nets using the values of those properties.

You can also specify a Calibre Layout Netlist Names (LNN) file, if available, for mappings to more meaningful names. For example, the following represents a typical set of lines in a Calibre LNN file:

```
% register
1 vdd
2 reset
```

where `%` represents a cell name and the next lines show the mapping of a number to a net name.

The following command will name nets by querying the values of `"STREAM PROPERTY #5"` properties, then mapping those values to the cell name entries in Calibre LNN file `register1.lnn`.

```
extract_net_connectivity -all -net_name_property "STREAM PROPERTY #5" \
-lnn_file register1.lnn
```

Using the example entries given previously for the LNN file, for an object in the `register` cell with a `"STREAM PROPERTY #5"` property value of `1`, the extractor will name its net `vdd`. An object with a `"STREAM PROPERTY #5"` property value of `2` will have its net named `reset`.

get_placement_grid

```
get_placement_grid
```

Returns the placement grid parameters for the active window.

Arguments

None

Value Returned

`f_xgrid f_ygrid f_xoffset f_yoffset`

Four numeric values representing the following:

- x-axis placement grid
- y-axis placement grid
- x-axis placement grid offset
- y-axis placement grid offset

`-1` Indicates the values have not been set.

Example

The following example requests the placement grid values for the current window and shows the response from Space-based Router and Chip Optimizer.

```
get_placement_grid
0.000500 0.000500 0.000000 0.000000
```

Related Information

Tcl Command	set_placement_grid

get_preferred_layers

```
get_preferred_layers
     [ -route_spec s_routeSpec ]
     [ -net s_netName ]
```

Returns the constraint command that sets the preferred routing layers for the given route spec or net. If neither the `-net` nor the -`route_spec` argument is specified, the constraint command for the active window are returned.

Arguments

`-net` *`s_netName`* Specifies a net.

`-route_spec` *`s_routeSpec`*

Specifies a route spec.

Value Returned

`s_constraintCmd` Is the constraint command (`set_constraint -constraint validRoutingLayers -hardness soft`) that can be used to set the current state of the preferred layers.

`-1` Indicates that the preferred layers have not been set.

Example

The following example requests the preferred layers for the active window.

```
get_preferred_layers
```

The following example gets the preferred layers for the `mem_data[14]` net and shows the results.

```
get_preferred_layers -route_spec LEFDefaultRouteSpec
set_constraint -constraint validRoutingLayers -hardness soft -LayerArrayValue
  { Metal2 }
```

Related Information

Tcl Command	set_preferred_layers unset_preferred_layers

get_route_on_grid

```
get_route_on_grid
```

Returns the on-grid setting for routing in the active document.

Arguments

None

Value Returned

`true` | `false`	If true, the system will only place routes on the routing grid. If false, routes must be placed on the manufacturing grid. This is the default setting.

Related Information

Tcl Command	set_route_on_grid

get_route_via_info

```
get_route_via_info
     -name s_viaName
     -infotype s_infoType
     [ -group s_groupName ]
```

Returns the requested information for the given via. You can use the returned information to sort vias based on user-defined cost functions.

Arguments

`-group s_groupName`

Specifies the name of the constraint group, or rule spec, where the via exists as a valid routing via.

Default: design rule spec

`-infotype s_infoType`

Specifies the type of information as one of the following:

cutHeight	Cut class height
cutLayerBound	Via bound for the cut layer
cutLayerName	Name of the cut layer
cutWidth	Cut class width
effectiveNumCuts	Number of effective cuts in the via
	For example, numCuts for a bar via is 2, but effectiveNumCuts is 1
indexForRouting	Current position in the list of preferred routing vias
inlinedPreferred	Specifies whether inline extensions are preferred: true (1) or false (0)
lowerLayerBound	Extensions bound for the layer below the cut layer
lowerLayerName	Name of the layer below the cut layer
meetsExtensions	Specifies whether the via meets extension rules

numCols	Number of columns in the via
numCuts	Number of cuts in the via
numRows	Number of rows in the via
offsetPreferred	Specifies the offset preference as no preference (0), centered (1) or offset (2)
prefExtDirLower	Specifies the preferred extension direction for the lower layer as vertical (0) or horizontal (1)
prefExtDirUpper	Specifies the preferred extension direction for the upper layer as vertical (0) or horizontal (1)
upperLayerBound	Extensions bound for the layer above the cut layer
upperLayerName	Name of the layer above the cut layer

`-name` *`s_viaName`* Specifies the name of the via.

Value Returned

`value` Is the value for the requested information type.

Example

The following example returns the `lowerLayerBound` for `VIA1X` in the `wide` route spec.

```
get_route_via_info -group wide -name VIA1X -infotype lowerLayerBound
```

get_router_tax

```
get_router_tax
```

Returns the current value of the wrong-way router tax that is recognized by the detail router. The wrong-way tax specifies the relative cost of using a wrong-way path to complete a connection versus using a preferred direction path and possibly additional routing.

Arguments

None

Value Returned

f_wrongWayTax	Represents a multiplier to the internal wrong-way cost set by the router. A tax greater than 1.0 tends to reduce the amount of wrong-way routing and increase the number of vias used. A tax less than 1.0 and greater than 0.0 tends to increase the amount of wrong-way routing used.

Example

The following command requests the current value for the router tax and shows the response from Space-based Router and Chip Optimizer.

```
get_router_tax
1.0
```

Related Information

Tcl Command	set_router_tax

get_routespec_taper

```
get_routespec_taper
     -route_spec s_routeSpec
```

Reports the taper route specs to input pins and to output pins for the given route spec.

Arguments

`-route_spec` *`s_routeSpec`*	Specifies the route spec to get the taper route specs for.

Value Returned

`s_taperSpecs`	Are the route specs in the following order: ■ Rulespec (the given route spec) ■ InputTaperSpec (taper route spec to input pins) ■ OutputTaperSpec (taper route spec to output pins)

Example

The following command requests the taper route spec for `isolate_double` and shows the output.

```
get_routespec_taper -route_spec isolate_double
Rulespec: isolate_double InputTaperSpec: single OutputTaperSpec: single
0
```

Related Information

Tcl Command	set_routespec_taper

get_routing_grid

```
get_routing_grid
     [ [ -layer s_layerName ][ -report] ]
     | [ -rs s_routeSpec ]
```

Returns routing grid step and offset values for the given layer or route spec.

Arguments

`-layer s_layerName`
: Specifies the name of the layer. If not specified, the entry layer is used, if it is a routing layer.

`-report`
: Outputs the routing and manufacturing grid for all layers. For more information on the grid values, refer to "report_grids" on page 1047.

`-rs s_routeSpec`
: Specifies the name of the route spec.

Value Returned

`f_xgrid f_ygrid f_xoffset f_yoffset`

Four numeric values representing the following:

- x-axis routing grid
- y-axis routing grid
- x-axis routing grid offset
- y-axis routing grid offset

Note: For values that have not been set, the routing pitch is returned for the x- and y-axis routing grid values, and the routing offset is returned for the x- and y-axis routing grid offset values.

Example

The following example gets the routing grid values for the current entry layer and shows the results.

```
get_routing_grid
2.40000 2.40000 1.200000 1.20000
```

Related Information

Tcl Command	set_routing_grid report_grids

get_routing_style

```
set_routing_style
     -style { auto | asic | chip_assembly | device }
```

Returns the routing style for the active cellview.

Arguments

None

Return Value

`asic`
`| chip_assembly`
`| device`
	Is the routing style for the active cellview.

Example

The following example sets the routing style for a device level design.

```
set_routing_style -style device
```

Related Information

Tcl Command	set_routing_style

get_soft_rule_adherence

```
get_soft_rule_adherence
     [ -default [ true | false ] ]
```

Returns the current effort level that the router will use to satisfy soft spacing rules. For a description of the effort levels, refer to Table 11-2 on page 804.

Arguments

`-default [ true | false ]`

Returns the default effort level setting for the current rules, instead of the current effort level. If soft spacing rules exist, the default setting is `medium`. If there are only hard rules, the default setting is `maximum`.

Value Returned

`{low | medium | high | maximum}`	Is the effort level.
`-1`	Command syntax error occurred.

Related Information

Tcl Command	set_soft_rule_adherence

get_sorted_route_via_list

```
get_sorted_route_via_list
     [ -group s_groupName ]
```

Returns a list of routing vias sorted in order of increasing cost, as seen by the router.

Important

> To use this command, at least one of `db.use_separate_pref_ext_dir` or `db.preserve_routing_via_order` must be set to `true`.

Via cost is based on a via's bound on the metal and cut layers, extension values, preferred orientation and offset. If `db.use_separate_pref_ext_dir` is `true`, then settings for `preferredExtensionDirection`, `inlineViaPreferred`, and `preferredViaOrigin` constraints will be considered when determining costs, otherwise those settings are ignored.

Arguments

`-group` *`s_groupName`*

Specifies the name of the constraint group, or rule spec, for which the list of sorted vias will be output.

Default: design rule spec

Value Returned

`s_viaName…`	Is the sorted list of via names.

Example

The following shows the issued command and example output.

```
setvar db.use_separate_pref_ext true
get_sorted_route_via_list
"VIA1V" "VIA1X" "VIA1H" "VIA1XR90"
```

get_treat_blockage_as_metal

```
get_treat_blockage_as_metal
     -layers {s_layerName…}
     [ -list ]
```

Returns the blockage settings for the given layers. By default, all pre-existing effective width and/or minimum spacing properties on blockages will be used and any blockages without properties will be treated like metal and their exact widths will be used to calculate spacing requirements. Properties of nearby shapes will be considered.

Arguments

`-layers` *`s_layerName`*

Specifies the name of the layers.

`-list`

Outputs the blockage settings as a Tcl list. For each layer, the layer name, followed by the settings corresponding to these set_treat_blockage_as_metal arguments: `min_width`, `min_space`, `override`, and `force_min_space`. For example,

```
"Metal1" "true" "false" "false" "false"
```

Value Returned

s_setting…

The settings for each requested layer are output to the Transcript area in the following format:

- `min_width`: If true, all unspecified shapes are assigned to a minimum effective width.
- `min_space`: If true, all unspecified shapes are assigned to a minimum space rule.
- `override`: If true, all preassigned widths and/or spacings are ignored and blockage settings are based on the other arguments given:
    - ❑ `min_space true`: All blockages are subjected to the minimum spacing rule.
    - ❑ `min_width true`: All blockages are subjected to the effective width.
    - ❑ `min_width false` and `min_space false`: All blockages are treated as metal and the width is used to calculate spacing requirements.
- `force_min_space`: If true, the minimum spacing rule on a blockage is a hard override, regardless of the rules for neighboring shapes. If false, the maximum of the spacing requirements for neighboring shapes is used.

Example

The following example gets the treatBlockageAsMetal rule for the `Metal1`, `Metal2`, and `Metal4` layers.

```
get_treat_blockage_as_metal -layers {Metal1 Metal2 Metal4}
```

The following is example output.

```
Layer Metal1 treatBlockageAsMetal: min_width true, min_space false, override false,
force_min_space false
Layer Metal2 treatBlockageAsMetal: min_width true, min_space false, override false,
force_min_space false
Layer Metal4 treatBlockageAsMetal: min_width true, min_space false, override false,
force_min_space false
```

Related Information

Tcl Command	set_treat_blockage_as_metal

get_treat_via_as_abstract

```
get_treat_via_as_abstract
```

Returns the setting for the global flag that indicates whether the via shapes should be treated as abstracts. When true, the via shapes inside a via master and minimum adjacent via spacings are not checked, but bounding box-to-bounding box spacing is checked.

Arguments

None

Value Returned

s_bool	Is the setting for the global flag.

Example

The following example gets the current setting for the whether vias should be treated as abstracts by the checker.

```
get_treat_via_as_abstract
```

Related Information

Tcl Command	set_treat_via_as_abstract

get_use_existing_shapes_for shielding

```
get_use_existing_shapes_for_shielding
```

(Virtuoso Routing IDE and Space-based Router only) Returns true if existing shapes (for example, power rails) can be used to shield nets. This is the default setting. If `false`, the router will reserve space around nets that require shielding.

Arguments

None

Related Information

Tcl Command	set_use_existing_shapes_for_shielding

get_user_grid

```
get_user_grid
```

Returns the settings for the user-specified grid.

Arguments

None

Value Returned

f_xgrid f_ygrid f_xoffset f_yoffset

Four numeric values representing the following:

- x-axis user grid
- y-axis user grid
- x-axis user grid offset
- y-axis user grid offset

Example

The following example gets the user grid values and shows the results.

```
get_user_grid
0.660000 0.660000 0.330000 0.330000
```

Related Information

Tcl Command	set_user_grid

get_via_grid

```
get_via_grid
     [ [ -layer s_layerName ][ -report ] ]
     | [ -rs s_routeSpec ]
```

Returns the grid settings for a via layer or a route spec.

Arguments

`-layer` *`s_layerName`*

Specifies the name of the via layer. If not specified, the entry layer is used if it is a via layer.

`-report`

Outputs the routing and manufacturing grid for all layers. For more information on the grid values, refer to "report_grids" on page 1047.

`-rs` *`s_routeSpec`*

Specifies the name of the route spec.

Value Returned

`f_xgrid f_ygrid f_xoffset f_yoffset`

Four numeric values representing the following:

- x-axis via grid
- y-axis via grid
- x-axis via grid offset
- y-axis via grid offset

Example

The following example gets the grid values for the `V1` via layer and shows the results.

```
get_via_grid -layer V1
1.20000 1.20000 0.400000 0.40000
```

Related Information

Tcl Command	set_routing_grid report_grids

pattern_route

```
pattern_route
     { -all | -net {s_netName…} | -set d_setObj }
     [ -check_fully_enclosed [ true | false ] ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type { [power][ground] [clock] } ]
     [ -passes i_limit ]
     [ -route_Z [ true | false ] ]
     [ -router_type { [enclosed_metal] [line] } ]
```

Routes guides using simple routing configurations (straight lines, L-shaped patterns, and Z-shaped patterns) by applying the embedded metal router, followed by the orthogonal line router. Only routes that do not violate DRC rules are added.

It is particularly useful for:

- Working with data provided by other vendors that does not maintain centerline connectivity.
- Connecting aligned pins on a datapath.

Before using this command, you should run update_net_connectivity to establish centerline connectivity and create the guides.

Arguments

`-all`

Attempts to route all guides, then automatically updates connectivity. If there are guides remaining, additional passes are run until the pattern router can no longer complete any guides. The `-passes` argument can be used to limit the number of passes attempted.

`-check_fully_enclosed [ true | false ]`

(Applies only to the enclosed metal router) If set to `true` and a proposed segment is involved in a DRC or grid violation, no segment will be added, even if the segment is fully enclosed.

If set to `false`, no checking is performed if added segments are fully enclosed. Normally, adding segments that are fully enclosed should not introduce new violations.

Default: `false`

`-exclude_net {s_netName …}`

Prevents listed nets from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-exclude_set d_setObj`

Prevents nets in the given set from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-exclude_type {[power][ground][clock]}`

Prevents nets of the given type from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-net {s_netName…}` Attempts to route all guides in the net list. Only one pass is performed.

`-passes i_limit` (Applies only when `-all` is specified) Limits the number of passes that the pattern router will attempt.

Default: no limit

`-route_Z [ true | false ]`

Determines whether the embedded metal router will attempt to use Z-shape configurations.

Default: `true`

`-router_type {[enclosed_metal][line]}`

Determines the pattern routers to try. By default, all router types are used.

`enclosed_metal`	Embedded metal router
`line`	Orthogonal line router

`-set d_setObj` Routes all guides in the set. If a net or route is included in the set, all guides in the net or route are attempted. Only one pass is performed.

Example

The following example checks the connectivity of the design, then runs the pattern router to connect same-layer disconnects.

```
update_net_connectivity -all
pattern_route -all
```

Related Information

Tcl Commands	clean_nets update_net_connectivity

read_net_connectivity

```
read_net_connectivity
     [ -lib s_libName ]
     [ -cell s_cellName ]
     [ -view s_viewName ]
     [ -verbose ]
```

Reads and updates connectivity from an OpenAccess view, usually a schematic or netlist. This command is useful after reading a design from Virtuoso® which has incorrect or incomplete connectivity with respect to a *source* schematic or netlist.

The layout instances must have names matching the source instances.

Space-based Router and Chip Optimizer will step through the instances in the source. For each instance, it will look for a like-named instance in the layout. If one is found, it will step through net connections on each pin of the source instance, making sure that the corresponding layer instance pins are connected to the same nets. It will create nets in the layout, if necessary, and ensure that top-level terminals in the layout are in the correct net, as long as the layout terminal names match the source.

Cases are supported where the source is hierarchical but the layout is *flat*. For example, given a schematic component A with an instance B1 (of component B), where component B contains two instances, C1 and C2 (of component C), the layout for A may contain instances with names B1|C1 and C1|C2, where | is the hierarchy delimiter.

By default, `read_net_connectivity` silently skips mismatch items (for example, a source instance with no corresponding layout instance). For additional information from the command, use the -`verbose` argument.

Arguments

`-cell` *`s_cellName`*
: Specifies the name of the cell containing the source view.
Default: Cell name for the active occurrence.

`-lib` *`s_libName`*
: Specifies the name of the library containing the source view.
Default: Library name for the active occurrence.

`-verbose`
: Outputs information to the Transcript area about the connectivity changes.

`-view` *`s_viewName`*
: Specifies the name of the view containing the source information.
Default: `netlist`

repair_net

```
repair_net
     [ -all | -net {s_netName…} | -set d_setObj ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -repairs {[loops][dangles][redundantSteiners][redundantRouteElements]} ]
     [ -verbose [ true | false ] ]
```

Repairs connectivity problems in the entire design, on specified nets, or on nets in a given set. Use this command to remove loops, dangles, redundant Steiner points, and/or redundant route elements.

Arguments

`-all` Repairs all nets in the design. This is the default.

`-exclude_net {s_netName…}`

Excludes the given nets from processing.

`-exclude_set d_setObj`

Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Excludes one or more given types of nets from processing.

`-net {s_netName…}` Repairs nets in the list.

`-repairs {s_name…}` Specifies one or more types of connectivity issues to repair: `dangles`, `loops`, `redundantSteiners`, and `redundantRouteElements`. By default, all of the repairs are attempted. If this argument is given without a value, no repair is performed.

`-set d_setObj` Repairs nets in the given set.

`-verbose [ true | false ]`

When set `true`, additional information about the operation is output.

Example

This command repairs all nets in the design and can change the footprint of the metal.

```
repair_net
```

Related Information

Tcl Commands	clean_nets

set_is_trunk

```
set_is_trunk
     -set d_setObj
```

Designates shapes in the set as trunk components for routing. This command is useful when establishing a *trunk* topology using set_route_topology and some of the shapes that should be part of the trunk do not meet the criteria. For example, if some shapes are under terminals or instance terminals, and are not in shape terms or in a cover macro, this command can be used to identify them as part of the trunk.

Arguments

`-set` *`d_setObj`*	Designates shape terms in the given set as trunk components.

Example

The following command designates shapes in the selected set as trunk components.

```
set_is_trunk -set [get_selection_set]
```

Related Information

Tcl Command	convert_routing_shapes set_route_topology

set_net_fix_status

```
set_net_fix_status
     {-all | -net {s_netName…} | -set d_setObj}
     -status {fixed | locked | unfixed}
```

Sets the status of all top-level nets, the named nets, or the nets in the given set to `fixed`, `locked`, or `unfixed`, which controls whether the nets can be changed by automatic tools and wire editing.

When a net’s status is `locked` or `fixed`, the routes of the net inherit the net’s status, regardless of their individual route status. When the status of a net is `unfixed`, the status of its routes can be set individually using set_route_fix_status.

A net’s status is given by the `OAConnStatus` property for the net in the Properties Browser as `normal` (unfixed), `fixed`, or `locked`. The value for the `OAConnStatus` property can also be changed in the Properties Browser or using *Edit – Fix/Unfix Nets*.

Arguments

`-all`	Sets the status of all top-level nets in the design.	
`-net {`*`s_netName…`*`}`	Sets the status of the nets in the list.	
`-set` *`d_setObj`*	Sets the status of the nets in the given set.	
`-status` *`s_statusName`*	Chooses the status to apply to the nets.	
	`fixed`	Allows the net to be changed by wire editing and prevents changes by automatic tools, including the creation of guides. All routes that are already present are considered to be `fixed`.
	`locked`	Prevents the net from being changed by any tool. All routes that are already present are considered to be `locked`.
	`unfixed`	Allows the net to be changed by all tools. The status of the individual routes of an unfixed net can be set using set_route_fix_status.

Example

The following command prevents all nets in the `$myNets` set from being changed.

```
set_net_fix_status -set $myNets -status locked
```

Related Information

Tcl Command	set_route_fix_status

set_net_priority

```
set_net_priority
     -all | -net s_netName | -set d_setObj
     -priority i_priority
```

Sets the priority property for all nets, a given net, or nets in a set.

Use this command to assign a higher net priority to critical nets or any nets that you want to route using more direct paths. If a net is *long* due to some other constraint, then you can apply your own formula to determine when a net is unacceptably routed and then re-route it using `global_route -mode eco`. Use `report_net_stats` and the Net Manager to determine the quality of the length of the net.

There are eleven levels of net priority available in any given run, with 0 being the lowest priority and 10 the highest priority. Some data translation utilities include net priority assignments. Unless you know that there are previously set net priorities that you want to keep, it is advisable to reset all net priorities prior to setting a few. The priority is relative, therefore, setting a net priority to 10 does nothing more than setting it to 1 over a default of 0.

Arguments

`-all`	Sets the priority for all nets.
`-net` *`s_netName`*	Specifies the net to set the priority for.
`-priority` *`i_priority`*	Specifies the priority value.
`-set` *`d_setObj`*	Specifies a set of nets to set the priority for.

Example

The following example resets the net priority for all nets, then sets the net priority for nets whose names begin with `data_input_bus`.

```
set_net_priority -all -priority 0
replace_set -set1 [find_net -name {data_input_bus*} -ignore_case true -no_wildcard
false ] -set2 [get_selection_set]
set_net_priority -set [get_selection_set] -priority 1
```

Related Information

Tcl Command	report_net_stats

set_net_signal_type

```
set_net_signal_type
     -net s_netName
     -signal_type {signal | power | ground | clock | tieoff | tieHi | tieLo | analog
     | scan | reset}
```

Sets the net signal type for the named net.

Arguments

`-net` *`s_netName`* Sets the signal type for the named net.

`-set` *`d_setObj`* Sets the signal type for nets in the set.

`-signal_type {signal | power | ground | clock | tieoff | tieHi | tieLo | analog | scan | reset}`

Specifies the signal type to assign to the named net.

Example

The following example sets the signal type for `mypowerNet` to `power`.

```
set_net_signal_type -net mypowerNet -signal_type power
```

set_placement_grid

```
set_placement_grid
     -snap {f_xgrid f_ygrid}
     [ -x_offset f_offset ]
     [ -y_offset f_offset ]
```

Sets placement grid parameters for the design in the active window.

Arguments

`-snap {f_xgrid f_ygrid}`	
	Specifies the placement grid.
`-x_offset f_offset`	Specifies the offset from 0 on the x-axis.
`-y_offset f_offset`	Specifies the offset from 0 on the y-axis.

Example

The following example sets the x- and y-axis placement grid values.

```
set_placement_grid -snap {2.0 2.0}
```

Related Information

Tcl Command	get_placement_grid

set_power_type

```
set_power_type
     -set d_setObj
     -power_type {unknown | cell_row | stripe | block_ring | core_ring | pad_ring
     | tap | tie_off | standard_cell_term | shield | shield_tie_off}
     [ -convert_to_term [ true | false ] ]
```

Sets the type property for routes and terms on power or ground nets only. Other objects, such as nets in the set, are ignored. If no valid objects are found in the set, the command fails.

This command is used to ensure that power structure objects are correctly labeled. Space-based Router and Chip Optimizer uses a heuristic to guess what these objects are, especially when they are created externally. An incorrect guess can prevent the object from being properly connected. For example, a *tap* pin in one design, that should be connected pin-to-trunk, was incorrectly labeled as a tie-off, so the power router did not connect to it. With this command, the pin can be identified as a tap without modifying the OpenAccess database.

Arguments

`-convert_to_term [ true | false ]`

If `true`, converts routes in the set to terminals.

Caution

After this command is run, the conversion cannot be undone.

Default: `false`

`-power_type` *`s_powerType`*

Specifies the power type. In each case, the corresponding property is given in parentheses.

The following settings are acceptable for route objects:

`shield` Shield wire (`PowerShieldRouteType`)

`shield_tie_off` Used to tie a shield wire to a power rail (`PowerShieldTieOffRouteType`)

`tie_off`	Used to tie off a route that was added by the signal router (`PowerTieOffRouteType`)
`tap`	Pin-to-trunk route (`PowerTapRouteType`)

The following settings are acceptable for term objects:

`unknown`	(`UnknownTermType`)
`cell_row`	Followpin (`PowerStrapType`)
`stripe`	Power stripe (`PowerStrapType`)
`block_ring`	Block ring (`PowerBlockRingType`)
`core_ring`	Core ring (`PowerCoreRingType`)
`pad_ring`	Pad ring (`PowerPadRingType`)
`tap`	Connects to a ring or stripe by pin-to-trunk (`PowerTapTermType`)
`tie_off`	Connected to a ring or stripe by the signal router (`PowerTieOffTermType`)
`standard_cell_term`	Standard cell power pin that should be connected with a followpin (`PowerStandardCellTermType`)

`-set` *`d_setObj`* Sets the power type for route or term objects in the set.

Example

The following example sets a term power type to `PowerTapTermType`. The term must be on a power or ground net and an element of the term is selected.

```
set_power_type -set [get_selection_set] -power_type tap
```

set_preferred_layers

```
set_preferred_layers
     -layers {s_layerName…}
     -hardness {hard|soft}
     [ -net s_netName | -set d_setObj ]
```

Sets the preferred routing layers for the given net or nets in the given set.

Arguments

`-hardness {hard|soft}`

Specifies whether the preferred layers given are `hard` (required) or `soft` (requested) constraints. When `-hardness hard` is used, only the given preferred layers can be used for routing. With `-hardness soft`, the given layers are preferred for routing but other layers can be used, for example, to avoid congestion.

`-layers {s_layerName …}`

Specifies the preferred routing layers.

`-net s_netName`	Assigns the preferred routing layers to the given net.
`-set d_setObj`	Specifies a set of nets for which to assign the preferred routing layers.

Example

The following example sets the preferred layers for the `mem_data[14]` net.

```
set_preferred_layers -layers { "met3" "met4" } -net "mem_data\[14\]" -hardness soft
```

Related Information

Tcl Command	destroy_unused_netoverride_groups get_preferred_layers set_routespec_taper unset_preferred_layers

set_route_fix_status

```
set_route_fix_status
     {-all | -layers {s_layerName…} | -net {s_netName…} | -set d_setObj}
     -status {fixed | locked | unfixed | blocked}
     [ -type {all | shield | tieoff} ]
```

Sets the status of routes to `fixed`, `locked`, `blocked`, or `unfixed`. The status of nets are not changed by this command, use set_net_fix_status instead. If a net is `locked` or `fixed`, the routes of the net inherit the net's status, regardless of the individual status of the routes.

The status of a route is given by the `routeFix` property in the Properties Browser. The value of the `routeFix` property can also be changed in the Properties Browser.

Arguments

`-all` Sets the status of all top-level routes in the design.

`-layers {`*`s_layerName…`*`}`

Sets the status of all shapes on the listed layers by breaking up routes as needed. For example, if a segment on a given layer is in the middle of a route, the route will be broken into three routes: one route on either side of the segment, and the segment as a separate route. The route status for the segment will be set by the command and the other two routes will keep their setting.

`-net {`*`s_netName…`*`}` Sets the status of the routes on the listed nets.

`-set` *`d_setObj`* Sets the status of the objects in the given set.

If a net is included, the status of the routes on the net are changed, not the status of the net.

If a segment or via is included, the command can break up a route into two or more routes, if the command changes the route status. For example, if a segment in the middle of a route is given, the route will be broken into three routes: one route on either side of the segment, and the segment as a separate route. The route status for the segment will be set by the command and the other two routes will keep their setting.

`-status` *`s_statusName`* Chooses the status to apply to the routes.

	`blocked`	Blocked routes and all of the shapes on them are treated like blockages. The autorouter cannot move, delete, or connect to blocked routes. Antennas are not removed.
	`fixed`	Fixed routes can be hand-edited with the interactive wire editor. The autorouter cannot move or delete fixed routes but can connect to them. Antennas are not removed.
	`locked`	Locked routes cannot be changed by any tools, except to add Steiner points. Locked routes cannot be hand-edited. The autorouter cannot move or delete locked routes but can connect to them. Antennas are not removed.
	`unfixed`	Allows the routes to be changed by all tools.
`-type` *`s_typeName`*	Limits processing to routes of the given type.	
	`all`	All types are included. This is the default.
	`shield`	Limits processing to shield routes.
	`tieoff`	Limits processing to tieoff routes.

Example

The following command prevents all shapes on the `Metal2` layer from being changed.

```
set_net_fix_status -layer Metal2 -status locked
```

Related Information

Tcl Command	set_net_fix_status

set_route_on_grid

```
set_route_on_grid
     -on_grid [ true | false ]
     | -style {hybrid_gridded | strictly_gridded | manufacturing}
```

Chooses the on-grid setting for routing in the active design.

Arguments

`-on_grid {true | false}`

Chooses one of the following routing modes for the router:

`false` (Equivalent to `-style manufacturing`) This is gridless mode. Permits routing off the routing grid. Edges of all shapes must be on the manufacturing grid. If the manufacturing grid is not defined, a warning message is issued.

`true` (Default, equivalent to `-style hybrid_gridded`) The router tries to stay on a routing grid and can connect to off-grid pins following the manufacturing grid, if necessary. If either the manufacturing or the routing grid is not defined, a warning message is issued.

`-style` *`s_type`*

Specifies the grids that will be used for routing.

`hybrid_gridded` (default, equivalent to `-on_grid true`) The router tries to stay on a routing grid and can connect to off-grid pins following the manufacturing grid, if necessary.

`manufacturing` (Equivalent to `-on_grid false`) This is gridless mode. Edges of all shapes must be on the manufacturing grid.

`strictly_gridded`	Requires routing on a routing grid with no exceptions.

Related Information

Tcl Command	get_route_on_grid

set_route_style

```
set_route_style
     -from_style {global | detail | redundant}
     -to_style {global | detail | redundant}
     {-all | -net {s_netName…} | -set d_setObj}
     [ -ignore_active_route_status [ true | false ] ]
```

Changes the route style for the named nets, nets/routes/vias in the given set, or all top-level routes. For example,

- Change global routes to detailed routes.
- Change detailed routes to global routes.
- Change via:redundant to via:global.
- Change via:redundant to via:detail.

(Virtuoso Routing IDE and Space-based Router only) This command is useful if the global router is run in the ECO flow and makes a long connection. For these cases, you would typically use the full routing flow (croute, then detail_route) to complete the connection. In this scenario, use `set_route_style` instead of `croute` to quickly change routes from global to detail, then use the detail router to finish the routing.

Arguments

`-all` Operates on top-level routes in the entire design.

`-from_style {global | detail | redundant}`

Specifies the current style of routes/vias to change. When `redundant` is specified, `via:redundant` and `wire:detail` shapes will be changed to the `to_style`.

`-ignore_active_route_status [ true | false ]`

If `true`, then fixed/locked/blocked routes and vias belonging to fixed/locked/blocked routes will also be processed.

Default: Fixed/locked/blocked routes are not processed. Vias on fixed/locked/blocked routes are not processed unless they are explicitly included in the set given by -set.

`-to_style {global | detail | redundant}`

Specifies the new route style for the selected routes/vias. When `redundant` is specified, vias of the current style will be changed to `via:redundant` and wires of the current style will be changed to `wire:detail`.

`-net` **{***`s_netName…`***}**

Names the nets to operate on.

`-set` *`d_setObj`*

Operates on nets/routes/vias in the given set. Vias in the set will be processed even when they are part of a fixed/locked/blocked route. By default, vias on a fixed/locked/blocked route are skipped.

Example

The following command changes routes in net `netA` from global to detailed:

```
set_route_style -from_style global -to_style detail -net netA
```

The following command changes redundant vias on nets in the selected set to `via:detail`:

```
set_route_style -from_style redundant -to_style detail -set [get_selection_set]
```

Redundant vias on fixed routes can be changed to `via:detail` by creating a set containing those redundant vias and issuing the command:

```
set_route_style -from_style redundant -to_style detail -set $setOfRedundantVias
```

Alternatively, the following command changes redundant vias on nets in the selected set to `via:detail`, including redundant vias on fixed routes:

```
set_route_style -from_style redundant -to_style detail -set [get_selection_set]
-ignore_active_route_status
```

Related Information

Documentation	“ECO Routing Examples” on page 698

set_route_topology

```
set_route_topology
     -net s_netName | -set dsetObj
     -pattern {steiner | trunk}
```

Sets the route pattern for the given net or nets in the given set that determines how the nets will be routed.

By default, a *Steiner* minimum spanning tree topology is used, where a vertex can connect to any other vertex and steiner points can be added to decrease the length of the interconnection.

In *trunk* routing, some portion of the net, typically a preroute, is identified as a trunk. All other vertices are connected to the trunk. Only shapes in shape terms or in a cover macro are automatically designated as parts of the trunk. For shapes that do not meet this criteria but should be included, some additional processing is needed:

- For shapes that are placed under routes, use convert_routing_shapes to convert them to shape terms.

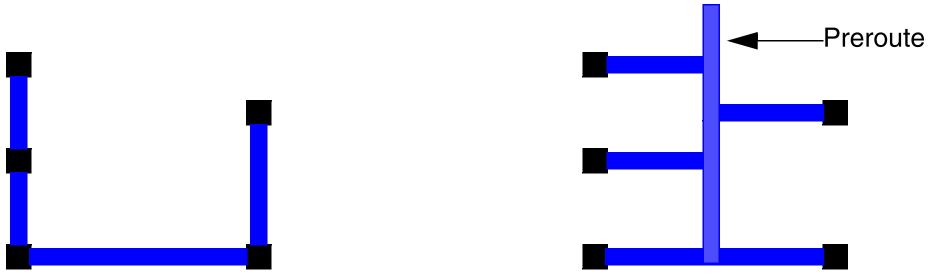
  ```
  convert_routing_shapes -to_special -set d_setOobj
  ```

- For shapes that are placed under terminals or instance terminals, and not in shape terms or in a cover macro, use set_is_trunk to include them in the trunk.

  ```
  set_is_trunk -set d_setObj
  ```

If trunk topology is specified and no trunk shapes are found, the route topology defaults to the Steiner topology.

The following figure compares the implementation of the two topology types for a net.



Steiner Routing

Trunk Routing

Arguments

`-net s_netName`

Specifies the name of the net.

`-pattern s_patternName`

Specifies the route pattern as one of the following:

`steiner`	Permits each vertex to connect to any other vertex and allows steiner points to be added to decrease the length of the interconnection. This is the default.
`trunk`	Connects all vertices to a trunk.

`-set d_setObj` Sets the route pattern for nets in the set.

Example

The following command sets `netA` to use the steiner topology.

```
set_route_topology -net netA -pattern steiner
```

Related Information

Tcl Command	convert_routing_shapes set_is_trunk

set_router_tax

```
set_router_tax
     -wrongway f_wrongWayTax
```

Modifies the internal costs used by the router. These costs influence how the router routes wires.

Arguments

`-wrongway` *`f_wrongWayTax`*	Specifies the relative cost of using a wrong-way path to complete a connection versus using the preferred direction path and possibly additional ways. A real value greater than 0.0 and less than or equal to 100.0 can be specified, and represents a multiplier to the internal wrong-way cost set by the router. A tax greater than 1.0 tends to reduce the amount of wrong-way routing and increase the number of vias used. A tax less than 1.0 and greater than 0.0 tends to increase the amount of wrong-way routing used. The wrong-way tax is recognized by the detail router. Default: 1.0

Example

The following example reduces wrong-way tax from the default value of 1.0, increasing the likelihood that wrong-way routing will be used to complete connections.

```
set_router_tax -wrongway 0.50
```

Related Information

Tcl Command	get_router_tax

set_routespec_taper

```
set_routespec_taper
     [ -route_spec s_routeSpec ]
     [ [ -taper_route_spec s_routeSpec ]
     | [ -input_taper_spec s_routeSpec -output_taper_spec s_routeSpec ] ]
```

Sets the *taper-to* route spec for a route spec or all route specs or disallows tapering on a route spec or all route specs.

By default, all non-default route specs taper to the global net default route spec (typically LEFDefaultRouteSpec) and the `maxTaperWindow` is 10 tracks (a track is calculated as the average pitch of the layers). The `maxTaperWindow` determines the distance, in microns, from a term where the influence of the taper rule stops and switches to the net's constraints.

For more information on setting tapers, refer to "Using Tapers" on page 790.

Arguments

`-input_taper_spec` *`s_routeSpec`*

Specifies the input taper route spec.

`-output_taper_spec` *`s_routeSpec`*

Specifies the output taper route spec.

`-route_spec` *`s_routeSpec`*

Specifies the target route spec.

`-taper_route_spec` *`s_routeSpec`*

Specifies the taper route spec for the given target route spec. If this argument is not specified, tapering is not allowed on the target route spec.

Example

The following example allows tapering on the `isolate_double` route spec to the `double` route spec.

```
set_routespec_taper -route_spec isolate_double -taper_route_spec double
```

The following example disallows tapering on all route specs.

```
set_routespec_taper
```

The following example disallows tapering on the `isolate_double` route spec.

```
set_routespec_taper -route_spec isolate_double
```

Related Information

Tcl Command	get_routespec_taper

set_routing_grid

```
set_routing_grid
     {[ -x f_grid ] [ -y f_grid ] [ -x_offset f_offset ] [ -y_offset f_offset ]}
     [ -layer {s_layerName …} | -rs s_routeSpec ]
```

Sets routing grid parameters for one or more layers, a route spec, or the entry layer in the active window.

Important

Grid and offset values must be multiples of the manufacturing grid.

Arguments

`-layer {`*`s_layerName …`*`}`

Specifies the layer names. If not specified, all layers are set.

`-rs` *`s_routeSpec`* — Specifies the name of the route spec.

`-x` *`f_grid`* — Specifies where route segments can begin and end, and where vias can be placed on the x-axis.

`-x_offset` *`f_offset`*

Specifies the offset from 0 on the x-axis for the router to use for this layer.

`-y` *`f_grid`* — Specifies where route segments can begin and end, and where vias can be placed on the y-axis.

`-y_offset` *`f_offset`*

Specifies the offset from 0 on the y-axis for the router to use for this layer.

Example

The following example sets the x- and y-axis routing grid values for `met1` and `met2`.

```
set_routing_grid -layer {met1 met2} -x 2.0 -y 2.0
```

Related Information

Tcl Command	get_routing_grid set_route_on_grid

set_routing_style

```
set_routing_style
     -style { auto | asic | chip_assembly | device }
```

Sets the routing style based on the design type.

Arguments

`-style` *`s_style`*	Specifies the routing style as one of the following:	
	`auto`	The routing style is determined based on the design characteristics. This is the default.
	`asic`	Design is mostly standard cells, may contain a few macros, and is routed on the routing grid.
	`chip_assembly`	Design is comprised mostly of macros, may contain some standard cells, and is routed on the manufacturing grid.
	`device`	Design is a block design that contains transistors, has poly routing, and is routed on the manufacturing grid.

Example

The following example sets the routing style for a device level design.

```
set_routing_style -style device
```

Related Information

Tcl Command	get_routing_style
Menu Command	(Space-based Router only) *Route – Design Setupr*

set_soft_rule_adherence

```
set_soft_rule_adherence
     [ -check_rules [ true | false ] ]
     [ -effort {low | medium | high | maximum} | -default [ true | false ] ]
```

Specifies how much effort the router should use to satisfy soft spacing rules. The effort levels trade off soft rule adherence with convergence, wire length, and run time.

Arguments

`-check_rules [ true | false ]`

If set to `true`, determines whether the soft rules are the same as the hard rules. If they are the same, no effort level change is made.

Default: `true`

`-default [ true | false ]`

When `true`, restores the effort level to the system default.

`-effort` *`s_effort`*

Specifies the level of effort the router should use to satisfy soft rules.

Default: If soft spacing rules exist, `medium` effort is used. If there are only hard rules, `maximum` effort is used.

`low`	Router makes little effort to make space for soft rules, but may use them if there is space available.
`medium`	Router attempts to make room for soft rules at earlier stages but abandons the attempt in detail route.
`high`	Router attempts to make room for soft rules until several pass in detail route.
`maximum`	Router attempts to apply the soft rules, only abandoning the attempt if it fails to make the connection with the soft rules.

Related Information

Tcl Command	get_soft_rule_adherence

set_taper_width_nets

```
set_taper_width_nets
     -all | -set d_setObj | -net {s_netName…}
     [ -exclude_net {s_netName…}
     [ -exclude_set d_setObj ]
     [ -net_width_value f_distance
     | -net_width [max_pin | min_pin | avg_pin | none ] ]
     [ -report [ true | false ] ]
     [ -use_pref_dir [ true | false ] ]
     [ -base_CG_for_taper [s_routeSpec] ]
     [ -create_net_width_taper [ true | false ] ]
     [ -taper_style [to_first_via | within_pin_halo]
     [ -adjust_width_to_mfg_grid [ true | false ] ]
     [ -honor_net_width [ true | false ] ]
     [ -min_width_on_all_layers [ true | false ] ]
     [ -pin_layers {s_layerName…} ]
     [ -clear_dangling_rule_specs [ true | false ] ]
     [ -include_power_gnd_nets [ true | false ] ]
     [ -create_taper_to_avoid_min_edge [ true| false ] ]
```

Creates a taper spec for the pins of the specified nets which forces the router to connect to those pins using the pin width in the preferred access direction. If the pin can be accessed from all directions, use `-use_pref_dir false` to use the minimum dimension of the pin shape. The tapers for pins that are comprised of multiple shapes will use the net's `minWidth`. The taper specs are assigned as `taper` constraint groups for the terminals or instance.

To remove taper specs that were created using `set_taper_width_nets` but are no longer used, use the `-clear_dangling_rule_specs` argument.

To unset taper specs that are created using this command, use unset_taper_width_nets.

Arguments

`-adjust_width_to_mfg_grid [ true | false ]`

When true, taper width is adjusted, if needed, to be on the manufacturing grid and satisfy the `minEdgeLength` rules.

Default: `false`

`-all` Creates taper specs for all nets in the design.

`-base_CG_for_taper [s_routeSpec]`

When this argument is specified, the created taper spec will inherit all constraints (except `minWidth`) from the given route spec or, by default, from the net's route spec if no route spec is named.

`-clear_dangling_rule_specs [ true | false ]`

Removes all taper specs there were originally created by `set_taper_width_nets` but are no longer used. By default, unused taper specs are not removed.

`-create_min_taper_window [ true | false ]`

Specifies whether the `minTaperWindow` constraint will be set for the created taper spec. By default (`false`), the `minTaperWindow` constraint is not set and if the routing can connect to pins using the net's width, no tapering is used. When set `true`, the taper spec's `minTaperWindow` constraint is set to 5*pitch, forcing tapering.

`-create_net_width_taper [ true | false ]`

When `true`, creates a taper spec even when the net width matches the computed taper width. When `false`, a taper spec is only created if the net width does not match the computed taper width.

Default: `true`

`-create_taper_to_avoid_min_edge [ true | false ]`

When `true`, creates a taper spec only when needed to avoid minimum edge length violations with the pin shape.

Default: `false`

`-exclude_net` **{***`s_netName`***…}**

Excludes nets in the list from processing. No taper spec is created for these nets.

`-exclude_set` *`d_setObj`* Excludes nets in the set from processing. No taper spec is created for these nets.

`-honor_net_width [ true | false ]`

If true, will not override net `minWidth` and `maxWidth` constraints when creating the taper spec.

Default: `false`

`-include_power_gnd_nets [ true | false ]`

When `true`, power and ground nets will be included. By default (`false`), power and ground nets are excluded.

`-min_width_on_all_layers [ true | false ]`

Creates a taper spec, with `minWidth` equal to the pin width, for each routing layer. By default (`false`), creates a taper spec, with `minWidth` equal to the pin width, for only the pin layer.

`-net` **{***`s_netName`***…}** Creates taper specs for the nets in the list.

`-net_width [max_pin | min_pin | avg_pin | none ]`

Uses the specified pin width per net to route the non-taper section of the net. The net width is determined by the value of this argument and the `-use_pref_dir` setting. Use `-report` to compare the chosen width with the values of the other options.

`avg_pin`	Uses the average pin width.
`max_pin`	Uses the largest pin width.
`min_pin`	Uses the smallest pin width.
`none`	(Default) Uses the pin's width for the routes within the taper window, and the net's `minWidth` for the non-taper routes.

`-net_width_value` *`f_distance`*

Sets this distance, in microns, as the minWidth for the nets.

`-pin_layers` **{***`s_layerName`***…}**

Restricts processing to pin shapes on the specified layers.

`-report [ true | false ]`

(Applies only when `-net_width` is given) Outputs the minimum, maximum, and average pin widths, and the chosen net width for each processed net.

`-set` *`d_setObj`* Creates taper specs for the nets in the given set.

`-taper_style [to_first_via | within_pin_halo]`

Controls the taper window. If this argument is not specified for the command, the default is `to_first_via`.

`to_first_via`	Sets the `taperToFirstVia` constraint to `true` for the created taper spec. This results in tapering from the pin to the first layer change for the specified nets.
`within_pin_halo`	Performs tapering only when needed within `oaTaperHalo` (preferred), `maxTaperWindow`, or within 10 tracks of the pin if neither of these constraints is set.

`-use_pref_dir [ true | false ]`

If `true`, uses the width in the preferred direction for pins that can be accessed from all directions. If `false`, the minimum dimension of the pin shape is used.

Default: `true`

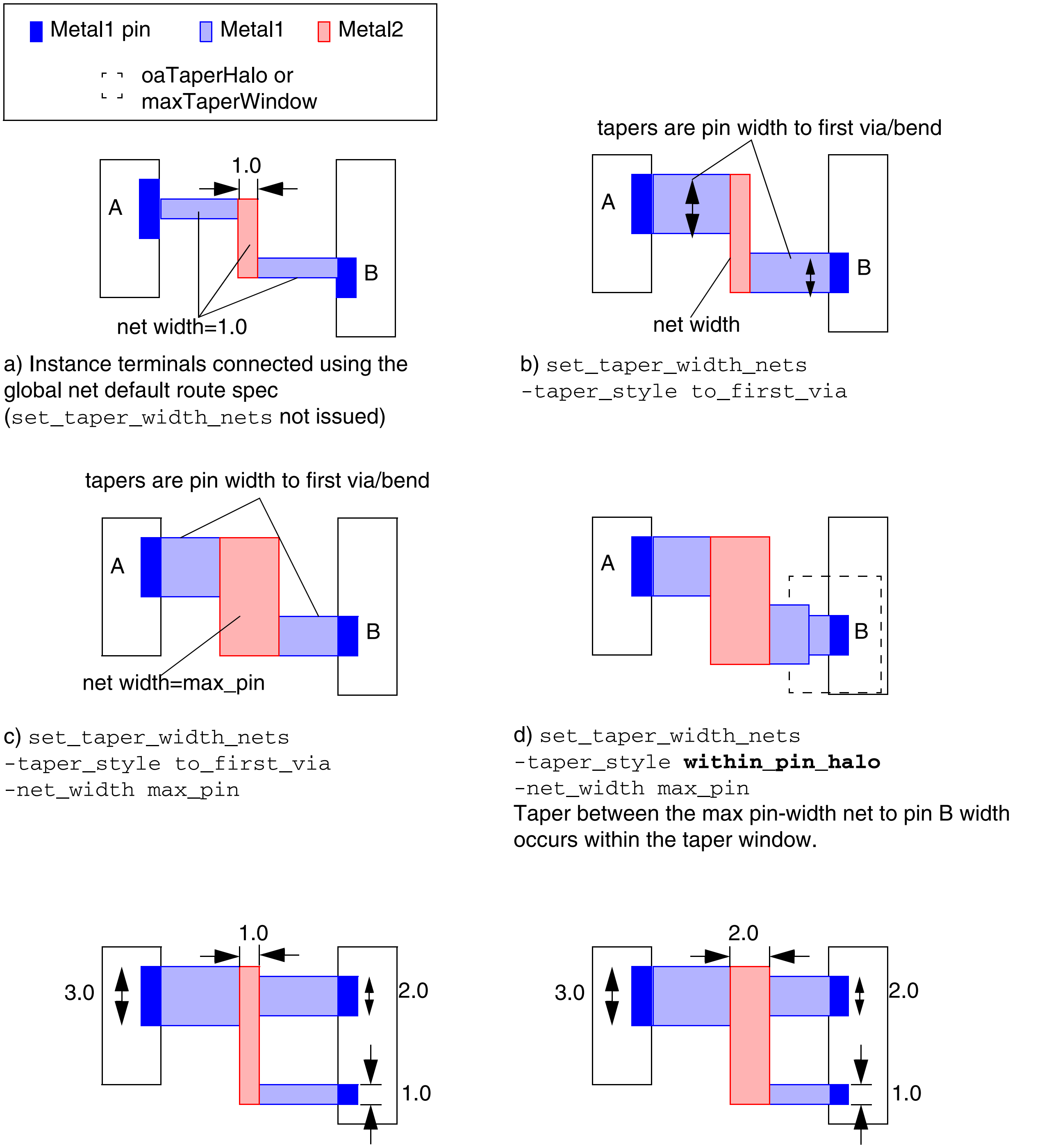
Example

The following command creates a taper spec for `netA` and causes the net to be connected to the pins using the pin width in the preferred access direction.

```
set_taper_width_nets -net netA
```

Figure 7-4 on page 448 shows a comparison example of a net routed using the global net default route spec (with `minWidth` less than the width of the pins) and the same net routed after `set_taper_width_nets` is issued for the net. In b), the pin taper is routed using the width of the pins to the first layer change or via. In c) and d), the entire net is routed using the maximum pin width and tapers are at pin width to the first via or layer change for c) and within the pin halo for d). Finally, e) and f) show a three-pin net routed using the minimum pin width for e) and the average pin width for f).

Figure 7-4 Examples for set_taper_width_nets -net_width -taper_style



a) Instance terminals connected using the global net default route spec (`set_taper_width_nets` not issued)

b) `set_taper_width_nets -taper_style to_first_via`

c) `set_taper_width_nets -taper_style to_first_via -net_width max_pin`

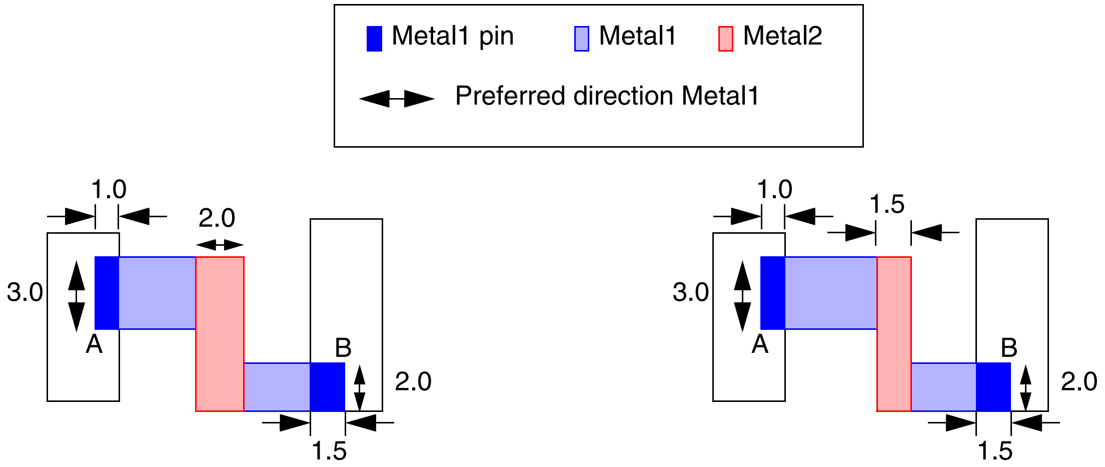
d) `set_taper_width_nets -taper_style` **`within_pin_halo`** `-net_width max_pin`
Taper between the max pin-width net to pin B width occurs within the taper window.

e) `set_taper_width_nets -net_width min_pin`
Tapers to first via/bend by default. Net width is the minimum of the three pin widths.

f) `set_taper_width_nets -net_width ave_pin`
Tapers to first via/bend by default. Net width is the average of the three pin widths.

Figure 7-5 on page 449 shows the effect of `use_pref_dir` settings on a net.

Figure 7-5 Examples for set_taper_width_nets -use_pref_dir



a) `set_taper_width_nets -net_width min_pin -use_pref_dir true`
Pin B width (2.0) in the preferred direction is smaller than the Pin A width (3.0) so net width is 2.0. When `-taper_style` is not specified, tapers to first via/bend by default.

b) `set_taper_width_nets -net_width min_pin -use_pref_dir false`
Pin B can be accessed horizontally and vertically so its minimum dimension (1.5) is used for the net width. Tapers to first via/bend by default.

Related Information

Tcl Command	set_constraint_group set_routespec_taper unset_taper_width_nets

set_treat_blockage_as_metal

```
set_treat_blockage_as_metal
     {-layers {s_layerName…} | -get_layers {s_layerType…}}
     {[ -min_width [ true | false ] ] | [ -min_space [ true | false ] ]}
     [ -override [ true | false ] ]
     [ -eol_style [ preferred | nonpreferred | ring | orthogonal ] ]
     [ -force_min_space [ true | false ] ]
     [ -span_style [ preferred | nonpreferred | ring | orthogonal ] ]
```

Determines the effective width and/or spacing to use for blockages on one or more layers or layer types. By default, all blockages are treated as metal and use the minimum spacing rule for blockage-to-neighbor shape spacing.

Using environment variables, you can control which blockages are loaded when you open a design. For more information on these variables, refer to "Database Environment Variables" on page 1142.

If you load a design with the LEF property `USEMINSPACING OBS ON`, the blockage settings will automatically be set to `min_width true`, `min_space false`, `override false`, `force_min_space false`.

Note: It is illegal for a blockage to have both a minimum spacing property and a minimum width property, so the `-min_width` and `-min_space` arguments are mutually exclusive.

Arguments

`-eol_style [preferred | nonpreferred | ring | orthogonal]`

Specifies how to treat blockages with an effective width for end-of-line (EOL) checking. Refer to Figure 7-6 on page 452 for graphic examples of each style.

`nonpreferred`	Parallel wires of effective width oriented perpendicular to the layer's preferred direction
`orthogonal`	Combination of preferred and nonpreferred wires
`preferred`	Parallel wires of effective width oriented in the layer's preferred direction

`ring`	(Default) A single ring of effective width inward from the outer edge of the blockage

`-force_min_space [ true | false ]`

When `true`, the minimum spacing rule on all blockages is used for blockage-to-neighboring shape spacing. When `false`, Space-based Router and Chip Optimizer calculates and uses the larger of minimum spacing for the blockage and the neighboring shape. By default, this value is `true`.

`-get_layers [[metal][cut][poly][other][all]]`

Processes layers of the given types. By default, all types are processed.

`-layers` **{***`s_layerName`***…}**

Specifies the name of the layers to set.

`-min_space [ true | false ]`

Assigns all unspecified shapes to a minimum space rule. By default, this value is `false`.

`-min_width [ true | false ]`

Assigns all unspecified shapes to a minimum effective width. By default, this value is `false`.

`-override [ true | false ]`

Ignores all preassigned widths and/or spacings and determines blockage settings based on the other arguments given:

`-min_space`	Subjects all blockages to the min space rule.
`-min_width`	Subjects all blockages to the effective width.
(none)	Treats all blockages as metal and uses the width to calculate spacing requirements.

The default for this argument is `false.`

`-span_style [preferred | nonpreferred | ring | orthogonal]`

Specifies how to treat blockages with an effective width for span checking. Refer to Figure 7-6 on page 452 for graphic examples of each style.

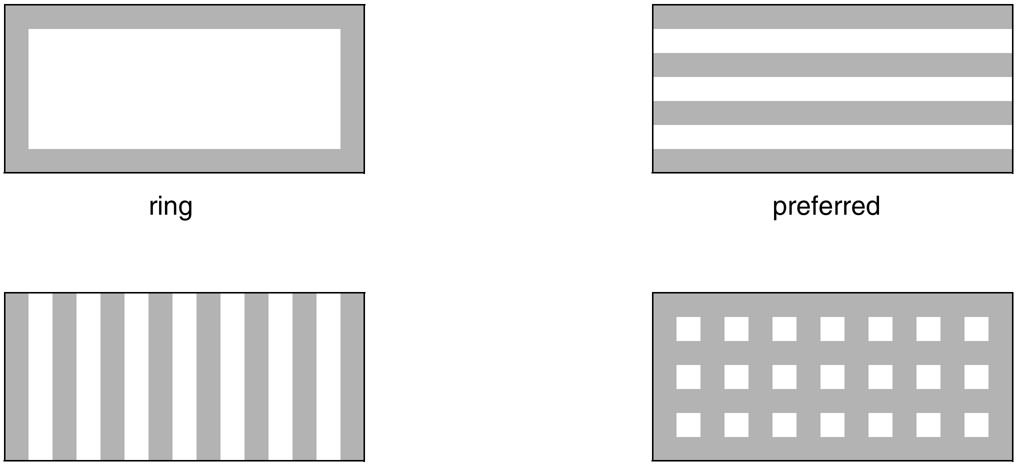
`nonpreferred` Parallel wires of effective width oriented perpendicular to the layer's preferred direction

`orthogonal` Combination of preferred and nonpreferred wires

`preferred` (Default) Parallel wires of effective width oriented in the layer's preferred direction

`ring` A single ring of effective width inward from the outer edge of the blockage

Figure 7-6 Blockage Modeling Examples for eol_style and span_style



ring

preferred

nonpreferred

orthogonal

Example

The following table gives examples of how to interpret the argument settings.

Description	`min_width`	`min_space`	`override`	`force_min_space`
Use preset effective width properties and use minimum effective width for all blockages with no effective width properties	true	false	false	false
Use preset effective spacing properties and use minimum effective spacing for all blockages with no minimum spacing properties	false	true	false	false
Use minimum width for all blockages regardless of effective minimum width properties	true	false	true	false
Use minimum spacing for all blockages regardless of effective minimum spacing properties	false	true	true	false
Use the minimum spacing rule around blockages regardless of the rules for neighboring shapes (Default)	false	false	false	true

The following example represents the default settings, which use the minimum spacing rule around blockages, regardless of the rules of neighboring shapes.

```
set_treat_blockage_as_metal -layers $metal_layers -min_width false -min_space
false -override false -force_min_space true
```

Related Information

Tcl Command	get_treat_blockage_as_metal

set_treat_via_as_abstract

```
set_treat_via_as_abstract
     -abstract [ true | false ]
```

Sets the global flag that indicates whether vias should be treated as abstracts by the checker.

Arguments

`-abstract [ true | false ]`	
	Specifies the setting for the global flag. If true, vias will be treated as abstracts by the checker and the via shapes inside a via master and minimum adjacent via spacings will not be checked, but bounding box-to-bounding box spacing will be checked.

Example

The following example sets the global flag to treat vias as abstracts.

```
set_treat_via_as_abstract -abstract true
```

Related Information

Tcl Command	get_treat_via_as_abstract

set_use_existing_shapes_for_shielding

```
set_use_existing_shapes_for_shielding
     -share [ true | false ] ]
```

(Virtuoso Routing IDE and Space-based Router only) Specifies whether existing power/ground shapes can be used to shield nets.

Arguments

`-share [true	false]`	
	If set `true`, existing power/ground shapes can be used to shield nets. If set `false`, existing power/ground shapes will not be used for shielding, so the router will reserve space around nets for shield wires where required. The default setting is `true`.

Related Information

Tcl Command	get_use_existing_shapes_for shielding

set_user_grid

```
set_user_grid
     {[ -x f_grid ] [ -y f_grid ] [ -x_offset f_offset ] [ -y_offset f_offset ]}
```

Sets the user grid parameters for the active window.

Arguments

`-x` *`f_grid`*	Specifies the x-axis user grid.
`-x_offset` *`f_offset`*	Specifies the x-axis user grid offset from 0.
`-y` *`f_grid`*	Specifies the y-axis user grid.
`-y_offset` *`f_offset`*	Specifies the y-axis user grid offset from 0.

Example

The following example sets the user grid values for the active window.

```
set_user_grid -x 0.660000 -y 0.660000 -x_offset 0.330000 -y_offset 0.33000
```

Related Information

Tcl Command	get_user_grid

set_via_grid

```
set_via_grid
     {[ -x f_grid ] [ -y f_grid ] [ -x_offset f_offset ] [ -y_offset f_offset ]}
     [ -layer {s_layerName …} | -rs s_routeSpec ]
```

Sets via grid parameters for one or more via layers or the entry layer (if it is a via layer) in the active window.

Arguments

`-layer {`*`s_layerName`*` …}`	
	Specifies the via layer names. If not specified, all via layers are set.
`-rs` *`s_routeSpec`*	Specifies the name of the route spec.
`-x` *`f_grid`*	Specifies where vias can be placed on the x-axis.
`-x_offset` *`f_offset`*	Specifies the offset from 0 on the x-axis for placing vias on this via layer.
`-y` *`f_grid`*	Specifies where vias can be placed on the y-axis.
`-y_offset` *`f_offset`*	Specifies the offset from 0 on the y-axis for placing vias on this via layer.

Example

The following example sets the x- and y-axis via grid values for `via1` and `via2`.

```
set_via_grid -layer {via1 via2} -x 2.0 -y 2.0
```

Related Information

Tcl Command	get_via_grid

set_width_priority_nets

```
set_width_priority_nets
     -width f_userunit | -tracks f_tracks
     [ -set d_setObj ]
     [ -priority i_priority ]
     [ -ignore_nets {s_netName…} ]
     [ -ignore_set d_setObj ]
```

Sets the priority for nets with widths greater than or equal to a given width (in user units) or a given number of tracks. You can optionally add the qualifying nets to a set or exclude some nets from consideration.

Arguments

`-ignore_nets {s_netName…}`

Specifies the names of nets to exclude from consideration.

`-ignore_set d_setObj`

Excludes the nets in the given set from consideration.

`-priority i_priority`

Specifies the priority to set the qualifying nets to. If this argument is not given, qualifying nets will be set to the highest priority value.

`-set d_setObj` Puts qualifying nets into this set.

`-tracks f_tracks` Specifies the width (in tracks) that qualifying nets must meet or exceed.

`-width f_userunit` Specifies the width (in user units) that qualifying nets must meet or exceed.

Example

The following example causes all nets on any layer that are at least 1.3 user units to be set to the highest priority value and puts the nets into the selected set.

```
set_width_priority_nets -width 1.3 -set [get_selection_set]
```

unassign_term

```
unassign_term
     -inst {s_instName | d_ctuObj}
     -term s_termName
```

Removes a term from a net.

Arguments

`-inst {`*`s_instName`*`| `*`d_ctuObj`*`}`	
	Specifies the name of the instance or the object identifier for the instance.
`-term `*`s_termName`*	Specifies the terminal name.

Example

The following example disconnect term `QN` of the `I1` instance from the net.

```
unassign_term -inst I1 -term QN
```

Related Information

Tcl Commands	assign_term

unset_preferred_layers

```
unset_preferred_layers
     [ -net {s_netName…} | -set d_setObj ]
     [ -rule_spec s_ruleSpec ]
```

Removes the preferred routing layers for the given nets, objects in the given set, or the given rule spec.

Arguments

`-net {s_netName…}`	Removes the preferred routing layers for the given nets.
`-rule_spec s_ruleSpec`	
	Removes the preferred routing layers only on the given rule spec.
`-set d_setObj`	Removes the preferred routing layers for objects in the given set.

Related Information

Tcl Command	get_preferred_layers set_preferred_layers

unset_taper_width_nets

```
unset_taper_width_nets
     -all | -set d_setObj | -net {s_netName…}
     [ -include_power_gnd_nets [ true | false ] ]
     [ -delete_taper_specs [ true | false ] ]
```

Unsets taper specs that were created by set_taper_width_nets for a given set of nets, all nets in a list, or all the nets in the design. You can include power and ground nets or remove taper specs using optional arguments.

Arguments

`-all`

Unsets taper specs that were created by set_taper_width_nets for all nets in the design.

`-delete_taper_specs [ true | false ] ]`

Removes taper specs on the term/instTerm belonging to the nets processed by this command.

`-include_power_gnd_nets [ true | false ]`

When `true`, power and ground nets are included when this command is processed. By default (`false`), power and ground nets are excluded from processing.

`-net {s_netName…}`

Creates taper specs that were created by set_taper_width_nets for the nets in the list.

`-set d_setObj`

Creates taper specs that were created by set_taper_width_nets for the nets in the given set.

Example

The following command unsets the taper spec for `netA` that was created by set_taper_width_nets.

```
unset_taper_width_nets -net netA
```

Related Information

Tcl Command	set_taper_width_nets

8

Power Route Commands

This chapter describes the Tcl commands for the power router. These commands are available when running Virtuoso® Space-based Router.

For a simple power supply network, you add power components in the following sequence:

- Pad Ring
- Core Rings
- Block Rings
- Stripes
- Standard Cell Row Straps
- Vias for Interlayer Connections
- Pin-to-Trunk Connections

If you have trouble connecting power structure objects, refer to "Troubleshooting the Power Router" on page 547.

The Power Route Tcl commands are presented in alphabetical order:

- proute_stripes on page 520
- proute_trim_stripes on page 536
- proute_via_insertion on page 539

delete_proute

```
delete_proute
     [ -net s_netName
     | -set d_setObj [ -delete_connected_routes [ true | false ] ] ]
     [ -layers {s_layerName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -power_only [ true | false ] ]
```

Removes power objects. By default, if no arguments are given, all power objects from all power nets, including all routes and top-level shapeTerms, are removed. You can optionally remove objects within a given region or on only specific layers. You can also remove signal net objects with `-power_only false`.

Arguments

`-delete_connected_routes [ true | false ]`

Used with `-set`, checks each term containing each low-level type object (rectShapes, routeSegments, routeVias) in the given set and, if the term contains only the set object, then any routes connected to that term are removed.

`-layers` **{***s_layerName…***}** Only removes objects on these layers.

`-net` *s_netName* Removes all power objects in the net.

`-power_only [ true | false ]`

(Applies only with `-net` or -`set`) When `true`, only power and ground nets are processed. When `false`, signal nets are also processed.
Default: `true`

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

Removes, splits, or trims objects inside this rectangular region.

`-set` *d_setObj* Removes objects in the set (nets, routes, rectShapes, routeSegments, routeVias) from their power nets.

proute_block_ring

```
proute_block_ring
     -layers {s_layerName s_layerName …}
     -nets {s_netName…}
     {-net_width f_userunit | -layer_width {f_width…}}
     {-set d_setObj | -instances {s_instName…}}
     [ -block_clearance {f_userunit | {f_left f_bottom f_right f_top}}
     |-in_block_clearance f_userunit ]
     [ -contour [ true | false ] [ -min_jog f_length ] ]
     [ -channels [ true | false ] ]
     [ -depopulate {[L] [B] [R] [T] [H] [V]} ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…} ]
     [ -lattice [ true | false ] ]
     [ -net_clearance f_userunit ]
     [ -power_only [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -undoable [ true | false ] ]
```

Adds rings for the selected set of instances.

Note: To add interlayer connections for the block rings, use proute_via_insertion.

Arguments

`-block_clearance {`*`f_userunit`*` | {`*`f_left f_bottom f_right f_top`*`}}`

If one argument value is given, it must be a non-negative number for the clearance outward from the block boundary to the innermost ring.

If four values are given, they may be any real numbers for clearances to the left, bottom, right, and top, respectively. Positive values are measured outward from the block boundary to the innermost ring. Negative values indicate distance inward from the block boundary to the innermost edge of the innermost ring.

By default, the clearance is the greater of the minimum clearance of the two layers specified by `-layers`.

`-channels [ true | false ]`

When `true`, adds rails in the channels between blocks. By default, rails are not added between blocks. For an example using channels, refer to “Example 3 — Contoured Block Ring Created Around a Pair of Blocks with Channels” on page 470.

`-contour [ true | false ]`

If true, the block ring will follow the contour of the selected blocks. By default, the block ring will be rectangular.

`-depopulate` **{**`[L] [B] [R] [T] [H] [V]`**}**

Prevents one or more segments of the block ring from being added. This argument does not apply to channel segments between blocks.

`B`	Bottom segment
`H`	Horizontal segments
`L`	Left segment
`R`	Right segment
`T`	Top segment
`V`	Vertical segments

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes` **{***`s_purposeName`*…**}**

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-in_block_clearance` *`f_userunit`*

Specifies the clearance inward from the block boundary to the outermost ring. The value must be non-negative. The block rings will be added inside the block boundary and will be rectangular.

`-instances` **{***`s_instName`*…**}**

Encloses the instances given in the list.

`-lattice [ true | false ]`

When `true`, extends ring segments of duplicate nets to form a lattice. By default and when `false`, concentric rings are maintained. For examples, refer to "Example 4 — Concentric Block Rings (Default)" on page 470 and "Example 5 — Latticed Block Rings" on page 470.

`-layer_width` **{***`f_width1 f_width2…`***}**

Specifies the width, in microns, to use for each layer given in the `-layers` list, respectively, for all nets.

`-layers` **{***`s_layer1Name s_layer2Name …`***}**

Specifies at least two orthogonal metal layers to use for the block ring of each net. One layer must have a preferred horizontal direction and one other layer must have a preferred vertical direction.

`-min_jog` *`f_length`*

Removes contour jogs shorter than the given length. The default value is equal to `net_width` plus `net_clearance`.

`-net_clearance` *`f_userunit`*

Specifies the spacing between the power nets routed. By default, this value is the greater of the minimum clearance of the two layers.

`-net_width` *`f_userunit`*

Specifies the total width for each net routed. By default, this value is the greater of the minimum widths of the two layers.

If you specify a net width value that is greater than the `maxWidth` of one of the given layers, multiple rings are added for each net such that the following are true:

- All added wire is equal in width.
- The width of the added wire is greater than or equal to the greater of the `minWidth` of the two layers and less than or equal to the lesser of the `maxWidth` of the two layers.
- The sum of the wire widths for each net in a ring set is equal to the given net width.

Note: Some roundoff error can cause the total width to be slightly greater than specified net width.

`-nets` **{***`s_netName…`***}**

Specifies the nets to route. One or more nets can be given. When more than one net is given, the list order of the nets determines the placement of the rings, with the first net on the innermost ring, followed by the second, and so on, to the last net of the list on the outermost ring.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-set` *`d_setObj`* Specifies the set of instances to enclose.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

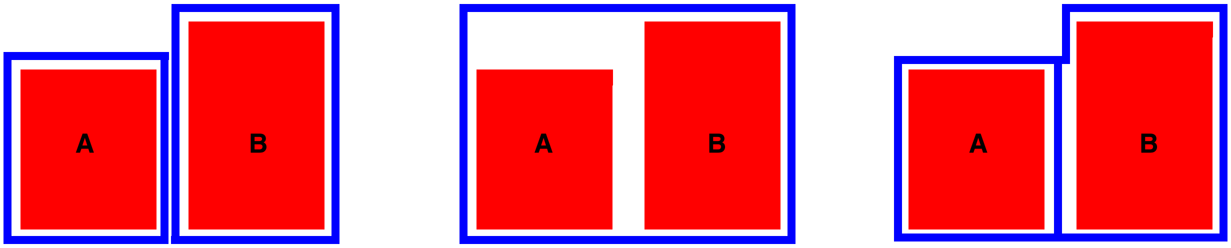
Default: `false`

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

Example

In the following examples, block rings are created for two blocks, A and B. In these examples, `BLKA` is the set containing block A, `BLKB` is the set containing block B.



Example 1
Block Rings Created Around Individual Blocks

Example 2
Rectangular Block Ring Created Around A Pair of Blocks

Example 3
Contoured Block Ring Created Around A Pair of Blocks with Channels

Example 1 — Block Rings Created Around Individual Blocks

```
proute_block_ring -set $BLKA -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS}
proute_block_ring -set $BLKB -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS}
```

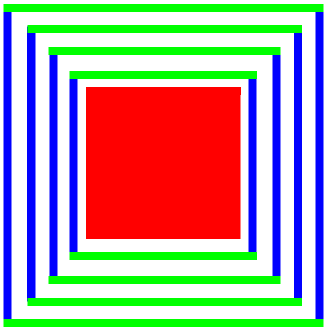
Example 2 — Rectangular Block Ring Created Around a Pair of Blocks - No Contouring

```
set BLKA_B [or_sets -set1 $BLKA -set2 $BLKB]
proute_block_ring -set [$BLKA_B] -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS}
```

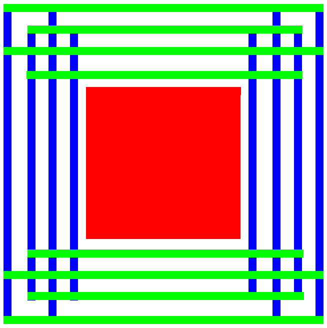
Example 3 — Contoured Block Ring Created Around a Pair of Blocks with Channels

```
set BLKA_B [or_sets -set1 $BLKA -set2 $BLKB]
proute_block_ring -set [$BLKA_B] -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS} -contour -channels
```

The next examples illustrate how to configure block rings as concentric rings or latticed rings. This option is only effective when you specify duplicate nets surrounding a block. For these examples, four rings are created around a block, two for VDD and two for VSS. The list order determines the placement of the rings from the innermost ring to the outermost ring: `VDD`, `VSS`, `VDD`, `VSS`.



Example 4
Concentric Block Rings
(Default)



Example 5
Latticed Block Rings

Example 4 — Concentric Block Rings (Default)

```
proute_block_ring -set $BLKA -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS VDD VSS}
```

Example 5 — Latticed Block Rings

```
proute_block_ring -set $BLKA -layers {met2 met3} -net_width 8 -net_clearance 2
-nets {VDD VSS VDD VSS} -lattice
```

proute_cell_row

```
proute_cell_row
     {-layers {s_layerName…} | -pin_layers {s_layerName…}}
     -nets {s_netName…}
     [ -routing_area {f_xlo f_ylo f_xhi f_yhi}]
     [ -extend_to_boundary_with_pins [ true | false ]
       | {-extend | -extend_long} [ true | false ]
      [ -ring_layers {s_layerName…} ]
      [ -partial_overlap [ true | false ] ] ]
     [ -class_applies_to_top_level_rings [ true | false ] ]
     [ -core_ring_layers {s_layerName…} ]
     [ -direction {horizontal | vertical} ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…} ]
     [ -incomplete_rings [ true | false ] ]
     [ -class1 {s_className …} ]
     [ -class2 {s_className …} ]
     [ -class3 {s_className …} ]
     [ -class4 {s_className …} ]
     [ -class5 {s_className …} ]
     [ -stop_at_rings_of_class1 {s_className…} [ -ring_layers1 {s_layerName…} ] ]
     [ -incomplete_rings1 [ true | false ] ]
     [ -stop_at_rings_of_class2 {s_className…} [ -ring_layers2 {s_layerName…} ]]
     [ -incomplete_rings2 [ true | false ] ]
     [ -stop_at_rings_of_class3 {s_className…} [ -ring_layers3 {s_layerName…} ] ]
     [ -incomplete_rings3 [ true | false ] ]
     [ -stop_at_rings_of_class4 {s_className…} [ -ring_layers4 {s_layerName…} ] ]
     [ -incomplete_rings4 [ true | false ] ]
     [ -stop_at_rings_of_class5 {s_className…} [ -ring_layers5 {s_layerName…} ] ]
     [ -incomplete_rings5 [ true | false ] ]
     [ -all_term_types {core_ring | block_ring | stripes | cell_row_straps
      | unknown} ]
     [ -ignore_blockage_of_class {s_className…} ]
     [ -stop_at_boundary_of_class {s_className…}
       [ -boundary_clearance {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class1 {s_className…}
       [ -boundary_clearance1 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class2 {s_className…}
       [ -boundary_clearance2 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class3 {s_className…}
       [ -boundary_clearance3 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class4 {s_className…}
       [ -boundary_clearance4 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class5 {s_className…}
       [ -boundary_clearance5 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
```

```
    [ -max_msg_count i_count ]
    [ -max_row_gap f_userunit ]
    [ -power_only [ true | false ] ]
    [ -row_end [ true | false ] ]
    [ -silent [ true | false ] ]
    [ -undoable [ true | false ] ]
    [ -use_cellname_as_class [ true | false ] ]
```

Adds straps along aligned pins of standard cells.

For faster processing, you can use proute_row_straps, which requires that rows be defined and that the placement of the straps for those rows be carefully specified.

Note: To add interlayer connections for the straps, use proute_via_insertion.

Tip

When adding cell row straps, check the `signalType` property for the net. If the net is not specified as `power` or `ground`, you must either change the property value or specify `-power_only false` to route the cell row straps.

Specifying Block Boundary Clearances

You can prevent straps from extending across block boundaries by specifying the class of macro blocks in a list given by one of the `stop_at_boundary_of_class` arguments. Up to five unique lists of classes can be specified, each with a different clearance specification, given by the corresponding `boundary_clearance` argument. For the boundary clearance, if a single value is given, it applies to all four sides of the block. If a list of four values is given, they must be given in the exact order of left, bottom, right, and top. Positive values are measured outward from the block boundary to the end of the strap. Negative values indicate the distance inward from the block boundary to the end of the strap.

Arguments

`-all_term_types {core_ring | block_ring | stripes | cell_row_straps | unknown}`

Treats the given classes of macro block terminals on specified ring layers as this power type.

-boundary_clearance {*f_clearanceValue* | **{***f_leftValue f_bottomValue f_rightValue f_topValue***}**}

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance1 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class1` argument. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance2 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class2` argument. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance3 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class3`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance4 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class4`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance5 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class5`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-class_applies_to_top_level_rings [ true | false ]`

Stops strap extensions at top-level block rings around blocks whose class is specified in the “of class” arguments.

Default: `false`

`-class1` **{***`s_className`***…}**

Specifies a list of classes of macro blocks that belong to class1 for “of_class1” arguments.

`-class2` **{***`s_className`***…}**

Specifies a list of classes of macro blocks that belong to class2 for “of_class2” arguments.

`-class3` **{***`s_className`***…}**

Specifies a list of classes of macro blocks that belong to class3 for “of_class3” arguments.

`-class4` **{***`s_className`***…}**

Specifies a list of classes of macro blocks that belong to class4 for “of_class4” arguments.

`-class5` **{***`s_className`***…}**

Specifies a list of classes of macro blocks that belong to class5 for “of_class5” arguments.

`-connect_inst_terms [ true | false ]`

If `true`, adds specific connections from cell row straps to corresponding instance terminals.

Default: `false`

`-core_ring_layers` **{***`s_layerName`***…}** `| all`

Limits the layers of core-ring segments for determining the core ring bounds. If this is not specified, then `ring_layers` applies. If `ring_layers` is not specified, then all core-ring segments apply.

`-direction {horizontal | vertical}`

Specifies the routing direction for the cell row straps.

Default: preferred routing direction

`-extend [ true | false ]`

If `true`, cell row straps are extended to the nearest power rail. If a cell row strap cannot at least partially overlap the power rail, it will not be extended. If `false`, cell row straps are not extended.

Default: `false`

`-extend_long [ true | false ]`

Same as `-extend` but, when `true`, attempts to extend any unextended cell row to the core ring.

Default: `false`

`-extend_to_boundary_with_pins [ true | false ]`

When `true`, attempts to extend cell row straps to the design prBoundary with added pins. Refer to Figure 8-1 on page 481 for an illustration.

Default: `false`, cell row straps are not extended beyond the cells

`-ignore_blockage_of_class` **{***`s_className…`***}**

Ignores the blockages in macro blocks that belong to a class in this list.

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes` **{***`s_purposeName…`***}**

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-incomplete_rings [ true | false ]`

When `true`, top-level block rings are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Note: If there are incomplete ring segments on concave U-turns surrounding a block, cell rows cannot be placed in the U-turn area.

Default: `false`

`-incomplete_rings1 [ true | false ]`

When `true`, internal block rings of class 1 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Default: `false`

`-incomplete_rings2 [ true | false ]`

When `true`, internal block rings of class 2are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Default: `false`

`-incomplete_rings3 [ true | false ]`

When `true`, internal block rings of class 3 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Default: `false`

`-incomplete_rings4 [ true | false ]`

When `true`, internal block rings of class 4 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Default: `false`

`-incomplete_rings5 [ true | false ]`

When `true`, internal block rings of class 5 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the cell rows, that is specific to this condition.

Default: `false`

`-layers` **{***`s_layerName`* …**}**

Specifies the layers to connect.

`-max_msg_count` *`i_count`* Specifies the maximum number of message for each message type to output.

Default: 10

`-max_row_gap` *`f_userunit`*

Specifies the largest cell-row gap to be connected. By default, all aligned standard cells in each row are strapped, regardless of gap distance.

`-nets` **{***`s_netName`*…**}** Specifies the nets to route.

`-partial_overlap [ true | false ]`

If `true`, allows strap extensions to partially overlap with other shapes on the same net and layer.

Default: `false`

`-pin_layers` **{***`s_layerName`* …**}**

Specifies the pin layers to connect.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-ring_layers` **{***`s_layerName`***}**

(Applies only when -`extend` or -`extend_long` is given) Specifies the layers on which to extend cell rows.

`-ring_layers1` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class1`. Default: all layers

`-ring_layers2` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class2`. Default: all layers

`-ring_layers3` **{***`s_layerName`***…}**

Specifies the ring layers for `-stop_at_rings_of_class3`. Default: all layers

`-ring_layers4` **{***`s_layerName`***…}**

Specifies the ring layers for `-stop_at_rings_of_class4`. Default: all layers

`-ring_layers5` **{***`s_layerName`***…}**

Specifies the ring layers for `-stop_at_rings_of_class5`. Default: all layers

`-routing_area` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the outer bounds for the area of the standard cells to add straps to. Connects only pins that intersect the area and extends cell row straps to the area bounds.

`-row_end [ true | fals e ]`

If true, routes standard cells to the end of the defined rows. By default, standard cells are routed only to the last cell in the row.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

`-stop_at_boundary_of_class` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class1` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance1` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class2` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance2` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class3` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance3` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class4` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance4` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class5` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance5` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_rings_of_class1` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class2` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class3` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class4` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class5` **{***`s_className…`***}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

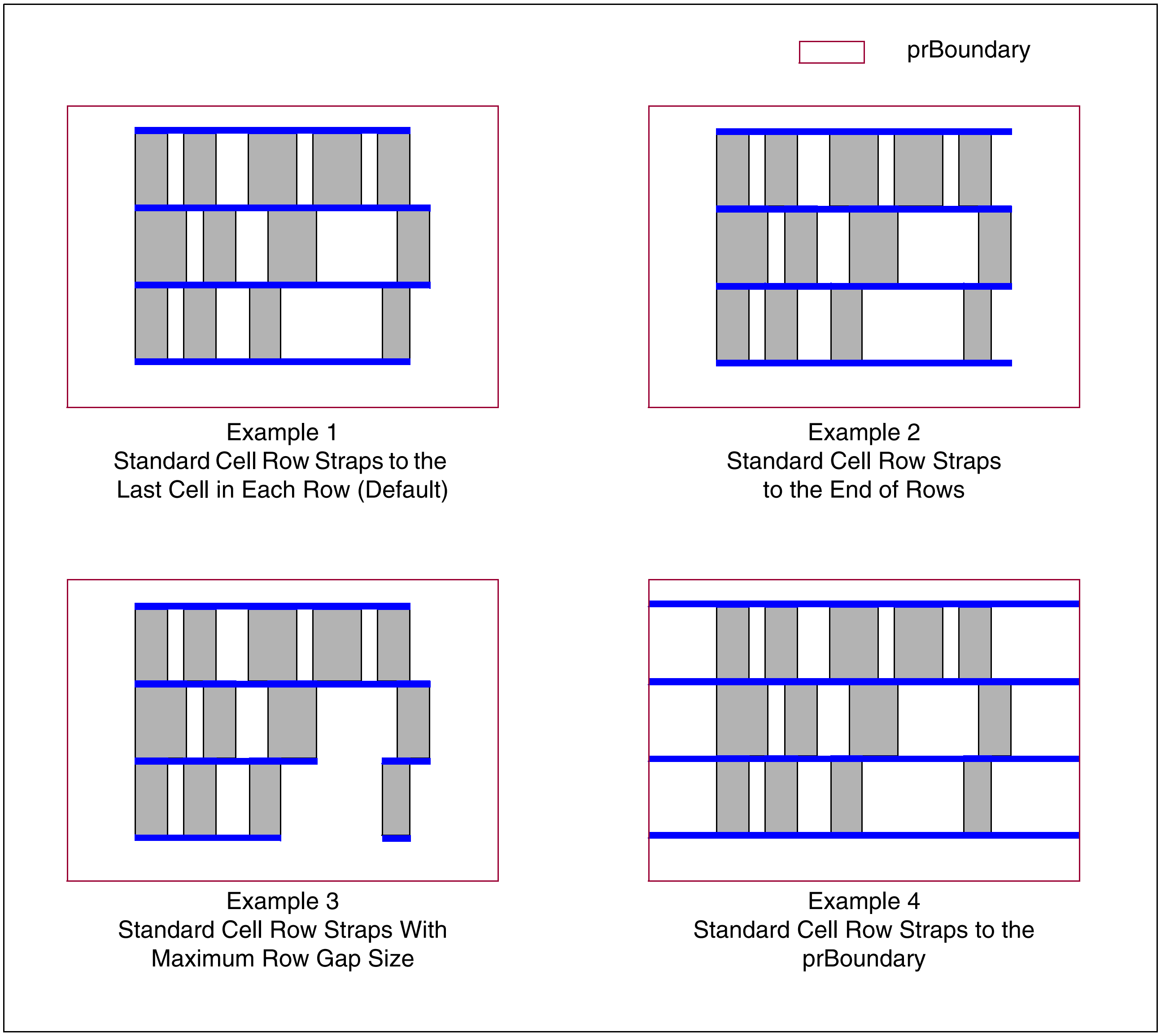
`-use_cellname_as_class [ true | false ]`

Allows specification of cellnames instead of class names for the `*of_class*` arguments.

Example

Figure 8-1 on page 481 shows examples that illustrate how the `row_end`, `max_row_gap`, and `extend_to_boundary_with_pins` arguments can be used to control the placement of standard cell row straps.

Figure 8-1 Examples of Cell Row Placement



Example 1
Standard Cell Row Straps to the
Last Cell in Each Row (Default)

Example 2
Standard Cell Row Straps
to the End of Rows

Example 3
Standard Cell Row Straps With
Maximum Row Gap Size

Example 4
Standard Cell Row Straps to the
prBoundary

Example 1 — Strap to the Last Cell in Each Row (Default)

```
proute_cell_row -layers met1 -nets {VDD VSS}
```

Example 2 — Strap to the End of Rows

```
proute_cell_row -layers met1 -nets {VDD VSS} -row_end
```

Example 3 —Strap with Maximum Row Gap Size

```
proute_cell_row -layers met1 -nets {VDD VSS} -max_row_gap 20
```

Example 4— Strap to the prBoundary with Pins

```
proute_cell_row -layers met1 -nets {VDD VSS} -extend_to_boundary_with_pins true
```

proute_core_ring

```
proute_core_ring
     -layers {s_layerName s_layerName …}
     -nets {s_netName…}
     {-net_width f_userunit | -layer_width {f_width …}}
     [ -contour [ true | false ] [ -min_jog f_userunit ] ]
     [ -core_clearance f_userunit
      | -pad_clearance f_userunit
      | -in_area_clearance f_userunit
        {-routing_area {f_xlo f_ylo f_xhi f_yhi}
        | -routing_area {f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 f_x4 f_y4
                        f_x5 f_y5 f_x6 f_y6 …}
        | -use_border_blockage}
      | -out_area_clearance f_userunit
        {-routing_area {f_xlo f_ylo f_xhi f_yhi}
        | -routing_area {f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 f_x4 f_y4
                        f_x5 f_y5 f_x6 f_y6 …}} ]
     [ -depopulate {[L] [B] [R] [T] [H] [V]} ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…} ]
     [ -lattice [ true | false ] ]
     [ -net_clearance f_userunit ]
     [ -power_only [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -undoable [ true | false ] ]
```

Adds rings around the core of a design that has pads, around the entire design without pads, inside the design bounds, or inside or outside of a given region.

The ring location is dependent on the optional setting of one of the following ring location reference arguments:

- `-core_clearance`
- `-pad_clearance`
- `-in_area_clearance` with `-routing_area` or -`use_border_blockage`
- `-out_area_clearance` with `-routing_area`

If none of these arguments is given and peripheral pads exist, then the rings are centered in the area between the core and the pads. If peripheral pads do not exist, rings are placed starting from the minimum clearance outward from the design bounds.

The *core bounds* is the rectilinear outline of the outermost prBoundaries of all non-pad, non-standard cell instances, and, if rows are defined, the bounds of all rows, or, if no rows are defined, the prBoundaries of all standard cells.

The *design bounds* is the prBoundary of the top-level design (occurrence). If no prBoundary is defined, then the design bounds is the bounding box of the design.

Arguments

`-contour [ true | false ]`

If `true`, the core ring will follow the rectilinear contour of the core, pads, or `routing_area`, depending on the ring location reference clearance specified. If `false` or when no ring location reference clearance is specified, the core ring will be rectangular.
Default: `false`

`-core_clearance` *`f_userunit`*

If peripheral pads exist, indicates the ring's clearance outward from the core bounds. If peripheral pads do not exist, indicates the ring's clearance outward from the design bounds. Must be a positive number.

`-depopulate` **{**`[L] [B] [R] [T] [H] [V]`**}**

Prevents one or more segments of the core ring from being added.

`B`	Bottom segment
`H`	Horizontal segments
`L`	Left segment
`R`	Right segment
`T`	Top segment
`V`	Vertical segments

`-layer_width` **{***`f_width1 f_width2…`***}**

Specifies the width, in microns, to use for each layer given in the `-layers` list, respectively, for all nets.

`-layers` **{***`s_layer1Name s_layer2Name`* …**}**

Specifies at least two orthogonal metal layers to use for the core ring of each net. One layer must have a preferred horizontal direction and one other layer must have a preferred vertical direction.

`-min_jog` *`f_userunit`*

Removes contour jogs that are less than the given length, in user units, measured centerline-to-centerline.

Default:

- `(lattice true)` `min_jog` = number of nets * (`net_width` + `net_clearance`)
- `(lattice false)` `min_jog` = (`net_width` + `net_clearance`)

`-net_clearance` *`f_userunit`*

Specifies the minimum spacing between the power nets. By default, this value is the greater of the minimum clearance of the two layers specified by `-layers`.

`-net_width` *`f_userunit`*

Specifies the total width for the nets routed. By default, this value is the greater of the minimum widths of the two layers.

If you specify a net width value that is greater than the `maxWidth` of one of the given layers, multiple rings are added for each net such that the following are true:

- All added wire is equal in width.
- The width of the added wire is greater than or equal to the greater of the `minWidth` of the two layers and less than or equal to the lesser of the `maxWidth` of the two layers.
- The sum of the wire widths for each net in a ring set is equal to the given net width.

Some roundoff error can cause the total width to be slightly greater than specified net width.

`-nets` **{***`s_netName…`***}**

Specifies the nets to route. One or more nets can be given. When more than one net is given, the list order of the nets determines the placement of the rings, with the first net on the innermost ring, followed by the second, and so on, to the last net of the list on the outermost ring.

`-out_area_clearance` *`f_userunit`*

Requires that `-routing_area` also be given. Indicates the ring's clearance outward from the routing area. Must be a non-negative number.

`-pad_clearance` *`f_userunit`*

If peripheral pads exist, indicates the ring's clearance inward from the pads. If peripheral pads do not exist, indicates the ring's clearance inward from the design bounds. Must be a positive number.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-routing_area {`*`f_xlo f_ylo f_xhi f_yhi`*`}`
`-routing_area` **{***`f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 f_x4 f_y4 f_x5 f_y5 f_x6 f_y6 …`***}**

Indicates the reference ring location in conjunction with an area clearance (`-in_area_clearance` or `-out_area_clearance`) given as two xy coordinate pairs for a rectangular area, or as six or more xy coordinate pairs for a polygonal area.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

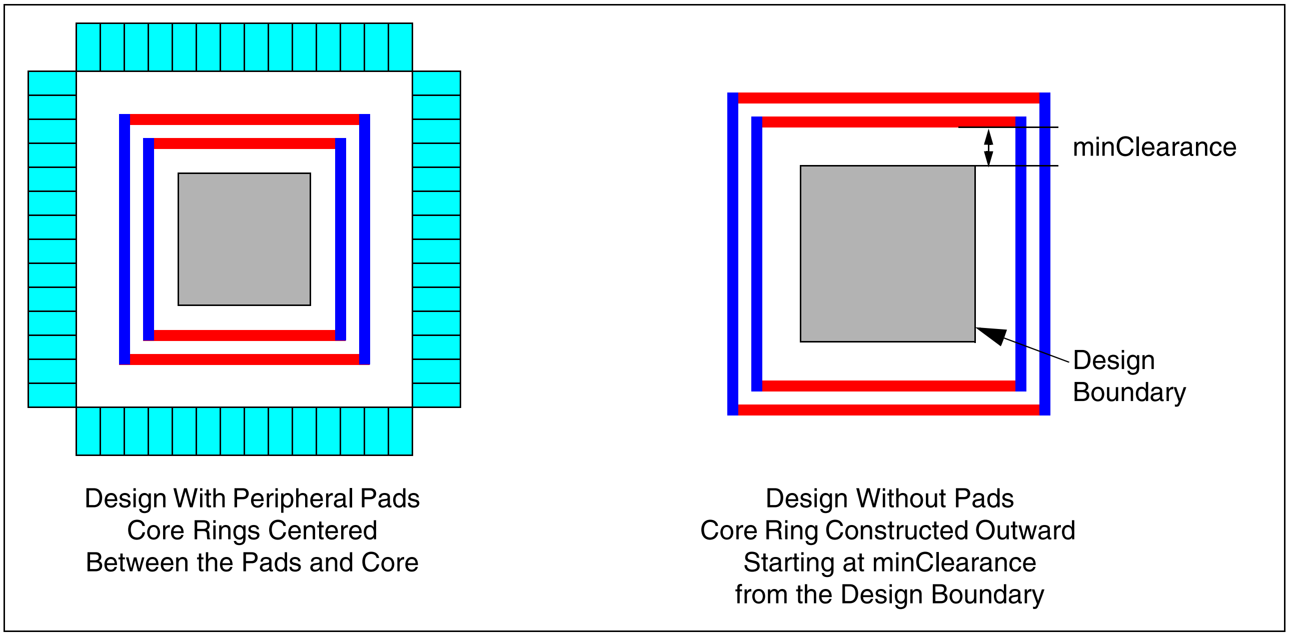
`-use_border_blockage`

Places rings inside blockages that touch the design boundary on the ring layers. If specified, both `-in_area_clearance` and `-contour` must also be given. Cannot be used with `-routing_area`.

Example

No Ring Location Reference Arguments Given

If the design has peripheral pads, the core rings are centered in the area between the pads and the core of the design. If the design has no pads, the core rings surround the entire design starting from `minClearance`, outward from the design boundary.



Design With Peripheral Pads
Core Rings Centered
Between the Pads and Core

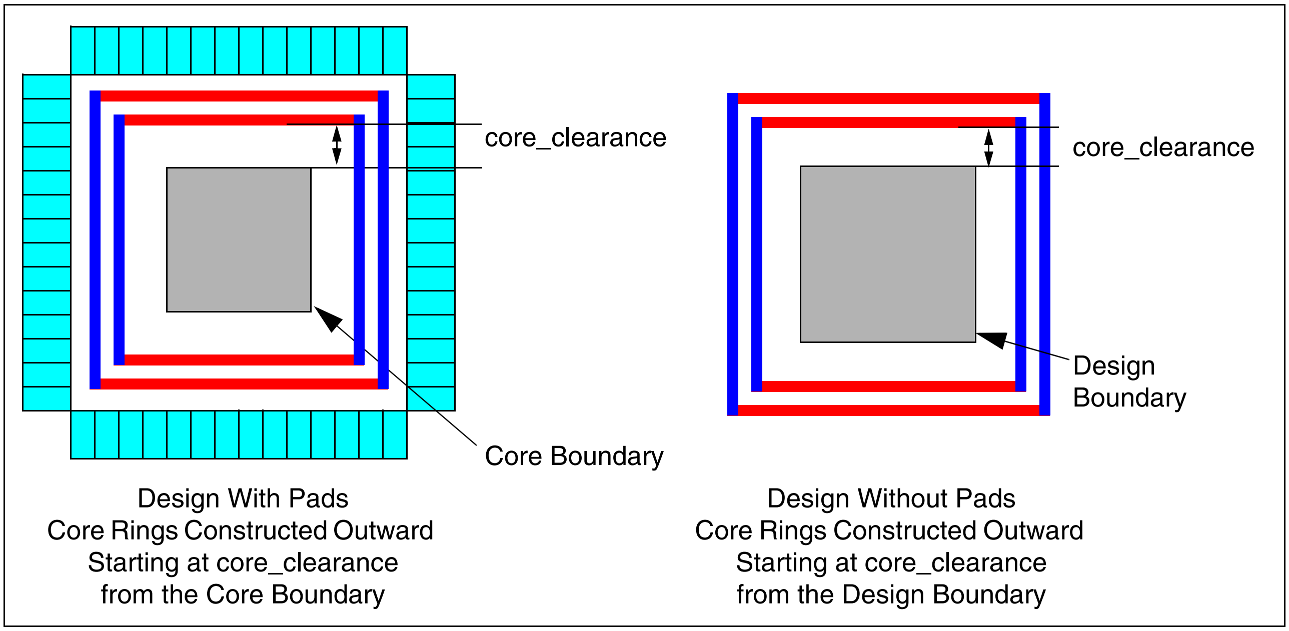
Design Without Pads
Core Ring Constructed Outward
Starting at minClearance
from the Design Boundary

The following example adds core rings for `VDD` and `VSS` on adjacent layers, `met2` and `met3`. If pads are present, the core rings are centered between the pads and the core. If pads do not exist, the core rings are constructed starting from `minClearance` outward from the design boundary.

```
proute_core_ring -layers {met2 met3} -net_width 8 -net_clearance 2 -nets {VDD VSS}
```

With -core_clearance

If the design has peripheral pads, the core rings are created outward starting at `core_clearance` from the core boundary or, if peripheral pads do not exist, the core rings are created outward starting at `core_clearance` from the design bounds.

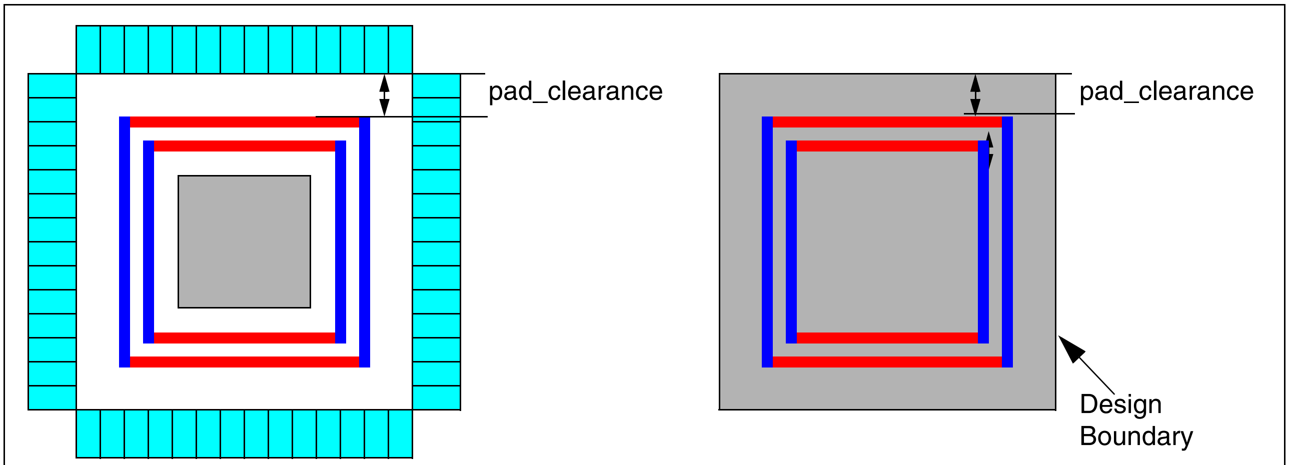


The following example adds core rings for `VDD` and `VSS` on adjacent layers, `met2` and `met3`. If pads are present, the core rings are constructed outward starting at `core_clearance` from the core boundary. If pads do not exist, the core rings are constructed outward starting at `core_clearance` from the design boundary.

```
proute_core_ring -layers {met2 met3} -net_width 8 -net_clearance 2 -nets {VDD VSS}
-core_clearance 0.6
```

With -pad_clearance

If the design has peripheral pads, the core rings are created inward starting at `pad_clearance` from the pads or, if peripheral pads do not exist, the core rings are created inward starting at `pad_clearance` from the design bounds.



Design With Pads
Core Rings Constructed Inward
Starting at pad_clearance
from the Pads

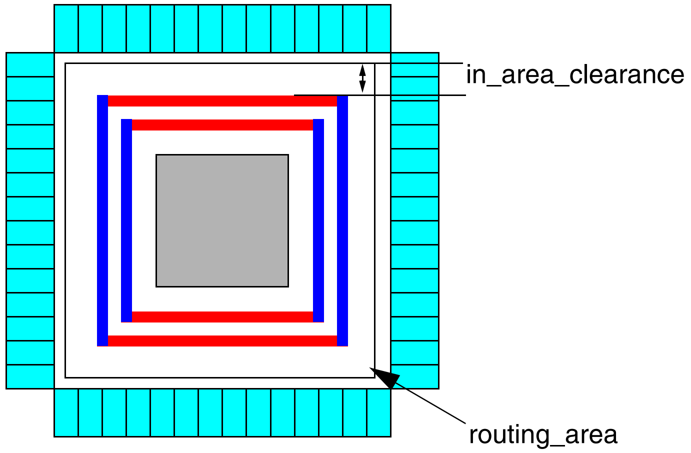
Design Without Pads
Core Rings Constructed Inward
Starting at pad_clearance
from the Design Boundary

The following example adds core rings for `VDD` and `VSS` on adjacent layers, `met2` and `met3`. If pads are present, the core rings are constructed inward starting at `pad_clearance` from the pads. If pads do not exist, the core rings are constructed inward starting at `pad_clearance` from the design boundary.

```
proute_core_ring -layers {met2 met3} -net_width 8 -net_clearance 2 -nets {VDD VSS}
-pad_clearance 0.6
```

With -in_area_clearance

When `-in_area_clearance` is given, the core rings are created inward starting at `in_area_clearance` from the `routing_area`.



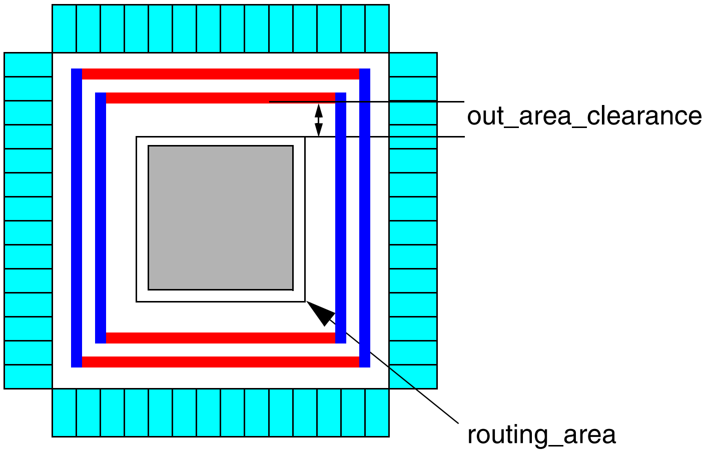
Design with or without Pads
Core Rings Constructed Inward
Starting at in_area_clearance
from the routing_area

The following example adds core rings for `VDD` and `VSS` on adjacent layers, `met2` and `met3`. The core rings are constructed inward starting at `in_area_clearance` from the `routing_area`.

```
proute_core_ring -layers {met2 met3} -net_width 8 -net_clearance 2 -nets {VDD VSS}
-in_area_clearance 0.6 -routing_area [get_window_area]
```

With -out_area_clearance

When `-out_area_clearance` is given, the core rings are created outward starting at `out_area_clearance` from the `routing_area`.



Design with or without Pads
Core Rings Constructed Outward
Starting at out_area_clearance
from the routing_area

The following example adds core rings for `VDD` and `VSS` on adjacent layers, `met2` and `met3`. The core rings are constructed outward starting at `out_area_clearance` from the `routing_area`.

```
proute_core_ring -layers {met2 met3} -net_width 8 -net_clearance 2 -nets {VDD VSS}
-out_area_clearance 0.6 -routing_area { 100 100 550 600 }
```

proute_create_via_array

```
proute_create_via_array
     {-set d_setObj | {-via_defs { s_viaDefName…} -net s_netName}}
     -x_start f_x
     -y_start f_y
     -x_step f_xStep
     -y_step f_yStep
     -x_stop f_x
     -y_stop f_y
     [ -class1 {s_className …} ]
     [ -class2 {s_className …} ]
     [ -class3 {s_className …} ]
     [ -class4 {s_className …} ]
     [ -class5 {s_className …} ]
     [ -stop_at_boundary_of_class1 {s_className…} [ -boundary_clearance1
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_boundary_of_class2 {s_className…} [ -boundary_clearance2
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_boundary_of_class3 {s_className…} [ -boundary_clearance3
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_boundary_of_class4 {s_className…} [ -boundary_clearance4
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_boundary_of_class5 {s_className…} [ -boundary_clearance5
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -prBoundary_clearance {f_clearanceValue | {f_leftValue f_bottomValue
     f_rightValue f_topValue}}]
     [ -stop_at_halo_of_class1 {s_className…} [ -halo_clearance1
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_halo_of_class2 {s_className…} [ -halo_clearance2
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_halo_of_class3 {s_className…} [ -halo_clearance3
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_halo_of_class4 {s_className…} [ -halo_clearance4
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -stop_at_halo_of_class5 {s_className…} [ -halo_clearance5
     {f_clearanceValue | {f_leftValue f_bottomValue f_rightValue
     f_topValue}} ] ]
     [ -use_cellname_as_class [ true | false ] ]
     [ -use_checker [ true | false ] ]
     [ -check_metal_below [ true | false ] ]
     [ -check_metal_above [ true | false ] ]
     [ -color { ColorGray | ColorRed | ColorBlue | ColorYellow } ]
     [ -color_status { ColorMutable | ColorLocked } ]
     [ -use_regions_on_scratch_layer i_scratchLayerID ]
     [ -avoid_regions_on_scratch_layer i_scratchLayerID ]
     [ -undoable [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -threads i_count ]
     [ -threads i_threads ]
     [ -slices i_slices ]
```

Creates an array of selected vias at specific locations.

Specifying Boundary or Halo Clearances

You can prevent vias from extending across block boundaries or halos by using the `stop_at_boundary_of_class` and/or `stop_at_halo_of_class` arguments. Up to five unique lists of classes can be specified, each with a different clearance specification, given by the corresponding `boundary_clearance` or `halo_clearance` argument. For the clearance, if a single value is given, it applies to all four sides of the block or halo. If a list of four values is given, they must be given in the exact order of left, bottom, right, and top. Positive values are measured outward from the block boundary or halo; negative values are measured inward from the block boundary or halo.

Arguments

`-avoid_regions_on_scratch_layer` *`i_scratchLayerID`*

If `true`, prevents the creation of vias inside the regions defined by the shapes on the given scratch layer.

To create shapes on a scratch layer, refer to "geom_add_shape" on page 316.

Default: `false` (all locations are valid)

`-boundary_clearance1 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class1` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-boundary_clearance2 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class2` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-boundary_clearance3 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class3` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-boundary_clearance4 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class4` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-boundary_clearance5 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class5` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-check_metal_above [ true | false ] ]`

If `true`, prevents via from being created if not covered by metal above.

Default: `false`

`-check_metal_below [ true | false ] ]`

If `true`, prevents via from being created if not covered by metal below.

Default: `false`

`-class1 `**`{`***`s_className…`***`}`**

Specifies a list of classes of macro blocks that belong to class1 for “of_class1” arguments.

`-class2` **{***`s_className…`***}**

Specifies a list of classes of macro blocks that belong to class2 for “of_class2” arguments.

`-class3` **{***`s_className…`***}**

Specifies a list of classes of macro blocks that belong to class3 for “of_class3” arguments.

`-class4` **{***`s_className…`***}**

Specifies a list of classes of macro blocks that belong to class4 for “of_class4” arguments.

`-class5` **{***`s_className…`***}**

Specifies a list of classes of macro blocks that belong to class5 for “of_class5” arguments.

`-color {ColorGray | ColorRed | ColorBlue | ColorYellow}`

Mark power shapes with this color.

Default: `ColorGray`

`-color_status {ColorMutable | ColorLocked}`

Mark power shapes with this color status.

Default: `ColorMutable`

`-halo_clearance1 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for the `stop_at_halo_of_class1` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-halo_clearance2 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for the `stop_at_halo_of_class2` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-halo_clearance3 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for the `stop_at_halo_of_class3` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-halo_clearance4 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for the `stop_at_halo_of_class4` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-halo_clearance5 {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for the `stop_at_halo_of_class5` argument. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances. If not specified, zero spacing applies.

`-net `*`s_netName`*

Specifies the name of the net to use for vias created from the viaDefs.

`-prBoundary_clearance {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the extra allowed clearance from the design prBoundary. For an explanation of the argument value, refer to Specifying Boundary or Halo Clearances.

`-set `*`d_setObj`*

Set of vias to copy into the array.

`-silent [ true | false ]`

When `true`, outputs no messages. When `false`, all message types are output.

Default: `false`

`-slices `*`i_slices`*

Specifies the number of region slices in each direction for connecting stripe sections.

Default: `10`

`-stop_at_boundary_of_class1` **{***`s_className`***…}**

Prevents inserting objects inside the boundary of macro blocks that belong to a class in this list. If this list is empty, then classes from class1 are used.

`-stop_at_boundary_of_class2` **{***`s_className`***…}**

Prevents inserting objects inside the boundary of macro blocks that belong to a class in this list. If this list is empty, then classes from class2 are used.

`-stop_at_boundary_of_class3` **{***`s_className`***…}**

Prevents inserting objects inside the boundary of macro blocks that belong to a class in this list. If this list is empty, then classes from class3 are used.

`-stop_at_boundary_of_class4` **{***`s_className`***…}**

Prevents inserting objects inside the boundary of macro blocks that belong to a class in this list. If this list is empty, then classes from class4 are used.

`-stop_at_boundary_of_class5` **{***`s_className`***…}**

Prevents inserting objects inside the boundary of macro blocks that belong to a class in this list. If this list is empty, then classes from class5 are used.

`-stop_at_halo_of_class1` **{***`s_className`***…}**

Prevents inserting objects inside the halo of macro blocks that belong to a class in this list. If this list is empty, then classes from class1 are used.

`-stop_at_rings_of_class2` **{***`s_className`***…}**

Prevents inserting objects inside the halo of macro blocks that belong to a class in this list. If this list is empty, then classes from class2 are used.

`-stop_at_rings_of_class3` **{***`s_className`***…}**

Prevents inserting objects inside the halo of macro blocks that belong to a class in this list. If this list is empty, then classes from class3 are used.

`-stop_at_rings_of_class4` **{***`s_className`***…}**

Prevents inserting objects inside the halo of macro blocks that belong to a class in this list. If this list is empty, then classes from class4 are used.

`-stop_at_rings_of_class5` **{***`s_className`***…}**

Prevents inserting objects inside the halo of macro blocks that belong to a class in this list. If this list is empty, then classes from class5 are used.

`-threads` *`i_count`*

Specifies the number of threads or processors to use for checking. By default, if multi-threading has been enabled, the session threads are used, otherwise, a single processor is used.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel for checking.

Default: `1`

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

`-use_cellname_as_class [ true | false ]`

Allows specification of cellnames instead of class names for the `*of_class*` arguments.

`-use_checker [ true | false ]`

If `true`, run the checker on the new vias.

Default: `true`

`-use_regions_on_scratch_layer` *`i_scratchLayerID`*

If `true`, creates vias only inside the regions defined by the shapes on the given scratch layer.

To create shapes on a scratch layer, refer to “geom_add_shape” on page 316.

Default: `false` (all locations are valid)

`-via_defs` **{***s_viaDefName…***}**

Names of viaDefs for vias to use in the array.

`-x_start` *f_x* The x-coordinate for the first copy of selected objects.

`-x_step` *f_xStep* The x-distance between adjacent array locations.

`-x_stop` *f_x* The x-coordinate limit for copies of selected objects.

`-y_start` *f_y* The y-coordinate for the first copy of selected objects.

`-y_step` *f_yStep* The y-distance between adjacent array locations.

`-y_stop` *f_y* The y-coordinate limit for copies of selected objects.

Examples

The following example creates a 2 x 3 via array using the via definition for `VIA_12`, starting at x- and y-coordinates {100 200} with x-steps of 20 and y-steps of 30, and no vias added past {130 270}.

```
proute_create_via_array -via_defs {STACKEDVIA_12 STACKEDVIA_23 STACKEDVIA_34} \
 -x_start 100 -y_start 200 -x_step 20 -y_step 30 \
 -x_stop 130 -y_stop 270 \
 -net {VDD}
```

The following example creates a three via layer stacked via array, using the via definitions for `STACKEDVIA_12`, `STACKEDVIA_23`, and `STACKEDVIA_34`, starting at the x- and y-coordinates of {100 2200}, with x-steps of 200 and y-steps of 30, and no vias added past {340 2520}.

```
proute_create_via_array -via_defs {STACKEDVIA_12 STACKEDVIA_23 STACKEDVIA_34} \
 -x_start 100 -y_start 2200 -x_step 200 -y_step 30 \
 -x_stop 340 -y_stop 2520 \
 -net {VDD}
```

proute_pad_ring

```
proute_pad_ring
     -nets {s_netName…}
     [ -layers {s_layerName…} | -pin_layers {s_layerName…} ]
     [ -set d_setObj | -pads {s_instName…} ]
     [ -connect_inst_terms [ true | false ] ]
     [ -edge_pins [ true | false ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…} ]
     [ -max_msg_count i_count ]
     [ -power_only [ true | false ] ]
     [ -rail_pins [ true | false ]
     [ -silent [ true | false ] ]
     [ -undoable [ true | false ] ]
     [ -use_checker [ true | false ] ]
```

Routes pad rings for the given nets between pads on the periphery of the design.

Arguments

`-connect_inst_terms [ true | false ]`

If `true`, adds specific connections from pad rails to corresponding instance terminals.

Default: `false`

`-edge_pins [ true | false ]`

Routes to pins on the edge of the pads. By default, edge pins are not routed.

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes {s_purposeName…}`

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-layers {s_layerName …}`

Limits connections to pad pins on the specified layers. If this argument is not given, all routing layers are included.

`-max_msg_count` *`i_count`* Specifies the maximum number of message to output for each message type.

Default: 10

`-nets` **{***`s_netName`***…}** Specifies the nets to route.

`-pads` **{***`s_instName`***…}** Limits connections to the pad instances in the list. By default, all pads are connected.

`-pin_layers` **{***`s_layerName`* **…}**

Limits connections to pad pins on the specified layers. If this argument is not given, all routing layers are included.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-rail_pins [ true | false ]`

Routes to rail pins on the pads. This is the default behavior. Set this argument to `false` to prevent the rail pins from being routed.

`-set` *`d_setObj`* Limits connections to the pad instances in the given set. By default, all pads are connected.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

`-use_checker [ true | false ]`

If `true`, runs the checker on new segments. If `false`, no checking is performed.

Default: `true`

Example

The following example adds inter-pad rails between pads around the periphery of the design.

```
proute_pad_ring -layers met3 -nets {VDD VSS}
```

The following example adds rails between edge pins and not rail pins.

```
proute_pad_ring -layers met3 -nets {VDD VSS} -edge_pins -rail_pins false
```

proute_pin_to_trunk

```
proute_pin_to_trunk
     -min_trunk_width f_userunit
     -nets {s_netName…}
     [ -set d_setObj | -instances {s_instName…} ]
     [ -connect_top_level_pins [ true | false ] ]
     [ -layers {s_layerName…} | -pin_layers {s_layerName…} ]
     [ -pin_set d_setObj ]
     [ -trunk_layer s_layerName | -trunk_layers {s_layerName…} ]
     [ -trunk_set d_setObj ]
     [ -allow_violations [ true | false ] ]
     [ -max_target_shapes i_count ]
     [ -guides_only [ true | false ] ]
     [ -ignore_routing_grid [ true | false ]
     [ -noGuides [ true | false ] ]
     [ -must_connect [ true | false ] ]
     [ -one_pin_source [ true | false ] ]
     [ -power_only [ true | false ] ]
     [ -route_spec s_groupName
     | -max_wire_width f_userunit -min_wire_width f_userunit ]
     [ -sig_route_type [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -snap_to_pin_center [ true | false ] ]
     [ -zero_extents [ true | false ] ]
     [ -undoable [ true | false ] ]
```

Adds connections for power pins on macro blocks and power pads to existing rings and rails.

Arguments

`-allow_violations [ true | false ] ]`

When `true`, allows connections that result in violations.

When `false` (default), connections that cause violations are removed.

`-connect_top_level_pins [ true | false ]`

Specifies whether to add connections for top-level pins.

Default: `false`

`-guides_only [ true | false ]`

If `true`, creates guides for connections but no routing. If `false`, routes connections.

Default: `false`

`-ignore_routing_grid [ true | false ]`

If `false`, follows the routing grid, otherwise (`true`), follows the manufacturing grid.

Default: `true`

`-instances` **{***`s_instName…`***}**

Limits connections to the instances in the list. By default, all blocks are connected.

`-layers` **{***`s_layerName …`***}**

Limits the connections to pins on the given layers. If this argument is not given, all routing layers are included.

`-max_target_shapes` *`i_count`*

Specifies the maximum number of nearest trunk shapes to search for.

Default: 100

`-max_wire_width` *`f_userunit`*

Specifies the maximum wire width for a wire to be routed from a pin to a trunk. If a pin width is greater than `max_wire_width`, then `max_wire_width` will be used for the pin connection.

`-min_trunk_width` *`f_userunit`*

Specifies the minimum trunk width for the trunk to be connected to a pin.

`-min_wire_width` *`f_userunit`*

Specifies the minimum wire width for a wire to be routed from a pin to a trunk. If a pin width is less than `min_wire_width`, the pin is ignored and no connection is made.

`-must_connect [ true | false ]`

When `true`, treats pins of the same terminal as must-connect. When `false`, pins of the same terminal are treated as weak-connect.

Default: `false`

`-nets` **{***`s_netName`***…}** Specifies the nets to route.

`-noGuides [ true | false ]`

When `true`, `-pin_set`, `-trunk_set` and `-rule_spec` must be given to indicate the pins that must be routed to the specific trunks, using the given set of constraints. For large numbers of known pin-to-trunk connections, this method can run faster than the default method of finding the pins and trunks to connect, creating guides for the connections, then routing.

Default: `false`

`-one_pin_source [ true | false ]`

When `true`, treats all pins as a single source. When `false`, each pin is treated as a separate source.

Default: `false`

`-pin_layers` **{***`s_layerName`* **…}**

Limits the connections to pins on the given layers. If this argument is not given, all routing layers are included.

`-pin_set` *`d_setObj`* Limits connections to the pin shapes in the given set.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-route_spec` *`s_groupName`*

Specifies the route spec, or constraint group, to use for the connections. The `minWidth` of the route spec will be used for all pin-to-trunk connections.

By default, if neither `-route_spec` nor `-min_wire_width -max_wire_width` is given, pin-to-trunk connections will be the width of the pin.

`-set` *`d_setObj`* Limits the connections to the instances in the set. By default, all blocks are connected.

`-sig_route_type [ true | false ]`

Sets the `routeType` property for the new pin-to-trunk routes to `PowerTapRouteType` (if `false`/default) or `UnknownRouteType` (if `true`).

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

`-snap_to_pin_center [ true | false ]`

Determines whether connections should be snapped to the centers of pins where possible.

Default: `false`

`-trunk_layer` *`s_layerName`*

Restricts pin-to-trunk connections to the given trunk layer. By default, trunks on all layers are connected.

`-trunk_layers` **{***`s_layerName`***…}**

Restricts pin-to-trunk connections to the given trunk layers. By default, trunks on all layers are connected.

`-trunk_set` *`d_setObj`* Limits connections to the trunk shapes in the given set.

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

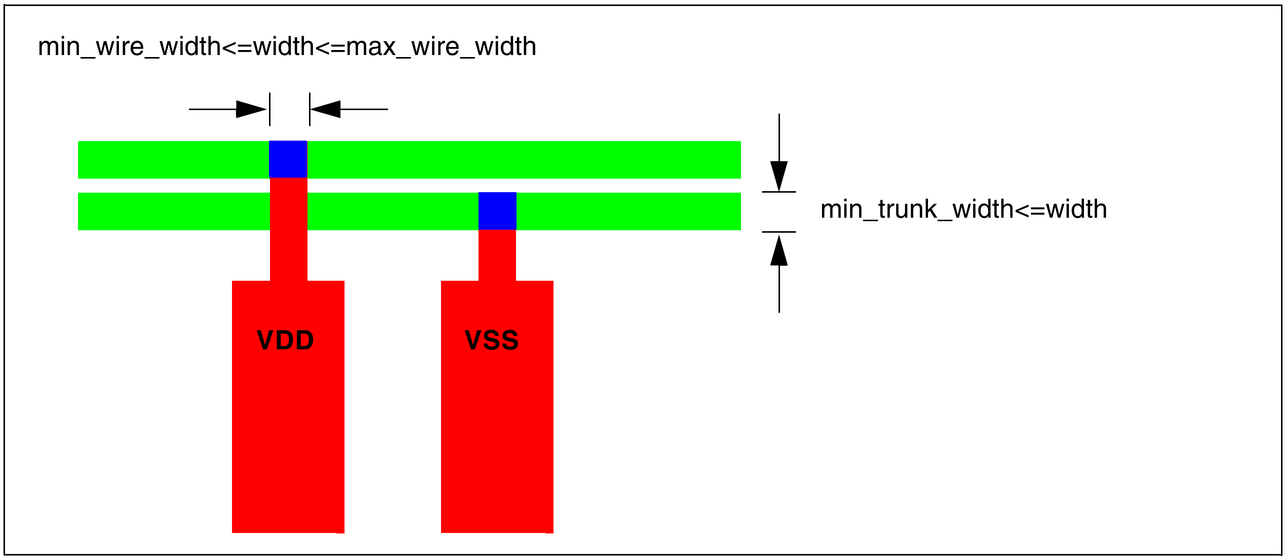
`-zero_extents [ true | false ]`

When set to `true`, allows zero-extents wires to connect to pins and trunks. Default: `false`

Example

The following example adds all pin-to-trunk connections for VDD and VSS.

```
proute_pin_to_trunk -nets {VDD VSS} -min_wire_width 4 -max_wire_width 8
-min_trunk_width 7
```



The following command connects only pins of the instances in the selected set. First, you select the instances for the pins to connect, then issue the command.

```
proute_pin_to_trunk -nets {VDD VSS} -min_wire_width 4 -max_wire_width 8
-min_trunk_width 7 -set [get_selection_set]
```

proute_row_straps

```
proute_row_straps
     -layers {s_layerName…}
     -nets {s_netName…}
     -net_width f_userunit
     [ -connect_inst_terms [ true | false ] ]
     [ -dump_rows i_numRows ]
     [ {-extend | -extend_long} [ true | false ] [ -direction
     {horizontal|vertical} ]
      [ -ring_layers {s_layerName…} ]
      [ -partial_overlap [ true | false ] ] ]
     [ -class_applies_to_top_level_rings [ true | false ] ]
     [ -class1 {s_className …} ]
     [ -class2 {s_className …} ]
     [ -class3 {s_className …} ]
     [ -class4 {s_className …} ]
     [ -class5 {s_className …} ]
     [ -core_ring_layers {s_layerName…} ]
     [ -full_overlap [ true | false ] ]
     [ -height f_height ]
     [ -check_for_obstacles [ true | false ] ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…} ]
     [ -jog_straps [ true | false ]
      [ -max_wrong_way_jog_of_class1 f_userunit]
      [ -max_wrong_way_jog_of_class2 f_userunit]
      [ -max_wrong_way_jog_of_class3 f_userunit]
      [ -max_wrong_way_jog_of_class4 f_userunit]
      [ -max_wrong_way_jog_of_class5 f_userunit]
      [ -max_wrong_way_jog f_userunit ]
      [ -max_wrong_way_ring_jog f_userunit ]
      [ -max_full_overlap_jog f_userunit ] ]
     [ -max_depth i_levelOfHierarchy ]
     [ -max_msg_count i_count ]
     [ -max_row_gap f_userunit ]
     [ -offset f_relativeDistance ]
     [ -power_only [ true | false ] ]
     [ -routing_area {f_xlo f_ylo f_xhi f_yhi} ]
     [ -silent [ true | false ] ]
     [ -site_names {s_siteName…}
     [ -undoable [ true | false ] ]
     [ -ignore_blockage_of_class {s_className…} ]
     [ -all_term_types {core_ring | block_ring | stripes | cell_row_straps
      | unknown} ]
     [ -stop_at_boundary_of_class {s_className…}
       [ -boundary_clearance {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class1 {s_className…}
       [ -boundary_clearance1 {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
     [ -stop_at_boundary_of_class2 {s_className…}
```

```
    [ -boundary_clearance2 {f_clearanceValue
      | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
  [ -stop_at_boundary_of_class3 {s_className…}
    [ -boundary_clearance3 {f_clearanceValue
      | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
  [ -stop_at_boundary_of_class4 {s_className…}
    [ -boundary_clearance4 {f_clearanceValue
      | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
  [ -stop_at_boundary_of_class5 {s_className…}
    [ -boundary_clearance5 {f_clearanceValue
      | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
  [ -incomplete_rings [ true | false ] ]
  [ -stop_at_rings_of_class1 {s_className…} [ -ring_layers1 {s_layerName…} ] ]
  [ -incomplete_rings1 [ true | false ] ]
  [ -stop_at_rings_of_class2 {s_className…} [ -ring_layers2 {s_layerName…} ] ]
  [ -incomplete_rings2 [ true | false ] ]
  [ -stop_at_rings_of_class3 {s_className…} [ -ring_layers3 {s_layerName…} ] ]
  [ -incomplete_rings3 [ true | false ] ]
  [ -stop_at_rings_of_class4 {s_className…} [ -ring_layers4 {s_layerName…} ] ]
  [ -incomplete_rings4 [ true | false ] ]
  [ -stop_at_rings_of_class5 {s_className…} [ -ring_layers5 {s_layerName…} ] ]
  [ -incomplete_rings5 [ true | false ] ]
  [ -trim_at_row_ends [ true | false ] ]
  [ -use_cellname_as_class [ true | false ] ]
  [ -ring_extension_upper_layer f_userunit ]
  [ -ring_extension_lower_layer f_userunit ]
```

Adds row straps for the given layers. With the same functionality as proute_cell_row, this command is designed for faster processing, but requires that rows be defined and placement of the straps with respect to those rows must be carefully specified using `-net_width` and `-offset`.

The command will fail under these circumstances:

- The row strap does not connect to power pins. In cases like this, use `-connect_inst_terms` to add specific connections between the row straps and corresponding instance terminals.
- There is an obstacle or blockage. Use `-jog_straps`, optionally with `-max_wrong_way_jog`, `-max_wrong_way_ring_jog`, and/or `-max_full_overlap_jog`, to set guidelines for the amount of jogging permitted to avoid obstacles when routing the row straps.

Note: To add interlayer connections for the straps, use proute_via_insertion.

Arguments

`-all_term_types {core_ring | block_ring | stripes | cell_row_straps | unknown}`

Treats the given classes of macro block terminals on specified ring layers as this power type.

`-boundary_clearance {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance1 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class1` argument . For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance2 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_class2` argument . For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance3 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class3` . For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance4 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class4` . For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-boundary_clearance5 {`*`f_clearanceValue`*` | `**`{`***`f_leftValue f_bottomValue f_rightValue f_topValue`***`}`**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class5`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

`-check_for_obstacles [ true | false ]`

When `true`, checks each row for obstacles. When `false`, no checking is done; this is useful to speed up processing when it is known that no obstacles are present, such as on an empty layer.

Default: `true`

`-class_applies_to_top_level_rings [ true | false ]`

Stops strap extensions at top-level block rings around blocks whose class is specified in the “of class” arguments.

Default: `false`

`-class1 `**`{`**`*s_className…*`**`}`**

Specifies a list of classes of macro blocks that belong to class1 for “of_class1” arguments.

`-class2 `**`{`**`*s_className…*`**`}`**

Specifies a list of classes of macro blocks that belong to class2 for “of_class2” arguments.

`-class3 `**`{`**`*s_className…*`**`}`**

Specifies a list of classes of macro blocks that belong to class3 for “of_class3” arguments.

`-class4 `**`{`**`*s_className…*`**`}`**

Specifies a list of classes of macro blocks that belong to class4 for “of_class4” arguments.

`-class5 `**`{`**`*s_className…*`**`}`**

Specifies a list of classes of macro blocks that belong to class5 for “of_class5” arguments.

`-connect_inst_terms [ true | false ]`

If `true`, adds specific connections from straps to corresponding instance terminals.

Default: `false`

`-core_ring_layers` **{***`s_layerName…`***}** `| all`

Limits the layers of core-ring segments for determining the core ring bounds. If this is not specified, then `ring_layers` applies. If `ring_layers` is not specified, then all core-ring segments apply.

`-direction {horizontal | vertical}`

Specifies the direction for the cell row straps. By default, the preferred routing direction for the layer is used.

`-dump_rows` *`i_numRows`*

If non-zero, dumps information on the given number of rows per site without performing any routing.

Default: 0

`{-extend | -extend_long}[ true | false ]`

If `-extend` is `true`, cell row straps are extended to the nearest power rail.

If `-extend_long` is `true`, cell row straps are extended to the nearest power rail. Then power router attempts to extend any unextended cell row to the core ring.

Default: `false`

`-full_overlap [ true | false ]`

When `true`, straps must fully overlap same net shapes. If a straight strap does not fully overlap an existing same net shape and `-jog_straps` is specified, the strap can be jogged to achieve full overlap; otherwise, the strap is not added.

Default: `false`

`-height` *`f_height`*

Routes only rows with sites of this height.

`-ignore_blockage_of_class` **{***`s_className…`***}**

Ignores the blockages in macro blocks that belong to a class in this list.

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes` **{***`s_purposeName…`***}**

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-incomplete_rings [ true | false ]`

When `true`, top-level block rings are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Note: If there are incomplete ring segments on concave U-turns surrounding a block, row straps cannot be placed in the U-turn area.

Default: `false`

`-incomplete_rings1 [ true | false ]`

When `true`, internal block rings of class 1 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Default: `false`

`-incomplete_rings2 [ true | false ]`

When `true`, internal block rings of class 2are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Default: `false`

`-incomplete_rings3 [ true | false ]`

When `true`, internal block rings of class 3 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Default: `false`

`-incomplete_rings4 [ true | false ]`

When `true`, internal block rings of class 4 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Default: `false`

`-incomplete_rings5 [ true | false ]`

When `true`, internal block rings of class 5 are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the row straps, that is specific to this condition.

Default: `false`

`-max_full_overlap_jog` *`f_userunit`*

(Applies only when `-full_overlap true` and `-jog_straps true`) Specifies the maximum amount of jog for straps to prefer full overlap when stopping at block rings.

Default: `net_width` + `minSpacing` for the net

`-max_wrong_way_jog` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging around obstacles that is permitted in the wrong-way direction.

Default: `net_width`

`-max_wrong_way_jog_of_class1` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class1.

Default: `net_width`

`-max_wrong_way_jog_of_class2` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class2.

Default: `net_width`

`-max_wrong_way_jog_of_class3` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class3.

Default: `net_width`

`-max_wrong_way_jog_of_class4` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class4.

Default: `net_width`

`-max_wrong_way_jog_of_class5` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class5.

Default: `net_width`

`-max_wrong_way_ring_jog` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging around obstacles that is permitted in the wrong-way direction when stopping at rings.

Default: `net_width`

`-net_width` *`f_userunit`* Specifies the width for the row straps in user units.

`-nets` **{***`s_netName…`***}** Specifies the nets to route.

`-offset` *`f_relativeDistance`*

For horizontal rows, specifies the y distance of the first net’s strap relative to the bottom side of a zero-orientation row. For vertical rows, specifies the x distance of the first net’s strap relative to the left side of a zero-orientation row. Default: 0

`-partial_overlap [ true | false ]`

If `true`, allows strap extension to partially overlap with other shapes on the same net and layer.

Default: `false`

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-ring_extension_lower_layer` *`f_userunit`*

Distance to extend stripe past inner edge of a ring segment on the metal layer below stripe.

Deafult: `0`

`-ring_extension_upper_layer` *`f_userunit`*

Distance to extend stripe past inner edge of a ring segment on the metal layer above stripe.

Deafult: `0`

`-ring_layers` **{***`s_layerName`*…**}**

(Applies only when -`extend` or -`extend_long` is given) Specifies the layers on which to extend cell rows. If not specified, all layers apply.

`-ring_layers1` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class1`. Default: all layers

`-ring_layers2` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class2`. Default: all layers

`-ring_layers3` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class3`. Default: all layers

`-ring_layers4` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class4`. Default: all layers

`-ring_layers5` **{***`s_layerName`*…**}**

Specifies the ring layers for `-stop_at_rings_of_class5`. Default: all layers

`-routing_area` **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the outer bounds for the area of the standard cells to add straps to. Connects only pins that intersect the area and extends cell row straps to the area bounds.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

`-site_names` **{***s_siteName…***}**

Routes only rows with sites of these names.

`-stop_at_boundary_of_class` **{***s_className…***}**

Stops stripe sections at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class1` **{***s_className…***}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance1` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class2` **{***s_className…***}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance2` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class3` **{***s_className…***}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance3` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class4` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance4` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class5` **{***`s_className`***…}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance5` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_rings_of_class1` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class2` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class3` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class4` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-stop_at_rings_of_class5` **{***`s_className`***…}**

Stops strap extensions at the internal block rings of macro blocks that belong to a class in the given list.

`-trim_at_row_ends [ true | false ]`

When `true`, trim main row strap back from obstacles near row ends. When `false`, entire strap fails when such obstacles are present.

Default: `false`

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

`-use_cellname_as_class [ true | false ]`

Allows specification of cellnames instead of class names for the `*of_class*` arguments.

Example

The following command adds row straps for the VSS and VDD nets on Metal1 and extends cell rows to the nearest power rails.

```
proute_row_straps -nets {VSS VDD} -layers Metal1 -net_width 0.36 -offset 0.18
-extend
```

Related Information

Tcl Command	proute_cell_row

proute_stripes

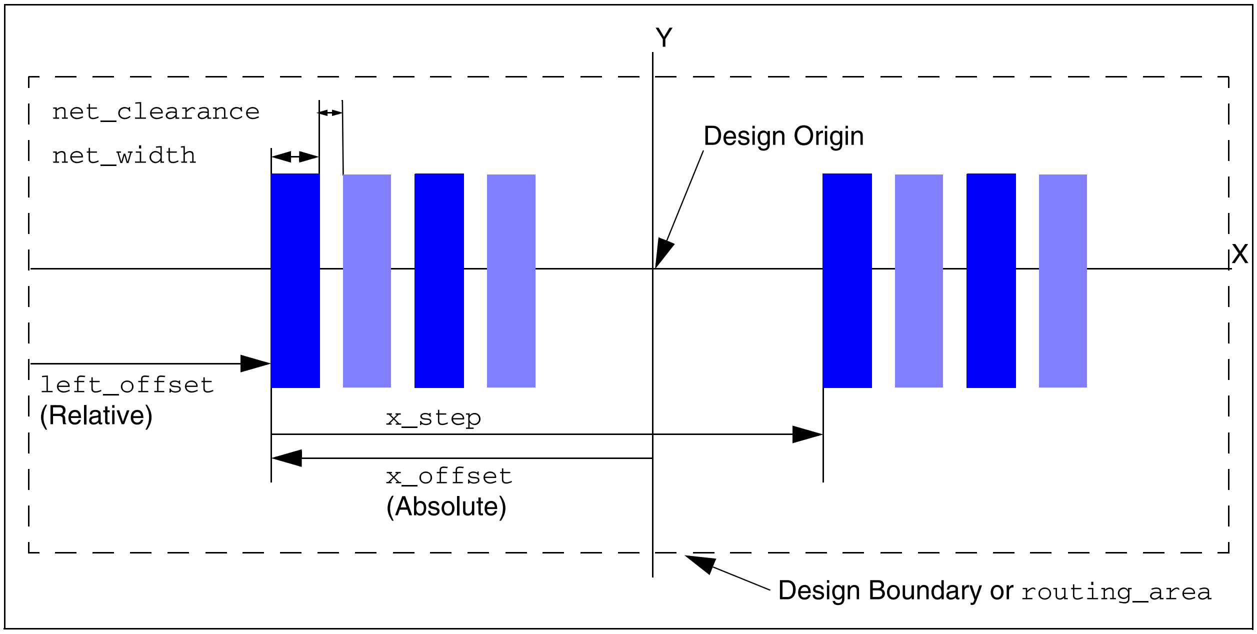
```
proute_stripes
     -layers {s_layerName…}
     -nets {s_netName…}
     -net_width f_userunit
     [ -set d_setObj | -instances {s_instName…} ]
     [ -blockage {f_xlo f_ylo f_xhi f_yhi} ]
     [ -direction {horizontal|vertical} ]
     [ -max_length f_userunit ]
     [ -max_width f_userunit ]
     [ -min_length f_userunit ]
     [ -min_length_to_rings f_userunit ]
     [ -net_clearance f_userunit ]
     [ -pin_clearance f_userunit ]
     [ -observe_rectilinear_prBoundary [ true | false ] ]
     [ -ignore_same_net_shape [ true | false ] ]
     [ -routing_area
       {{f_xlo f_ylo f_xhi f_yhi} |{f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 … f_xn f_yn}}
       [ -snap_to_core_ring [ true | false ] ]
      | -use_design_boundary [ true | false ]
      | -use_regions_on_scratch_layer i_scratchID ] ]
     [ -x_step f_userunit | -y_step f_userunit ]
     [ -bottom_offset f_userunit | -y_offset f_userunit ]
     [ -left_offset f_userunit | -x_offset f_userunit ]
     [ -all_term_types {core_ring | block_ring | stripes | cell_row_straps
      | unknown} ]
     [ -centerline [ true | false ] ]
     [ -full_overlap [ true | false ] ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…}]
     [ -interior_stripes [ true | false ] ]
     [ -interior_stripes_of_class1 [ true | false ] ]
     [ -interior_stripes_of_class2 [ true | false ] ]
     [ -interior_stripes_of_class3 [ true | false ] ]
     [ -interior_stripes_of_class4 [ true | false ] ]
     [ -interior_stripes_of_class5 [ true | false ] ]
     [ -jog_stripes [ true | false ]
      [ -max_wrong_way_jog_of_class1 f_userunit ]
      [ -max_wrong_way_jog_of_class2 f_userunit ]
      [ -max_wrong_way_jog_of_class3 f_userunit ]
      [ -max_wrong_way_jog_of_class4 f_userunit ]
      [ -max_wrong_way_jog_of_class5 f_userunit ]
      [ -max_wrong_way_jog f_length ] ]
      [ -max_wrong_way_ring_jog f_length ]
      [ -max_full_overlap_jog f_length ] ]
     [ -section_length f_userunit -section_step f_stepLength ]
     [ -power_only [ true | false ] ]
     [ -undoable [ true | false ] ]
     [ -stop_at_boundary_of_class {s_className…}
       [ -boundary_clearance {f_clearanceValue
         | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
```

```
[ -stop_at_boundary_of_class1 {s_className…}
  [ -boundary_clearance1 {f_clearanceValue
    | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
[ -stop_at_boundary_of_class2 {s_className…}
  [ -boundary_clearance2 {f_clearanceValue
    | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
[ -stop_at_boundary_of_class3 {s_className…}
  [ -boundary_clearance3 {f_clearanceValue
    | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
[ -stop_at_boundary_of_class4 {s_className…}
  [ -boundary_clearance4 {f_clearanceValue
    | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
[ -stop_at_boundary_of_class5 {s_className…}
  [ -boundary_clearance5 {f_clearanceValue
    | {f_leftValue f_bottomValue f_rightValue f_topValue}} ] ]
[ -ignore_blockage_of_class {s_className…}]
[ -core_ring_layers {s_layerName …} | all]
[ -stop_at_rings [ true | false ] [ -ring_layers {s_layerName…} ]
 [ -incomplete_rings [ true | false ] ]
 | -class_applies_to_top_level_rings [ true | false ] ]
[ -class1 {s_className …} ]
[ -class2 {s_className …} ]
[ -class3 {s_className …} ]
[ -class4 {s_className …} ]
[ -class5 {s_className …} ]
[ -stop_at_rings_of_class1 {s_className…} [ -ring_layers1 {s_layerName…} ]
 [ -incomplete_rings1 [ true | false ] ]
 [ -ring_pin_prop_name1 s_propName -ring_pin_prop_value1 s_propValue ] ]
[ -stop_at_rings_of_class2 {s_className…} [ -ring_layers2 {s_layerName…} ]
 [ -incomplete_rings2 [ true | false ] ]
 [ -ring_pin_prop_name2 s_propName -ring_pin_prop_value2 s_propValue ] ]
[ -stop_at_rings_of_class3 {s_className…} [ -ring_layers3 {s_layerName…} ]
 [ -incomplete_rings3 [ true | false ] ]
 [ -ring_pin_prop_name3 s_propName -ring_pin_prop_value3 s_propValue ] ]
[ -stop_at_rings_of_class4 {s_className…} [ -ring_layers4 {s_layerName…} ]
 [ -incomplete_rings4 [ true | false ] ]
 [ -ring_pin_prop_name4 s_propName -ring_pin_prop_value4 s_propValue ] ]
[ -stop_at_rings_of_class5 {s_className…} [ -ring_layers5 {s_layerName…} ]
 [ -incomplete_rings5 [ true | false ] ]
 [ -ring_pin_prop_name5 s_propName -ring_pin_prop_value5 s_propValue ] ]
[ -silent [ true | false ] ]
[ -use_cellname_as_class [ true | false ] ]
[ -ring_extension_upper_layer f_userunit ]
[ -ring_extension_lower_layer f_userunit ]
[ -use_wsp [ true | false ] ]
[ -wsp_pullback_left f_userunit]
[ -wsp_pullback_right f_userunit]
[ -wsp_pullback_top f_userunit]
[ -wsp_pullback_bottom f_userunit]
[ -ignore_boundary_tracks [ true | false ] ]
```

Adds net stripes at regular intervals. If more than one net is specified, the stripes are added in groups, in the order given, and repeated according to the stepsize.

Two methods are provided for placement of the stripes:

- Absolute Offset from Origin
  - ❑ For horizontal stripes, use `-y_offset` to indicate the y-axis distance from the origin to the lower edge of the first stripe.
  - ❑ For vertical stripes, use `-x_offset` to indicate the x-axis distance from the origin to the left edge of the first stripe.
- Relative Offset from Boundary
  - ❑ For horizontal stripes, use `-bottom_offset` to indicate the y-axis distance from the bottom boundary (either the design's bottom edge or *`f_ylo`* if `-routing_area` is given) to the lower edge of the first stripe.
  - ❑ For vertical stripes, use `-left_offset` to indicate the x-axis distance from the left boundary (either the design's left edge or *`f_xlo`* if `-routing_area` is given) to the left edge of the first stripe.



The power router will not route on blockages. In addition, you can prevent routing on all layers of a given area, using the `-blockage` argument.

Class options let you control stripe truncation at block rings inside cell hierarchy. You can specify up to five macro cell classes and a corresponding list of ring layers for which the truncation applies. If a macro cell has a `ctuPowerRouteClass` property, then the value of that property can be listed in the `*of_class*` options.

Two mutually required arguments, `-section_length` and `-section_step`, let you create tandem stripe sections in a regular pattern.

Note: To add interlayer connections, use proute_via_insertion.

Arguments

`-all_term_types {core_ring | block_ring | stripes | cell_row_straps | unknown}`

Treats the given classes of macro block terminals on specified ring layers as this power type.

`-blockage` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Prevents routing of stripes in this region. This allows you to create a hole in the stripes.

`-bottom_offset` *`f_userunit`*

Specifies the y location of the first horizontal stripe relative to the bottom bounds (entire design or given by `routing_area`).

`-boundary_clearance {`*`f_clearanceValue`*` | `**{***`f_leftValue f_bottomValue f_rightValue f_topValue`***}**`}`

Specifies the clearance for blocks of the class given by `stop_at_boundary_of_class`. For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

```
-boundary_clearance1 {f_clearanceValue | {f_leftValue
f_bottomValue f_rightValue f_topValue}}
-boundary_clearance2 {f_clearanceValue | {f_leftValue
f_bottomValue f_rightValue f_topValue}}
-boundary_clearance3 {f_clearanceValue | {f_leftValue
f_bottomValue f_rightValue f_topValue}}
-boundary_clearance4 {f_clearanceValue | {f_leftValue
f_bottomValue f_rightValue f_topValue}}
-boundary_clearance5 {f_clearanceValue | {f_leftValue
f_bottomValue f_rightValue f_topValue}}
```

Specifies the clearance for blocks of the class given by the `stop_at_boundary_of_classx` argument . For an explanation of the argument value, refer to Specifying Block Boundary Clearances.

```
-centerline [ true | false ]
```

If `true`, measures offsets to the first stripe centerline. If `false`, measures offsets to the left edge for vertical stripes or to the bottom edge for horizontal stripes. Default is `false`.

```
-class_applies_to_top_level_rings [ true | false ]
```

Stops strap extensions at top-level block rings around blocks whose class is specified in the “of class” arguments.

Default: `false`

```
-class1 {s_className…}
-class2 {s_className…}
-class3 {s_className…}
-class4 {s_className…}
-class5 {s_className…}
```

Specifies a list of classes of macro blocks that belong to class*x* for “of_class*x*” arguments.

```
-core_ring_layers {{s_layerName…} | all}
```

By default, stripes are terminated at core rings using all core ring layers to determine the bounds. If `-core_ring_layers` is set, it specifies the core ring layers to use for determining core ring bounds. If this argument is not set and `-ring_layers` is set, then `ring_layers` applies.

`-direction horizontal|vertical`

Specifies the routing direction for the stripes. By default, the preferred routing direction for the given layer is used.

`-full_overlap [ true | false ]`

When `true`, stripes must fully overlap same net shapes. If a straight stripe does not fully overlap an existing same net shape and `-jog_stripes` is specified, the stripe can be jogged to achieve full overlap; otherwise, the stripe is not added.

Default: `false`

`-ignore_blockage_of_class` **{***`s_className`***…}**

Ignores the blockages in macro blocks that belong to a class in this list.

`-ignore_boundary_tracks [ true | false ]`

Restricts from creating the tracks on the boundary of a region.

Default: `false`

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes` **{***`s_purposeName`***…}**

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-ignore_same_net_shape [ true | false ]`

When `true`, ignores pre-existing same net shapes.
Default: `true`

`-incomplete_rings [ true | false ]`

When `true`, top-level block rings are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the stripes, that is specific to this condition.

Note: If there are incomplete ring segments on concave U-turns surrounding a block, stripes cannot be placed in the U-turn area.

Default: `false`

```
-incomplete_rings1 [ true | false ]
-incomplete_rings2 [ true | false ]
-incomplete_rings3 [ true | false ]
-incomplete_rings4 [ true | false ]
-incomplete_rings5 [ true | false ]
```

When `true`, internal block rings of class *x* are assumed to be incomplete (depopulated). This causes an alternate algorithm to be used for adding the stripes, that is specific to this condition.

Default: `false`

```
-interior_stripes [ true | false ]
```

When `true`, allows stripes to be created within the interior of macros.

Default: `true`

```
-interior_stripes_of_class1 [ true | false ]
-interior_stripes_of_class2 [ true | false ]
-interior_stripes_of_class3 [ true | false ]
-interior_stripes_of_class4 [ true | false ]
-interior_stripes_of_class5 [ true | false ]
```

When `true`, allows stripes to be created within the interior of macros of class *x*.

Default: `true`

`-max_length` *`f_userunit`*

Specifies the maximum allowed stripe length.

`-max_width` *`f_userunit`*

Specifies the maximum stripe width (overrides the `maxWidth` rule).

`-max_wrong_way_jog` *`f_length`*

Specifies the maximum distance to jog in the wrong-way direction. Default is equal to the `net_width` value.

`-max_wrong_way_jog_of_class1` *`f_userunit`*
`-max_wrong_way_jog_of_class2` *`f_userunit`*
`-max_wrong_way_jog_of_class3` *`f_userunit`*
`-max_wrong_way_jog_of_class4` *`f_userunit`*
`-max_wrong_way_jog_of_class5` *`f_userunit`*

(Applies only when `-jog_straps true`) Specifies the maximum amount of jogging that is permitted in the wrong-way direction around obstacles associated with macro blocks of class*`x`*.

Default: `net_width`

`-max_wrong_way_ring_jog` *`f_length`*

Specifies the maximum distance to jog in the wrong-way direction when stopping at rings. Default is equal to the `net_width` value.

`-min_length` *`f_userunit`*

Specifies the minimum allowed stripe length.
Default: `minWidth`

`-min_length_to_rings` *`f_userunit`*

Specifies the minimum length for stripes that stop at rings.

Default: `min_length`

`-net_clearance` *`f_userunit`*

Specifies the spacing required between the power net stripes.

`-net_width` *`f_userunit`*

Specifies the total width for a stripe of each net routed. By default, each net stripe is one wire that is `minWidth` wide.

If you specify a net width value that is greater than the `maxWidth` of a given layer, each net stripe on that layer will be composed of multiple wires such that the following are true:

- The wires of a stripe are equal in width.
- The width of the added wire is greater than or equal to `minWidth` of the layer and less than or equal to the `maxWidth` of the layer.
- The sum of the wire widths for each net in a stripe set is equal to the given net width.

Note: Some roundoff error can cause the total width to be slightly greater than specified net width.

`-nets` **{***`s_netName…`***}**

Specifies the nets to route. One or more nets can be given. When more than one net is given, the list order of the nets determines the placement of the stripes, with the first net on the leftmost stripe for vertical stripes, or on the bottommost stripe for horizontal stripes.

`-observe_rectilinear_prBoundary`

If `true`, limits stripes to the design's rectilinear prBoundary.

Default: `false`

`-pin_clearance` *`f_userunit`*

Specifies the clearance required between signal pins and the power nets.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-ring_extension_lower_layer` *`f_userunit`*

Distance to extend stripe past inner edge of a ring segment on the metal layer below stripe.

Deafult: `0`

`-ring_extension_upper_layer` *`f_userunit`*

Distance to extend stripe past inner edge of a ring segment on the metal layer above stripe.

Deafult: `0`

`-ring_layers` **{***`s_layerName`***…}**

Specifies the ring layers for -`stop_at_rings`.

Default: all layers

`-ring_layers1` **{***`s_layerName`***…}**
`-ring_layers2` **{***`s_layerName`***…}**
`-ring_layers3` **{***`s_layerName`***…}**
`-ring_layers4` **{***`s_layerName`***…}**
`-ring_layers5` **{***`s_layerName`***…}**

Specifies the ring layers for `-stop_at_rings_of_class`*`x`*. Default: all layers

`-ring_pin_prop_name1` *`s_propName`*
`-ring_pin_prop_name2` *`s_propName`*
`-ring_pin_prop_name3` *`s_propName`*
`-ring_pin_prop_name4` *`s_propName`*
`-ring_pin_prop_name5` *`s_propName`*

Used with `-stop_at_rings_of_class`*`x`* to restrict rings to those comprised of class *`x`* block pin shapes with this property matching the value given by `-ring_pin_prop_value`*`x`*.

`-ring_pin_prop_value1` *`s_propValue`*
`-ring_pin_prop_value2` *`s_propValue`*
`-ring_pin_prop_value3` *`s_propValue`*
`-ring_pin_prop_value4` *`s_propValue`*
`-ring_pin_prop_value5` *`s_propValue`*

Used with `-stop_at_rings_of_class`*`x`* to restrict rings to those comprised of class *`x`* block pin shapes with the property given by `-ring_pin_prop_name`*`x`* and value given by this argument.

`-routing_area {{`*`f_xlo f_ylo f_xhi f_yhi`*`}` **|** `{`*`f_x1 f_y1 f_x2 f_y2 f_x3 f_y3 … f_xn f_yn`*`}}`

Specifies the outer boundary for the stripes as a rectangular boundary given by the lower left and upper right coordinates or as a rectilinear boundary given by a list of an even number of at least four (4) x-y coordinate pairs. Only stripes that fit entirely inside this area are drawn.

If this argument is not given and core rings exist, the core rings become the bounds for the stripes.

If this argument is not given and the core rings do not exist, the stripes will cover the entire design.

`-section_length` *`f_userunit`*

Breaks each logical stripe into tandem sections of this length.

`-section_step` *`f_stepLength`*

Specifies the distance or pitch between tandem stripe sections.

`-set` *`d_setObj`* Treats the prBoundary of the instances in the set as blockages.

`-silent [ true | false ]`

When `true`, only error messages are output. When `false`, all messages are output.

Default: `false`

`-snap_to_core_ring [ true | false ]`

(Used with `-routing_area`) Snaps stripe sections to the core ring.

`-stop_at_boundary_of_class` **{***`s_className…`***}**

Stops stripe sections at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance` to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_boundary_of_class1` **{***`s_className…`***}**
`-stop_at_boundary_of_class2` **{***`s_className…`***}**
`-stop_at_boundary_of_class3` **{***`s_className…`***}**
`-stop_at_boundary_of_class4` **{***`s_className…`***}**
`-stop_at_boundary_of_class5` **{***`s_className…`***}**

Stops strap extensions at the boundary of macro blocks that belong to a class in this list. Use `-boundary_clearance`*x* to specify the clearance around the boundary, otherwise minimum spacing will be used.

`-stop_at_rings [ true | false ]`

Stops stripes sections at top-level block rings and internal rings of blocks whose class is not specified in the *`of_class`* options.

`-stop_at_rings_of_class1` **{***`s_className`*…**}**
`-stop_at_rings_of_class2` **{***`s_className`*…**}**
`-stop_at_rings_of_class3` **{***`s_className`*…**}**
`-stop_at_rings_of_class4` **{***`s_className`*…**}**
`-stop_at_rings_of_class5` **{***`s_className`*…**}**

Stops stripe sections at the internal block rings of macro blocks that belong to a class in the given list.

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

`-use_cellname_as_class [ true | false ]`

Allows specification of cellnames instead of class names for the `*of_class*` arguments.

`-use_design_boundary [ true | false ]`

(Cannot be used with `-routing_area` or `-use_regions_on_scratch_layer`) When `true`, uses the design boundary as the routing area and ignores any existing core ring bounds.

Default: `false`

`-use_regions_on_scratch_layer` *`i_scratchID`*

(Cannot be used with `-routing_area` or `-use_design_boundary`) When `true`, uses the regions defined by the shapes on the given scratch layer as the routing areas for the stripes.

`-use_wsp [ true | false ]`

Creates stripes on the tracks from wsp regions instead of netwidth and regions provided.

Default: `false`

`-x_offset` *`f_userunit`*
Specifies the x location from the design origin to the first vertical stripe.

`-x_step` *`f_userunit`*
Specifies the x distance between sets of vertical stripes.

`-y_offset` *`f_userunit`*
Specifies the y location from the design origin to the first horizontal stripe.

`-y_step` *`f_userunit`*
Specifies the y distance between sets of horizontal stripes.

`-wsp_pullback_bottom`

Specifies pullback in the bottom direction by the value provided. The pullbacks are applied to vertical layers.

Default: `0`

`-wsp_pullback_left`

Specifies pullback in the left direction by the value provided. The pullbacks are applied to horizontal layers.

Default: `0`

`-wsp_pullback_right`

Specifies pullback in the right direction by the value provided. The pullbacks are applied to horizontal layers.

Default: `0`

`-wsp_pullback_top`

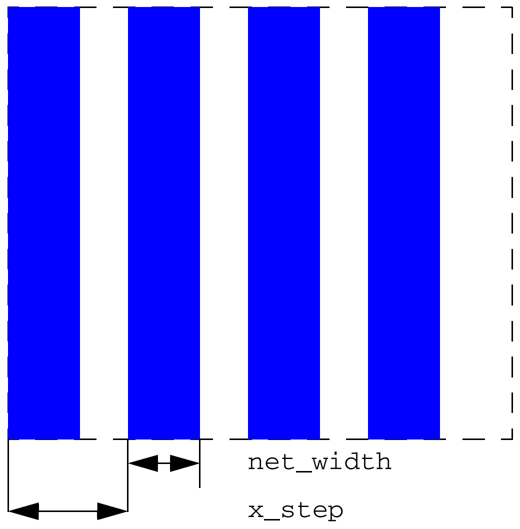
Specifies pullback in the top direction by the value provided. The pullbacks are applied to vertical layers. For example, on `metal3` tracks, `wsp_pullback_top 0.08` means that the track is short by `0.08` from routing region height).

Default: `0`

Example

The following example uses relative addressing for placing stripes in a region. Stripes on layer `M6` are added for the `VDD` net with a width of 8 and spaced 12 apart, left edge-to-left edge. The stripes are vertical because that is the preferred direction for layer M6.

```
proute_stripes -layers M6 -net_width 8 -nets VDD -left_offset 0 -x_step 12
-routing_area {200 300 250 350 }
```

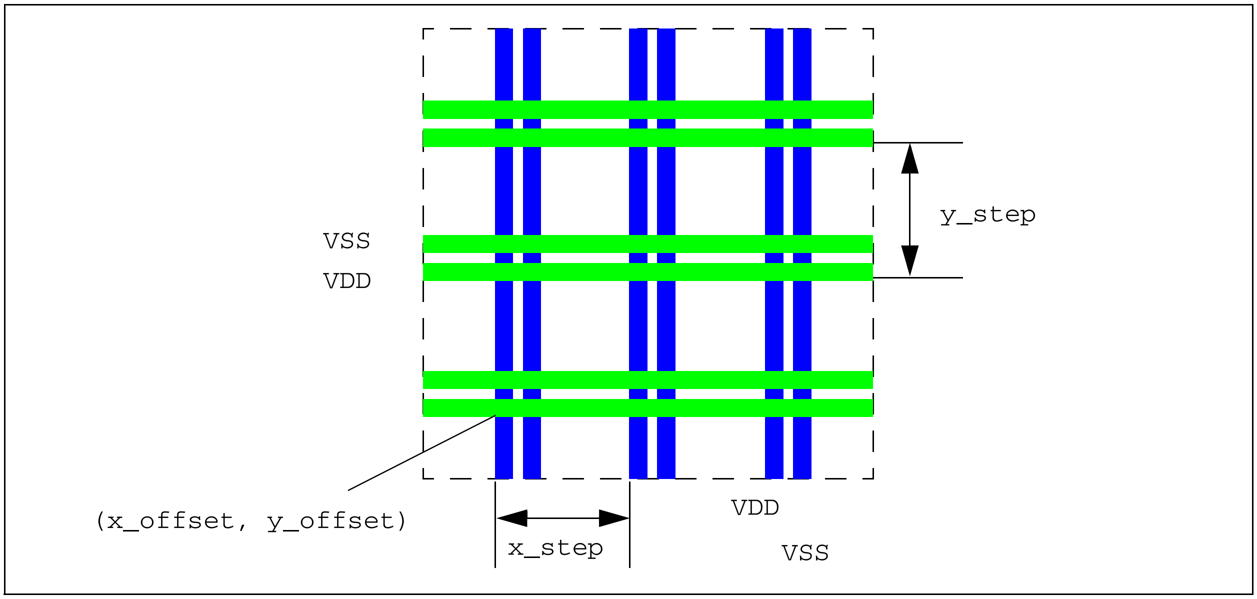


Stripes are created with the left edge of the first stripe on the left edge of the routing area (`-left_offset 0`).

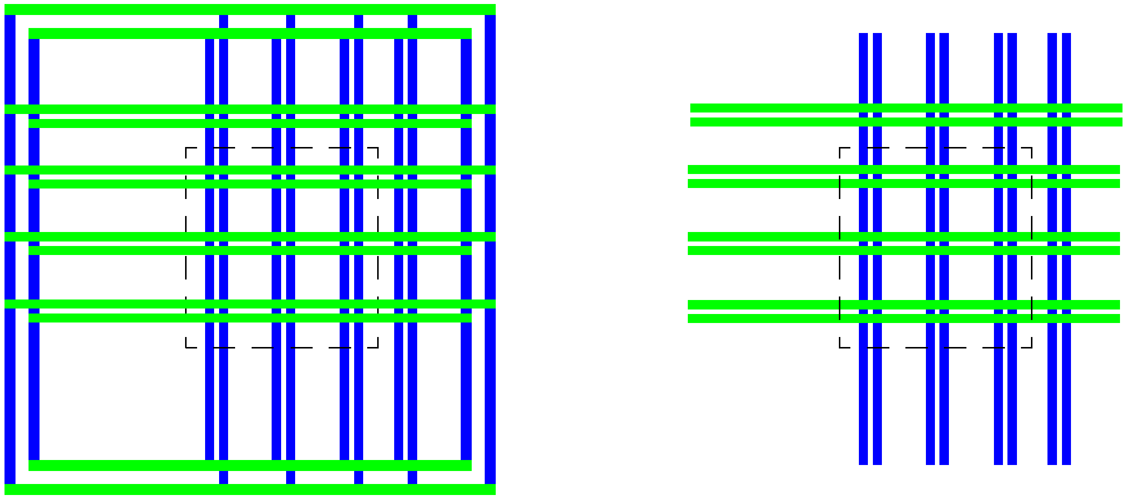
Only four stripes fit entirely in the routing area and are drawn.

The next example creates stripe pairs of VDD and VSS on two layers. `net_clearance` is the spacing between the nets. The stripes are restricted to the `routing_area`, with the position of the bottommost and leftmost stripes given by absolute offsets (`x_offset` and `y_offset`) from the origin.

```
proute_stripes -nets {VDD VSS} -layers {met4 met5} -net_width 8 -net_clearance 3.5
-x_offset 1450 -y_offset 1450 -x_step 100 -y_step 100 -routing_area {1400 1400 1720
1720}
```



If the `-routing_area` argument is not given, the stripes are extended to the core ring, if it exists, or to the bounds of the design if there is no core ring.

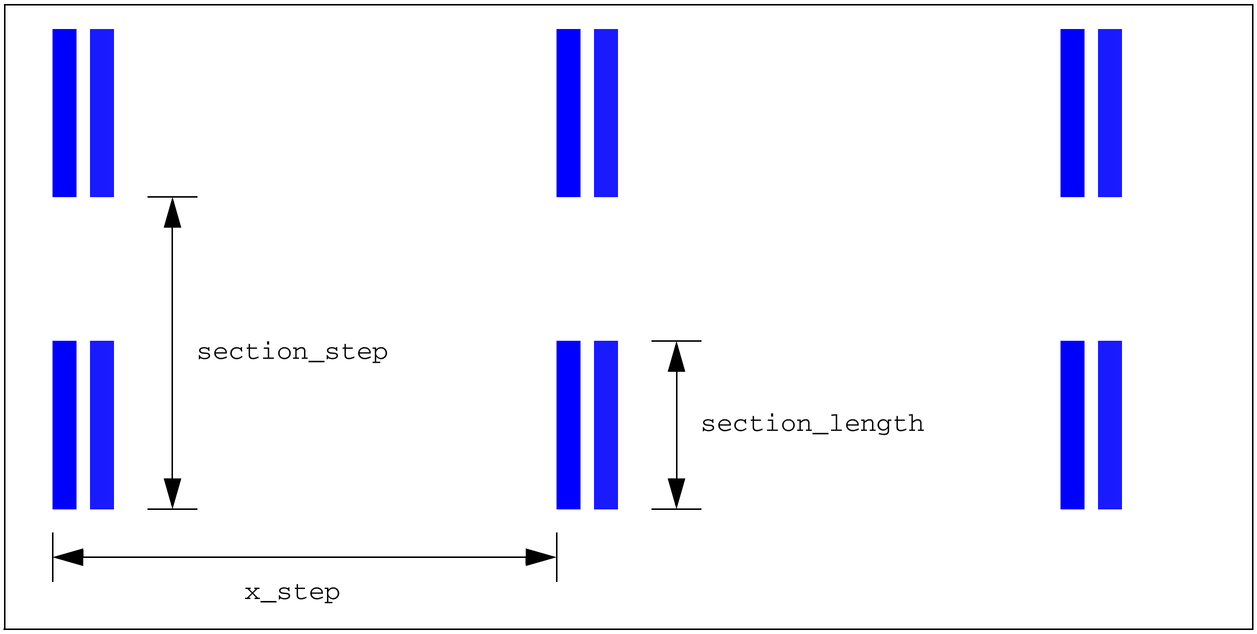


Stripes extend to the core ring when `routing_area` is not given.

Stripes extend to the boundary of the design when no core ring exists and `routing_area` is not given.

The following command creates tandem stripe sections of VDD and VSS in a regular pattern.

```
proute_stripes -nets {VDD VSS} -layers met4 -net_width 8 -net_clearance 3 -x_step
100 -routing_area { 1000 1500 1450 1950} -section_length 40 -section_step 100
```



```
proute_stripes -use_wsp true -nets "vccr_cw vss_cw vcc1 vss1 vcc2 vss2" -layers
Metal4 -centerline true -routing_area {0.000 0.000 10 10} -wsp_pullback_left 0.08
```

Related Information

Tcl Command	proute_trim_stripes

proute_trim_stripes

```
proute_trim_stripes
     -nets {s_netName…}
     [ -layers {s_layerName…} ]
     [ -trim_at_vias [ true | false ]
     | [ -shape_lpps {{s_lppName | s_layerName}…} ]
       [ -rail_types {[core_ring][block_ring][stripes][cell_row_straps]} ] ]
     [ -max_depth i_levelOfHierarchy ]
     [ -include_straps [ true | false ] ]
     [ -power_only [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -undoable [ true | false ] ]
     [ -trim_back_from_boundary [ true | false ] ]
```

Trims stripe ends back to an intersecting ring on the same net (if any), or to connecting vias. Run this command after all stripes and rings have been created.

Arguments

`-include_straps [ true | false ]`

If `true`, allows standard cell straps to be trimmed.

Default: `false`

`-layers {`*`s_layerName`* `…}`

Specifies the metal layers to trim.

`-max_depth` *`i_levelOfHierarchy`*

Specifies the maximum depth down in the cell hierarchy for intersecting shapes.

Default: 0

`-nets {`*`s_netName`*`…}` Trims stripes on the specified nets.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-rail_types [core_ring][block_ring][stripes][cell_row_straps]`

Limits intersecting shapes to the specified power rail types.

`-shape_lpps {{`*`s_lppName`* `|` *`s_layerName`*`}…}`

Limits trimming to intersecting shapes of the given layers and/or layer purpose pairs.

`-silent [ true | false ]`

When `true`, outputs only error messages. When `false`, all message types are output.

Default: `false`

`-trim_at_vias [ true | false ]`

When set to `true`, stripe ends will only be trimmed at connecting vias. If a stripe has zero or one connecting via, then the entire stripe is removed. Vias should be inserted before running this command.

Default: (`false`) Stripe ends are trimmed back to an intersecting ring on the same net.

`-trim_back_from_boundary [ true | false ]`

When set to `false`, stripes that touch the prBoundary bounds are not trimmed back away from the boundary.

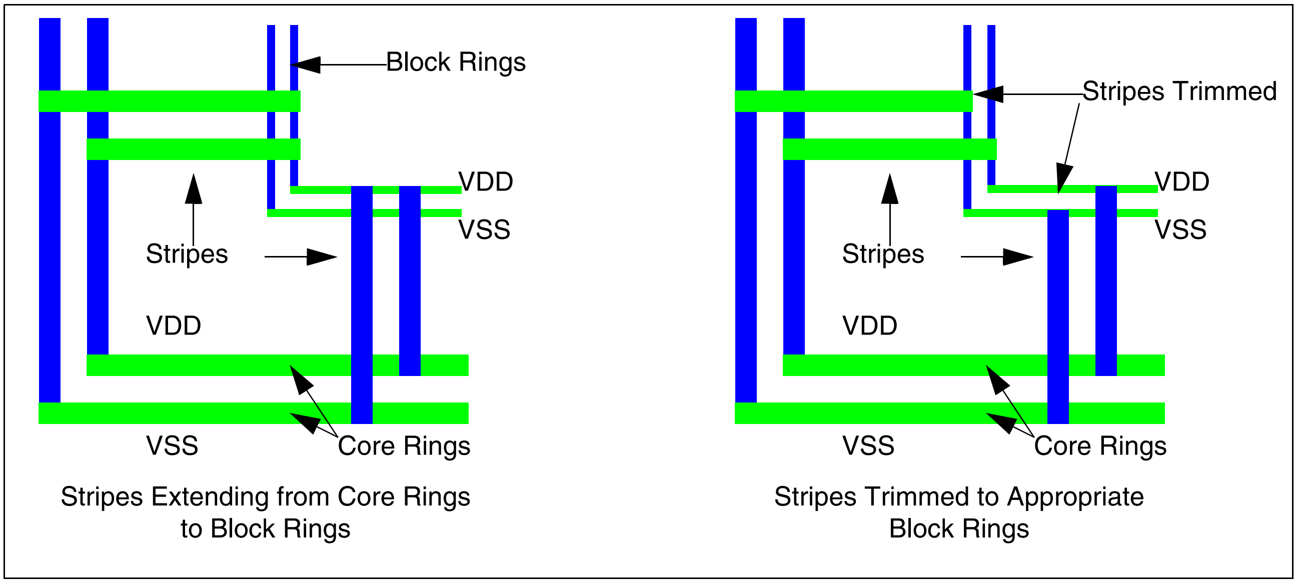
Default: `true`

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

Example

The following example trims power and ground stripes on Metal7 and Metal8 to an intersecting ring on the same net.


Block Rings
VDD
VSS
Stripes
VDD
VSS
Core Rings
Stripes Extending from Core Rings to Block Rings
Stripes Trimmed
VDD
VSS
Stripes
VDD
VSS
Core Rings
Stripes Trimmed to Appropriate Block Rings

```
proute_trim_stripes -net {VDD VSS} -layers {Metal7 Metal8}
```

Related Information

Tcl Command	proute_stripes

proute_via_insertion

```
proute_via_insertion
     -nets {s_netName…}
     [ -from {core_ring | block_ring | stripes | cell_row_straps | all} ]
     [ -to {core_ring | block_ring | stripes | cell_row_straps | all} ]
     [ -from_set d_setObj -to_set d_setObj ]
     [ { -from_layer s_layerName -to_layer s_layerName
         [ -cut_width {f_width…} -cut_height {f_height…} ] }
     | {{[ -set d_setObj | -instances {all | {s_instName…}} ]
         [ -include_straps [ true | false ] ]
         [ -connect_top_level_pins [ true | false ] ]}
       [ -pin_layers {s_layerName…} | -layers {s_layerName…}
       [ -to_layer s_layerName ] ] } ]
     [ -min_layer s_layerName ]
     [ -max_layer s_layerName ]
     [ -routing_area {f_xlo f_ylo f_xhi f_yhi} ]
     [ -rows {i_rows…} -columns {i_columns…} | -square_cut_array
     [ true | false ] ]
     [ -conservative_cut_space [ true | false ] ]
     [ -ignore_obstacles [ true | false ] | -ignore_purposes {s_purposeName…}]
     [ -ignore_rail_orientation [ true | false ] ]
     [ -include_narrow_pins [ true | false ] ]
     [ -max_msg_count i_count ]
     [ -observe_rectilinear_prBoundary [ true | false ] ]
     [ -power_only [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -skip_via_checks {[same_net][diff_net][min_area][num_cuts][extensions]
       [min_edge ][stack_limit]} ]
     [ -tap_to_depth i_tapToDepth ]
     [ -threads i_count ]
     [ -undoable [ true | false ] ]
     [ -use_checker [ true | false ] ]
     [ -use_valid_routing_vias [ true | false ] ]
     [ -via_through [ true | false ] ]
```

Adds vias for interlayer connections at intersections of rings, rails, stripes and straps, or between instance pins and stripes. You can choose the cut array dimensions as a specific number of rows and columns or an equal number of rows and columns. If specific dimensions are not given, then the number of cuts in rows and columns is calculated to maximize the number of cuts that will fit in the narrower dimension of the available area for each via.

Note: To use this command, `minWidth`, `minSpacing` and an extension rule (`minExtension` or `minDualExtension`) must be set for the via layers that are processed.

Arguments

`-columns` **{***`i_columns…`***}**

Specifies the number of columns in the cut array for all vias. If a single value is given, it applies to all cut layers. If a list is specified, there must be one value given for each cut layer between `-from_layer` and `-to_layer`, and the first value refers to the lowest cut layer. If the cut array with these dimensions cannot fit within a via area, no via is created for that metal intersection. Both arguments, `-row`s and `-columns`, must be given.

`-connect_top_level_pins [ true | false ]`

Specifies whether top-level pins are included in stripe-to-pin connections. By default, top-level pins are excluded.

If neither `-set` nor `-instances` is given, adds vias between stripes and top-level pins. No other connections are made.

`-conservative_cut_space [ true | false ]`

If `true` and `minAdjacentViaSpacing` is set, uses this larger cut spacing in one dimension. By default, if `false,` or if `minAdjacentViaSpacing` is not set, uses the appropriate cut spacing (`minSpacing` or `minSameNetSpacing`) for each via.

`-cut_height` **{***`f_height…`***}**

Specifies the Y dimension of each cut. A single value applies to all cut layers. If a list is specified, there must be one value given for each cut layer between `-from_layer` and `-to_layer`, and the first value refers to the lowest cut layer. Both arguments, `-cut_height` and `-cut_width`, must be given.

`-cut_width` **{***`f_width…`***}**

Specifies the X dimension of each cut. A single value applies to all cut layers.If a list is specified, there must be one value given for each cut layer between `-from_layer` and `-to_layer`, and the first value refers to the lowest cut layer. Both arguments, `-cut_height` and `-cut_width`, must be given.

`-from {core_ring | block_ring | stripes | cell_row_straps | all}`

Connects only from this rail type.

Default: `all`

`-from_layer` *`s_layerName`*

Must be given with `-to_layer` to limit processing to the cut layers between `-from_layer` and `-to_layer`. Cannot be used with `-set` or `-instance`.

Default: all layers

`-from_set` *`d_setObj`* Connects only from objects in the given set.

`-ignore_obstacles [ true | false ]`

When `true`, ignores all obstacles and creates power segments without regard to rule violations. This is useful to get a quick preliminary power layout or when it is known that there are no obstacles, such as on an empty layer.

Default: `false`

`-ignore_purposes` **{***`s_purposeName…`***}**

Similar to `-ignore_obstacles`, ignores objects on the given purposes. By default, no purposes are ignored.

`-ignore_rail_orientation [ true | false ]`

When `true`, ignores relative rail orientation and permits vias to be inserted between rails of the same orientation. When `false`, only adds vias between orthogonal rails of differing orientation.

Default: `false`

`-include_narrow_pins [ true | false ]`

When set to `true`, allows vias on pins that are narrower than one via.

Default: `true`

`-include_straps [ true | false ]`

When set to `true`, vias are inserted to connect straps to standard cell pins. When set to `false`, vias are not inserted on standard cell pins.

Default: `false`

`-instances {all |` **{***`s_instName…`***}**`}`

Adds only vias between stripes and either the pins of all instances or the pins of the instances in the list. No other connections are made.

If neither `-set`, `-instances`, nor `-connect_to_top_level_pins` is given, vias are added at intersections of rings, rails, straps and stripes. No pin connections are made.

`-layers` **{***`s_layerName`***…}**

(Same as `-pin_layers`) Used with `-set` or `-instances`, and/or `-connect_to_top_level_pins` to limit stripe-to-pin connections to the pins of the selected/listed instances and/or top level pins on the given layers.

`-max_layer` *`s_layerName`*

Specifies the highest metal layer to be connected. The default is the top metal layer.

`-max_msg_count` *`i_count`*

Specifies the maximum number of message for each message type to output.

Default: 10

`-min_layer` *`s_layerName`*

Specifies the lowest metal layer to be connected. The default is the bottom metal layer.

`-nets` **{***`s_netName`***…}** Specifies the nets to route.

`-observe_rectilinear_prBoundary [ true | false ]`

When `true`, only adds vias within the design's rectilinear prBoundary. When `false`, vias can be added outside the design boundary.

Default `false`

`-pin_layers` **{***`s_layerName`* **…}**

(Same as `-layers`) Used with `-set` or `-instances`, and/or `-connect_to_top_level_pins` to limit stripe-to-pin connections to the pins of the selected/listed instances and/or top-level pins on the given layers.

If neither `-set`, `-instances`, nor `-connect_to_top_level_pins` is given, vias are added at intersections of rings, rails, straps and stripes. No pin connections are made.

`-power_only [ true | false ]`

If `true` (default), only power and ground nets are allowed. If `false`, signal nets are also allowed.

`-routing_area` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Limits the via insertion to the region given.

`-rows` **{***`i_rows…`***}**

Specifies the number of rows in the cut array for all vias. If a list is specified, there must be one value given for each cut layer between `-from_layer` and `-to_layer`, and the first value refers to the lowest cut layer. If the cut array with these dimensions cannot fit within a via area, no via is created for that metal intersection. Both arguments, `-rows` and `-columns`, must be given.

`-set` *`d_setObj`*

Adds only vias between stripes and the pins of the selected instances. No other connections are made.

If neither `-set`, `-instances`, nor `-connect_to_top_level_pins` is given, vias are added at intersections of rings, rails, straps and stripes. No pin connections are made.

`-silent [ true | false ]`

When `true`, no messages are output. When `false`, all message types are output.

Default: `false`

`-skip_via_checks` **{**`[same_net][diff_net][min_area][num_cuts][extensions][min_edge ][stack_limit]`**}**

If running the checker (`-use_checker true`/default), skip the via checks in the given list.

`-square_cut_array [ true | false ]`

If `true`, specifies a square cut array for each via, with an equal number of rows and columns. However, the cut array might not be square in size, depending on spacing rules. If the `minAdjacentViaSpacing` constraint is set, then the spacing of the cuts can be different in the X and Y directions, producing a cut array that is rectangular in dimension.

The default is `false`, and does not require an equal number of rows and columns for the cut array.

`-tap_to_depth` *`i_tapToDepth`*

Adds vias at the top level for interlayer connections between top-level shapes, and between top-level and lower-level shapes for each level of the hierarchy down to and including the specified depth.

Default: Vias are added only at the top level for interlayer connections at the top level.

`-threads` *`i_count`* Specifies the number of threads or processors to use for checking. By default, if multi-threading has been enabled, the session threads are used, otherwise, a single processor is used.

`-to {core_ring | block_ring | stripes | cell_row_straps | all}`

Connects only to this rail type.

Default: `all`

`-to_layer` *`s_layerName`*

When used with `-from_layer`, connects only from the -`from_layer` metal layer to this layer.

When used with `-pin_layers` or `-layers`, connects only from the given pin layers to this layer.

Default: Connects to all layers

`-to_set` *`d_setObj`* Connects only to objects in the given set.

`-undoable [ true | false ]`

Permits this command to be undone. Default is the current setting for the `proute.undoable` environment variable which defaults to `false` on startup.

`-use_checker [ true | false ]`

If `true`, runs the checker on new vias. If `false`, no checking is performed.

Default: `true`

`-use_valid_routing_vias [ true | false ]`

If `true`, use the `validRoutingVias` constraint for vias to use. If `false`, create standard vias.

Default: `false`

`-via_through [ true | false ]`

If `true`, adds vias from bottom-most to top-most rail layer. If `false`, first adds vias between adjacent rail layers.

Default: `false`

Example

The following example inserts vias at intersections of rings, rails, straps and strips for `VSS` and `VDD`, but no pin connections are made.

```
proute_via_insertion -nets {VDD VSS}
```

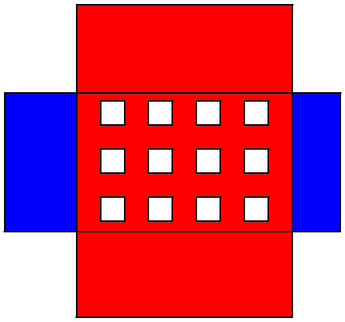
The following example only adds vias between `VSS` stripes and pins on the `M4` layer of the selected instances.

```
proute_via_insertion -nets VSS -set [get_selection_set] -layers M4
```

The following example adds vias between `VSS` stripes and all pins of the selected instances.

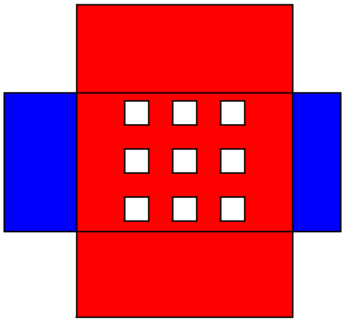
```
proute_via_insertion -nets VSS -set [get_selection_set]
```

The following examples show how the cut array arguments can be used for different results.



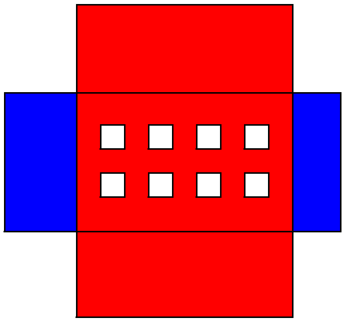
`-square_cut_array false (default)`

Maximum Number of
Rows and Columns



`-square_cut_array true`

Equal Number of
Rows and Columns



`-rows 2 -columns 4`

Specific Number of
Rows and Columns

Troubleshooting the Power Router

If you have trouble connecting power structure objects, particularly if the objects are created externally, check the following:

- If you are running the power router on a power or ground net, is the `signalType` property for the net set to `power` or `ground`?

  By default, the power router will only operate on power and ground nets. If the net is not specified as `power` or `ground`, then you must either change the net's `signalType` property value or use -`power_only false` with the power router Tcl commands to permit the power router to route signal nets.

- Are the power structure objects labeled properly so that they will be recognized by the power router?

  For example, proute_pin_to_trunk will not connect to a *tap* pin that is labeled as a *tie-off*. Use set_power_type to ensure that the objects are labeled properly so that they will be recognized by the power router.

- Do you have shapes on a power or ground net that are unassigned and are only physically connected to the net by overlap or abutment?

  If the unassigned objects are pins, the power router might report that no pains have been found for router. Run extract_net_connectivity on the net to resolve unassigned shapes.

9

Specialty Route Commands

This chapter describes the Tcl commands for mixed signal and other specialty routing.

The commands are sorted by type and presented in the following order:

Bus Routing Commands

This section includes commands related to routing buses.

bus_route

```
bus_route
     [ -allow_violations [ true | false ] ]
     [ -set d_setObj ]
     [ -critic [ true | false ] ]
```

Routes buses in the given set or in the entire design.

Before running this command, you must first form the bus, using create_group to designate the nets of the bus as a `net_bundle`. This grouping can be done with the nets in a set, or combined with append_group to add nets individually to the group.

Arguments

`-allow_violations [ true | false ]`

Specifies whether the new routing can include violations. When `false`, any new routing that causes violations is discarded.

Default: `false`

`-critic [ true | false ]`

When `true`, smooth wires by removing unnecessary jogs.

Default: `false`

`-set` *`d_setObj`*

Routes nets in the set. If this argument is not given, all buses in the design are routed.

Example

The following command creates and routes a bus from nets whose names begin with `dataIN`.

```
set dataINNets [find_net -name dataIN*]
create_group -type net_bundle -name dataINbus -set $dataINNets
bus_route -set $dataINNets
```

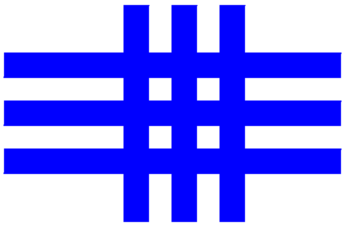
Related Information

Tcl Commands	append_group create_group

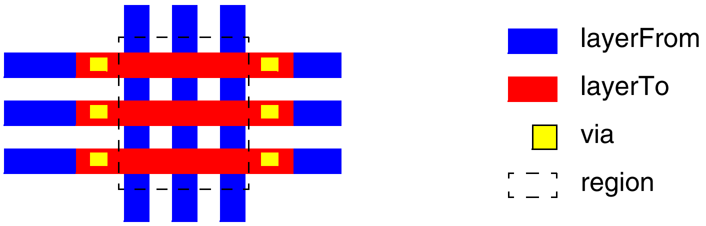
bus_tunnel

```
bus_tunnel
     -set d_setObj
     -region {f_xlo f_ylo f_xhi f_yhi}
     -layerFrom s_layerName
     -layerTo s_layerName
     [ -adjustVias [ true | false ] ]
     [ -keepDistance [ true | false ] ]
     [ -partialResult [ true | false ] ]
     [ -viaStag [ true | false ] ]
```

Re-routes nets in the given set by tunneling from the *from* layer to the *to* layer within the given region.



Shorts caused by overlapping busses on the same layer.



Shorts are removed by tunneling the horizontal routes.

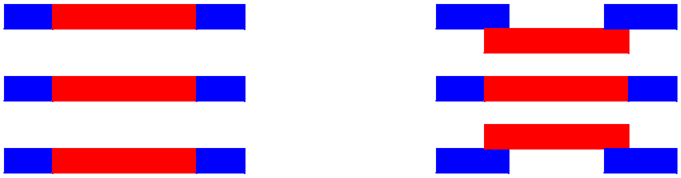
Arguments

`-adjustVias [ true | false ] ]`

When `true`, after tunneling, router will attempt to adjust vias to align via edges with connecting segment edges, and fix extension and numcut violations while maintaining existing via stacks. By default (`false`), vias will not be adjusted and will be centerline connected.

`-keepDistance [ true | false ]`

Specifies whether to keep existing spacing (`true`/default) or use minimum spacing (`false`) to gather wires when tunneling.



a) `-keepDistance true`

b) `-keepDistance false`
The re-routed wires use minimum spacing.

`-layerFrom` *`s_layerName`*

Specifies the *from* layer. Selected routes on this layer within the given region are re-routed to the *to* layer.

`-layerTo` *`s_layerName`*

Specifies the *to* layers. Selected routes on the *from* layer within the given region are re-routed to this layer.

`-partialResult [ true | false ]`

Specifies whether partially completed new routing will be kept even when they cause violations (`true`) or will be discarded (`false`/default).

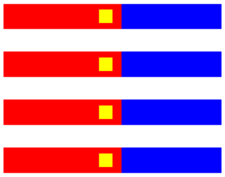
`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area within which the selected nets must be routed on the *to* layer, instead of the *from* layer.

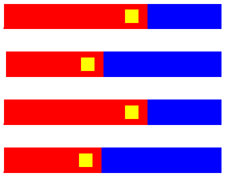
`-set` *`d_setObj`* Re-routes wires for nets in the set.

-viaStag [true | false]

Specifies whether vias will be staggered (true) or perpendicular (false/default).



Perpendicular (default)



Staggered

Example

The following example re-routes wires on the Metal1 layer to the Metal2 layer for nets in the myNets set within the given region with perpendicular vias, keeping existing spacing between wires and discarding any re-routes that cannot be completed.

```
bus_tunnel -set $myNets -region {10 10 20 20} -layerFrom Metal1 -layerTo Metal2 \
  -keepDistance true -viaStag false -partialResult false
```

Related Information

Tcl Commands	bus_route finish_bus_route

finish_bus_route

```
finish_bus_route
     -set d_setObj | -setFromTo d_setObj [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -allow_violations [ true | false ] ]
     [ -exclude_p2p [ true | false ] ]
     [ -no_wrong_way [ true | false ] ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -verbose [ true | false ] ]
```

Finishes bus routing for guides in the given set (`-set`) or for a set of nets (`-setFromTo`). For unrouted busses, use bus_route instead.

Arguments

`-allow_violations [ true | false ]`

Specifies whether the new routing can include violations (`true`). When `false`, any new routing that causes violations is discarded.

Default: false

`-exclude_p2p [ true | false ]`

When `true`, the guided point-to-point router will not be used as the last routing strategy. This can be useful as a time-saving measure if there are long routes that the point-to-point router is less likely to complete. When `false` (default), the guided point-to-point router will be tried as the final routing strategy.

`-no_wrong_way [ true | false ]`

When `true`, wrong-way routing is avoided. When `false` (default), wrong-way routing can be used.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

(Used with `-setFromTo`) Specifies the boundary points for the area in which to operate. If not specified, operates on all guides for the nets in the `-setFromTo` set.

`-set` *d_setObj* Finishes bus routing for the given set of guides.

`-setFromTo` *d_setObj* Specifies the set of nets for which guides will be created and used to finish bus routing. Use this when guides do not exist between the pins and pre-routed wires.

`-verbose [ true | false ]`

Specifies whether messages are output. By default (`true`), messages are output.

Example

The following command finishes bus routing for guides in the given set.

```
finish_bus_route -set [get_selection_set]
```

Related Information

Tcl Commands	bus_route bus_tunnel

Segment-style Routing Commands

These commands let you create segments, two- and three-segment trees, and vias to connect the segments.

add_segment

```
add_segment
     -net s_netName
     -layer s_layerName
     {-from_loc {f_x f_y}
     | -from {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}}
     {-to_loc {f_x f_y}
     | -to {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}}
     [ -begin_ext f_userunit ]
     [ -end_ext f_userunit ]
     [ -check [ true | false ] ]
     [ -convert_to_pin [ true | false ] ]
     [ -deviation f_userunit ]
     [ -deviation_dir {LEFT | RIGHT | BOTTOM | TOP} ]
     [ -fix_violation [ true | false ] ]
     [ -route_type {powerTieOff | shield | shieldTieOff | powerTap | spine} ]
     [ -remove_overlaps [ true | false ] ]
     [ -shield [ true | false ] ]
     [ -width f_userunit ]
```

Adds a route segment for the given net on the given layer *from* and *to* the given locations, relative pins, or relative boundaries. By default, if the segment causes a DRC violation, it will automatically be moved to the nearest DRC clean location within the bounds given by the `-deviation` argument and, optionally, in the direction given by `-deviation_dir`. The added segment can optionally be converted to a pin shape (-`convert_to_pin`).

For proper placement, at least one of the *from* and *to* locations must be given as a location or relative pin. You cannot use relative boundaries (*`BOUND`) for both `-from` and -`to` arguments.

Arguments

`-begin_ext` *`f_userunit`*

Specifies the begin extent of the segment. Refer to Figure Figure 9-1 on page 564 for an example of how extents are handled.

Default: One-half of the `minWidth` on the ruleSpec

`-check [ true | false ]`

When set to `true`, adds the segment only if the result is DRC clean. Use `-fix_violation` to permit the segment to be moved to a DRC clean location. When set to `false`, adds the segment without checking or fixing.

Default: `true`

`-convert_to_pin [ true | false ]`

Converts the added segment to a pin shape.

Default: `false`

`-deviation` *`f_userunit`*

Specifies the maximum deviation, in microns, from the given destination that the segment can be moved to prevent a DRC violation. The default value is one track.

`-deviation_dir {LEFT | RIGHT | BOTTOM | TOP}`

Specifies the direction for a deviation, if the added segment causes a DRC violation. By default, the closest DRC clean location is used.

`-end_ext` *`f_userunit`*

Specifies the end extent of the segment. Refer to Figure Figure 9-1 on page 564 for an example of how extents are handled.

Default: One-half of the `minWidth` on the ruleSpec

`-fix_violation [ true | false ]`

This argument applies only when `-check` is true.

`true`	If the added segment will cause a DRC violation, the segment will be moved to the nearest DRC clean location subject to the bounds given by `-deviation`. This is the default.
`false`	The segment will be added at the given location only if it does not cause a DRC violation.

`-from {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}`

Specifies the relative location, as a pin or boundary, from which to start the segment.

`BMOST`	Bottommost pin
`LMOST`	Leftmost pin
`RMOST`	Rightmost pin
`TMOST`	Topmost pin
`BBOUND`	Bottommost boundary
`LBOUND`	Leftmost boundary
`RBOUND`	Rightmost boundary
`TBOUND`	Topmost boundary

Important

If a relative boundary is given for `-from`, then a relative boundary cannot be specified for `-to`.

`-from_loc` **{***`f_x f_y`***}** Specifies the *from* location coordinates.

`-layer` *`s_layerName`* Specifies the layer for the segment.

`-net` *`s_netName`* Specifies the name of the net to add the segment to.

`-remove_overlaps [ true | false ] ]`

When `true`, will remove segments that overlap with pin shapes for more than half of their length.

Default: `true`

`-route_type {powerTieOff | shield | shieldTieOff | powerTap | spine}`

Specifies the route type for the added segment. The default is `UnknownRouteType`.

`-shield [ true | false ]`

When set to `true`, the added segment is treated as if it will be shielded, by increasing spacing requirements around it. This is useful if custom shields will be added using add_segment `-route_type shield`.

Default: `false`

`-to {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}`

Specifies the relative location, as a pin or boundary, at which the segment must end.

`BMOST`	Bottommost pin
`LMOST`	Leftmost pin
`RMOST`	Rightmost pin
`TMOST`	Topmost pin
`BBOUND`	Bottommost boundary
`LBOUND`	Leftmost boundary
`RBOUND`	Rightmost boundary
`TBOUND`	Topmost boundary

Important

If a relative boundary is given for `-to`, then a relative boundary cannot be specified for `-from`.

`-to_loc` **{***f_x f_y***}** Specifies the *to* location coordinates.

`-width` *f_userunit* Specifies the width of the segment. By default, the segment width is the `minWidth` of the route spec for the net.

Example

The following command adds a segment to `netA` on layer `Metal2`.

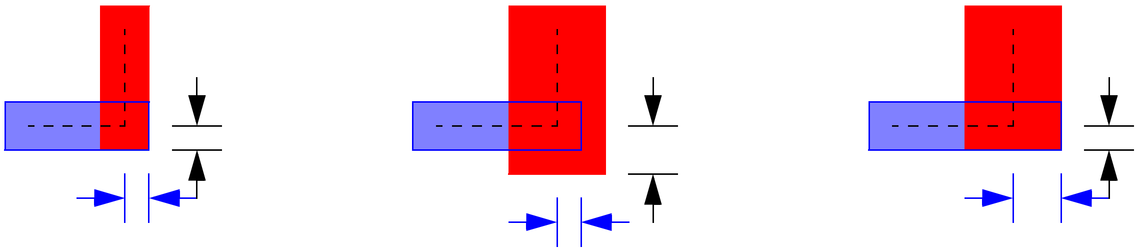
```
add_segment -net netA -from_loc {900 850} -to_loc {910 850} -layer Metal2
```

The following command adds a segment to `netA` on `M2` from the topmost pin of the net to the bottom boundary with a width of 0.6.

```
add_segment -net netA -layer M2 -width 0.6 -from TMOST -to BBOUND
```

Figure 9-1 on page 564 shows how setting the extents can affect the routing.

Figure 9-1 Specifying the Extents for a Segment



a) For both segments, the default extent of one-half the wire width is used.

b) For both segments, the default extent of one-half the wire width is used, which results in edges that are not aligned.

c) The extent for each segment is set to one-half the width of the other wire which results in edges that are aligned.

Related Information

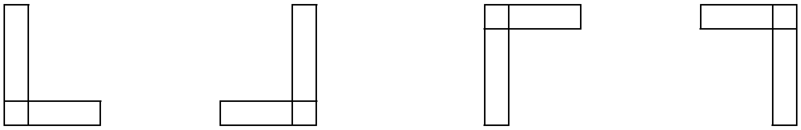
Tcl Commands	add_tree add_via

add_tree

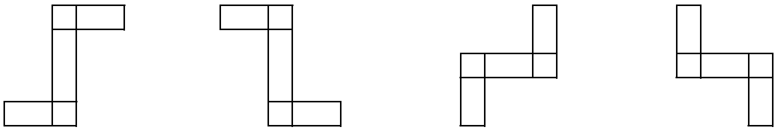
```
add_tree
     -set d_setObj | -net s_netName | {-bus s_busName [ -bits i_count:i_count]}
     -segA true
     -layerA s_layerName
     [ -dirA [ vertical | horizontal ] ]
     [ -widthA f_userunit ]

     -segC [ true | false ]
     -layerC s_layerName
     [ -dirC [ vertical | horizontal ] ]
     [ -widthC f_userunit ]

     [ -segB [ true | false ]
       -layerB s_layerName
       [ -dirB [ vertical | horizontal ] ]
       [ -widthB f_userunit ]
       [ -segB_from {LMOST | RMOST | BMOST | TMOST}
         [ -segB_offset i_tracks ]
         [ -segB_multiplier i_tracks ] ]

     {-from {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}
         [ -from_x f_x | -from_y f_y]
     |-from_pin s_pinName
     |-from_loc {f_x f_y} [ -from_offset i_tracks ]}

     {-to {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}
         [ -to_x f_x | -to_y f_y ]
     | -to_pin s_pinName
     | -to_loc {f_x f_y} [ -to_offset i_tracks ] }

     [ -check [ true | false ]
     [ -convert_to_pin [ true | false ] ]
     [ -deviation f_userunit ]
     [ -fix_violation [ true | false ] ]
```

Creates segments for nets in a set or for bits in a bus to form two-segment or three-segment trees.

Two-segment trees are typically L-shaped, using arguments for segments A and C, given by `layerA` and `layerC.` Configurations for two-segment trees are shown in the following figure.



Three-segment trees are typically Z-shaped, using arguments for segments A, B, and C, given by `layerA`, `layerB`, and `layerC`. Configurations for three-segment trees are shown in the following figure.

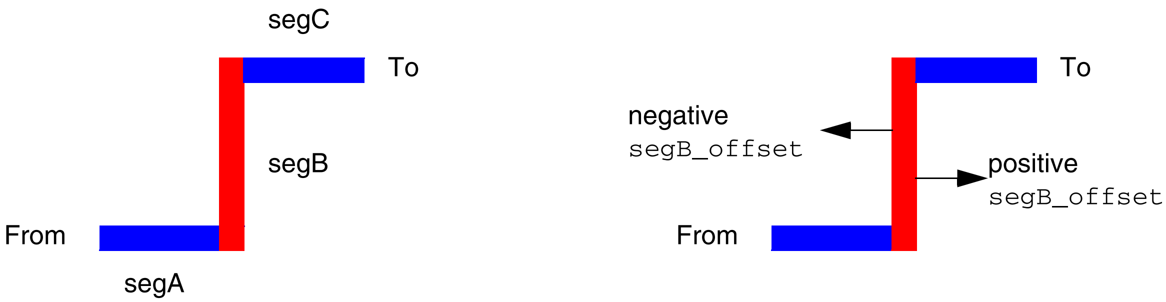


You must specify the layer to use for each segment, and the *from* and *to* location for the tree. You can specify the *from* and *to* locations using any of these methods:

- Pin name
- Coordinates
- Relative pin position (`TMOST` for topmost, `BMOST` for bottommost, `LMOST` for leftmost, `RMOST` for rightmost)
- Relative boundary position (`TBOUND` for topmost, `BBOUND` for bottommost, `LBOUND` for leftmost, `RBOUND` for rightmost)

By default, the preferred direction (horizontal or vertical) for the first segment layer is used, with the opposite direction used for next segment, and so on. You can override these settings using the `-dirA`, `-dirB`, and `-dirC` arguments.

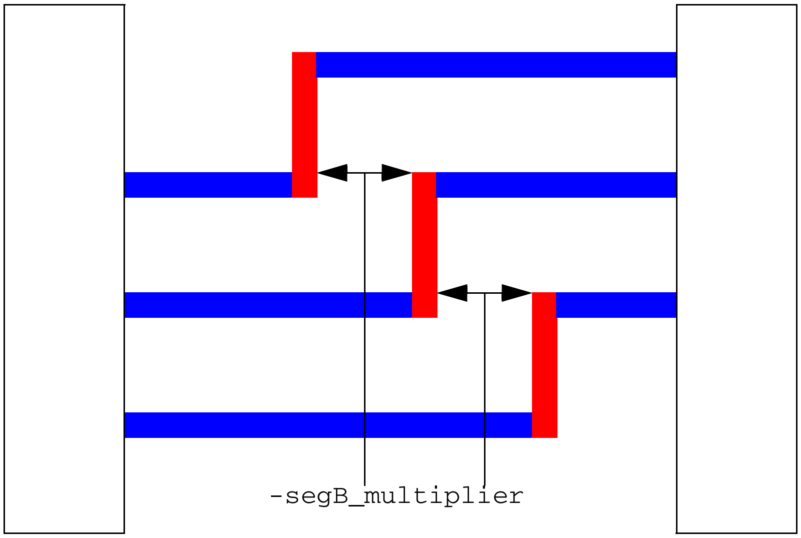
The following figure illustrates the placement of the middle segment of a three-segment tree.



By default, segB is placed midway between the endpoints.

Use `-segB_offset` to change the placement of segB between the endpoints.

To add three-segment trees for bits of a bus, use the `-segB_multiplier` to specify the spacing (in trunks) between nets.



The `add_tree` command is typically used after power routing to route critical nets, and before routing other signals. After the segments are added, connecting vias can be added using add_via, p2p_route, or detail_route.

Arguments

`-bits i_count:i_count` Creates segments for the specified bits of the bus, given by the range of bit numbers. Must be used with the `-bus` argument.

`-bus s_busName` Creates segments for the given bus. If `-bits` is given, segments are created for the given range of bits, otherwise, segments are created for all bits of the bus.

`-check [ true | false ]`

Checks whether the addition of a segment causes DRC violations. Default is `true`.

`-convert_to_pin [ true | false ]`

Converts the added segments to pin shapes.

Default: `false`

`-deviation f_userunit` Specifies the maximum deviation from the given destination that the segment can be moved to prevent a DRC violation. The default value is one track.

`-dirA [ horizontal | vertical ]`

Specifies the direction for the first segment.

`-dirB [ horizontal | vertical ]`

Specifies the direction for the second segment of a three-segment tree.

`-dirC [ horizontal | vertical ]`

Specifies the direction for the second segment of a two-segment tree, or the third segment of a three-segment tree.

`-fix_violation [ true | false ]`

This argument applies only when `-check` is true.

`true` If the added segment will cause a DRC violation, the segment will be moved to the nearest DRC clean location subject to the bounds given by `-deviation`. This is the default.

false	The segment will be added at the given location only if it does not cause a DRC violation.

-from {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}

Specifies the relative position (pin or boundary) to use for the first segment.

BMOST	Bottommost pin
LMOST	Leftmost pin
RMOST	Rightmost pin
TMOST	Topmost pin
BBOUND	Bottommost boundary
LBOUND	Leftmost boundary
RBOUND	Rightmost boundary
TBOUND	Topmost boundary

Important

If a relative boundary is given, then an x- or y-coordinate (-from_x for TBOUND or BBOUND; -from_y for LBOUND or RBOUND) must also be given.

-from_loc **{***f_x f_y***}** Specifies the absolute location for the first segment.

-from_offset *i_tracks* Specifies, in tracks, the offset to use for the first segment. Must be used with -from_loc. By default, this value is 0.

-from_pin *s_pinName* Specifies the name of the pin to use for the first segment.

-from_x *f_x* Specifies the x-coordinate for the first segment. Use this argument with -from TBOUND or -from BBOUND.

-from_y *f_y* Specifies the y-coordinate for the first segment. Use this argument with -from LBOUND or -from RBOUND.

-layerA *s_layerName* Specifies the layer for the first segment.

-layerB *s_layerName* Specifies the layer for the middle segment.

-layerC *s_layerName* Specifies the layer for the last segment.

-net *s_netName* Adds segments for the given net.

`-segA true` Includes the first segment of the tree.

`-segB true` Includes the second segment of the tree.

`-segB_from {TMOST|BMOST|LMOST|RMOST}`

Specifies the pin to use for relative positioning of the middle segment.

BMOST	Bottommost pin
LMOST	Leftmost pin
RMOST	Rightmost pin
TMOST	Topmost pin

`-segB_multiplier` *`i_tracks`*

Specifies the offset, in tracks, for spacing between multiple nets or bus bit routing.

`-segB_offset` *`i_tracks`* Specifies, in tracks, the offset for the middle segment, from the midpoint of the endpoints. By default, this value is 0.

Important

You must specify this argument if you are routing multiple trees to stagger the placement of the middle segments for the nets.

`-segC true` Includes the last segment of the tree.

`-set` *`d_setObj`* Adds segments for nets in the given set.

`-to {LMOST | RMOST | BMOST | TMOST | LBOUND | RBOUND | TBOUND | BBOUND}`

Specifies the relative location (pin or boundary) to use for the last segment.

BMOST	Bottommost pin
LMOST	Leftmost pin
RMOST	Rightmost pin
TMOST	Topmost pin
BBOUND	Bottommost boundary
LBOUND	Leftmost boundary

`RBOUND`	Rightmost boundary
`TBOUND`	Topmost boundary

Important

> If a relative boundary is given, then an x- or y-coordinate (`-to_x for TBOUND` or `BBOUND`; `-to_y` for `LBOUND` or `RBOUND`) must also be given.

`-to_loc` **{***`f_x f_y`***}**
Specifies the absolute location for the endpoint of the last segment.

`-to_offset` *`i_tracks`*
Specifies, in tracks, the offset to use for the last segment. Must be used with `-from_loc`. By default, this value is 0.

`-to_pin` *`s_pinName`*
Specifies the name of the pin to use for the last segment.

`-to_x` *`f_x`*
Specifies the x-coordinate for the last segment. Use this argument with `-to TBOUND` or `-to BBOUND.`

`-to_y` *`f_y`*
Specifies the y-coordinate for the last segment. Use this argument with `-to LBOUND` or `-to RBOUND.`

`-widthA` *`f_userunit`*
Specifies the width of the first segment. By default, the segment width is the `minWidth` of the route spec for the net.

`-widthB` *`f_userunit`*
Specifies the width of the middle segment. By default, the segment width is the `minWidth` of the route spec for the net.

`-widthC` *`f_userunit`*
Specifies the width of the last segment. By default, the segment width is the `minWidth` of the route spec for the net.

Example

The following command adds an L-tree from the leftmost pin of the net in the selected set to an absolute location {20 40}.

```
add_tree -set [get_selection_set] -segA true -dirA horizontal -layerA Metal3 -segC
true -dirC vertical -layerC Metal2 -from LMOST -to_loc {20 40}
```

The following command adds a Z-tree from the topmost pin to the bottommost pin for the nets in the selected set with a spacing of 10 trunks between nets, and the middle segment of the first net offset by one trunk from the midpoint between the endpoints.

```
add_tree -set [get_selection_set] -segA -dirA horizontal -layerA Metal3 \
-segB true -dirB vertical -layerB Metal2 -segB_offset 1 -segB_multiplier 10 \
-segC true -dirC horizontal -layerC Metal3 -from TMOST -to BMOST
```

Guides will be created at the intersections of the route segments and can be added using add_via or detail_route.

The following commands create a bus from a set of nets, then create trees for bits 2 through 4.

```
replace_set -set1 [find_net -name Bus* -ignore_case true -no_wildcard false ] \
-set2 [get_selection_set]
create_group -name myBus -set [get_selection_set] -type net_bundle
add_tree -bus myBus -bits "2:4" -segA -dirA horizontal -layerA Metal3 \
-segB true -dirB vertical -layerB Metal2 -segB_offset 1 -segB_multiplier 10 \
-segC true -dirC horizontal -layerC Metal3 -from TMOST -to BMOST
```

Related Information

Tcl Commands	add_segment add_via

add_via

```
add_via
     -net s_netName
     -name s_viaName
     -loc {f_x f_y}
     [ -check [ true | false ] ]
     [ -convert_to_pin [ true | false ] ]
```

Adds a via to a net at the given location.

Important

When using this command to add vias to existing routing, use the coordinates given by guides to ensure proper centerline placement.

Arguments

`-check [ true | false ]`

When set to `true`, adds the via only if the result is DRC clean. When set to `false`, adds the via without checking.

Default: `true`

`-convert_to_pin [ true | false ]`

Converts the added via to a pin shape.

Default: `false`

`-loc {f_x f_y}`

Is the absolute location for the origin of the via.

`-name s_viaName`

Specifies the name of the via from the technology library.

`-net s_netName`

Specifies the name of the net to add the via to.

Example

The following command adds a `VIA2X` via to `netA`.

```
add_via -net netA -name VIA2X -loc {907 1172}
```

Related Information

Tcl Commands

add_segment
add_tree

Spine-style Routing Commands

Spine-style routing is used to reduce delay from the output driver to the farthest load and minimize the global skew.

balance_route

```
balance_route
     {-set d_setObj | -net1 s_netName -net2 s_netName}
     -spine_on_layers {s_layerName…}
     [ -ignore_pins_in_instances {s_instanceName…}
     | -ignore_pins_in_set d_setObj ]
     [ -half_shield [ true | false ] ]
     [ -multiple_spines [ true | false ] ]
     [ -pin_pair_dist f_distance ]
     [ -spine_dist f_distance ]
     [ -tie_pins_to_spine [ true | false ] ]
```

Balances the geometry for clock pairs in a given set (`-set)`, or for a single net pair (`-net1 -net2`). By default, a single spine is created, but multiple spines can be used (`-multiple_spines true`).

To get a list of net pairs ordered by decreasing calculated spine length, run report_balanced_length. Then use the list to balance route the net pairs in order for the best results.

Power should be routed before using this command because the router will attempt to snap the spine to power rails.

Arguments

`-half_shield [ true | false ]`

When `true`, half-shields each net of a given clock pair with an existing power rail. When `false`, only one net of each clock pair is half-shielded.

Default: `true`

`-ignore_pins_in_instances` **{***`s_instanceName`*…**}**

Ignores pins for the specified instances when placing the spine. The ignored pins will be connected but will not influence the position of the spine.

`-ignore_pins_in_set` *`d_setObj`*

Ignores pins for the instances in the set when placing the spine. The ignored pins will be connected but will not influence the position of the spine.

`-multiple_spines {true|false ] ]`

When `true`, creates multiple spines close to pin pairs. When `false`, taps connect pins to a single spine. Default: `false`

`-net1` *`s_netName`* Balance routes the specified net with the net given by `-net2`.

`-net2` *`s_netName`* Balance routes the specified net with the net given by `-net1`.

`-pin_pair_dist` *`f_distance`*

Pairs pins of the nets for balance routing when pins are within the specified distance (in microns) of each other. Pins that are not paired will not be considered when generating the spine, but will be routed. If no pairs are found that meet the distance criteria, no routing will be done.

Default: `0.5`

`-set` *`d_setObj`* Balance routes nets in the set.

`-spine_dist` *`f_distance`* Will consider clustering pins into a single spine if they are within the specified distance.

Default: `pin_pair_dist*2`

`-spine_on_layers` **{***`s_layername`***…}**

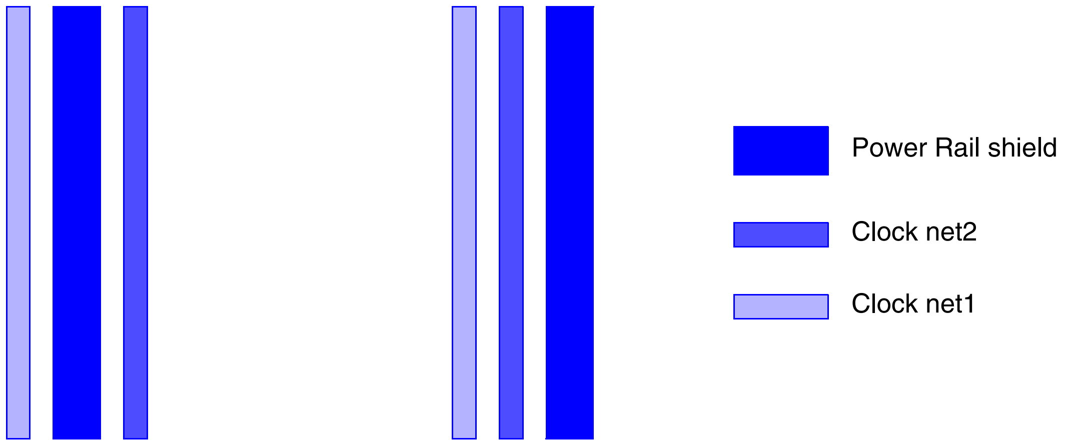
Creates spines on the specified layers.

`-tie_pins_to_spine [ true | false ]`

If `true`, ties pins to the generated spine. If `false`, pins in the net are not tied to the spine, but guides are drawn to indicate where pin connections are needed.
Default: `true`

Example

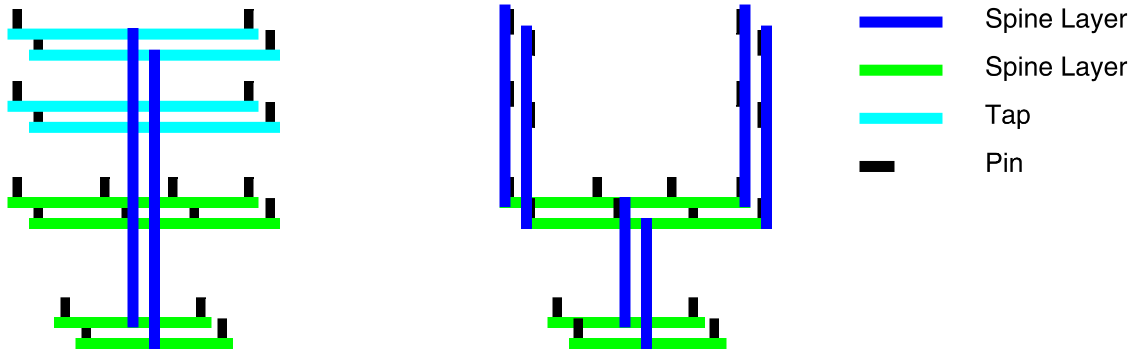
The following figure shows how the `half_shield` argument affects the placement of balanced routes.



`When -half_shield` is `true` (default), each net of the pair is shielded on one side by an existing power rail.

`When -half_shield` is `false`, one net of the pair is shielded by an existing power rail.

The following figure shows a generalized example of balance routing using a single spine (default) compared with multiple spines.



By default, -`multiple_spines` is `false` and a single spine is used.

When -`multiple_spines` is `true`, taps to pins are typically shorter in length.

The following command balances routes on nets N1 and N2 using multiple spines and with half-shields on both nets. When creating the spine, pins of instance INST1 are ignored, only pin pairs from N1 and N2 that are within 0.7 microns of each other will influence the placement of the spine, and layers M1 and M2 will be used to the route the spine.

```
balance_route -net1 N1 -net2 N2 -spine_on_layers {M1 M2} -ignore_pins_in_instances
INST1 -pin_pair_dist 0.7 -multiple_spines true -half_shield true
```

Related Information

Tcl Commands	<u>report_balanced_length</u> <u>report_net_stats</u>

report_balanced_length

```
report_balanced_length
     -set1 d_setObj
     -set2 d_setObj
     -file s_fileName
     -spine_on_layers {s_layerName…}
     [ -ignore_pins_in_instances {s_instanceName…}
     | -ignore_pins_in_set d_setObj ]
     [ -half_shield [ true | false ] ]
     [ -multiple_spines [ true | false ] ]
     [ -pin_pair_dist f_distance ]
     [ -spine_length [ true | false ] ]
```

Outputs an ordered list of net names to the named file. The list is created by calculating the balanced spine length for each net from set1 and its paired net from set2, and listing the pairs in order, starting with the longest spine length, to the shortest. Use the output to prioritize the routing of the nets for balance_route, which uses the same arguments to configure the spine.

Arguments

`-file` *`s_fileName`* Outputs an ordered list of balanced nets to the named file.

`-half_shield [ true | false ]`

When `true`, half-shields each net of a given clock pair with an existing power rail. When `false`, only one net of each clock pair is half-shielded.

Default: `true`

`-ignore_pins_in_instances` **{***`s_instanceName`*…**}**

Ignores pins for the specified instances when placing the spine. The ignored pins will be connected but will not influence the position of the spine.

`-ignore_pins_in_set` *`d_setObj`*

Ignores pins for the instances in the set when placing the spine. The ignored pins will be connected but will not influence the position of the spine.

`-multiple_spines [ true | false ] ]`

When `true`, creates multiple spines close to pin pairs. When `false`, taps connect pins to a single spine.
Default: `false`

`-pin_pair_dist` *`f_distance`*

Pairs pins of the nets for balance routing when pins are within the specified distance (in microns) of each other. Pins that are not paired will not be considered when generating the spine, but will be routed. If no pairs are found that meet the distance criteria, no routing will be done.

Default: `0.5`

`-set1` *`s_netName`* Calculates the balanced spine length for each net in this set and its pair from the `-set2` set.

`-set2` *`s_netName`* Calculates the balanced spine length for each net in this set and its pair from the `-set1` set.

`-spine_length [ true | false ]`

Determines whether the spine length measurement is included in the output.

Default: `false`

`-spine_on_layers` **{***`s_layername`***…}**

Creates spines on the specified layers.

Example

The following example includes a procedure for ordering net pairs in a file for balanced routing by their calculated spine length (`clockSort`), and a procedure to balance route the net pairs in decreasing spine length order (`clockRoute`).

```
# clockSort
# Input:  infile is a file containing two net names per row
# Output: outfile is a file containing the sorted net pair names by decreasing
#         spine length; includes calculated spine length for each pair.
proc clockSort {infile outfile} {

  set clkset1 [create_set]
  set clkset2 [create_set]

  set fileID [open $infile r]
  while {[gets $fileID line ] >= 0} {
    set clk1 [lindex [split $line \t] 0]
    set clk2 [lindex [split $line \t] 1]
    set clkset1 [or_sets -set1 [find_net -name $clk1 -ignore_case true \
      -no_wildcard true -silent] -set2 $clkset1]
    set clkset2 [or_sets -set1 [find_net -name $clk2 -ignore_case true \
      -no_wildcard true -silent] -set2 $clkset2]
  }
  report_balanced_length -set1 $clkset1 -set2 $clkset2 -file $outfile \
    -spine_length true
  close $fileID
}


# clockRoute
# Input: infile is the name of a file containing two net names per row
#        The nets are balance routed, one pair at a time.


proc clockRoute {infile} {

  set fileID [open $infile r]
  while {[gets $fileID line ] >= 0} {
    set clk1 [lindex [split $line \t] 0]
    set clk2 [lindex [split $line \t] 1]
    set tmppair [or_sets -set1 [find_net -name $clk1 -ignore_case true \
      -no_wildcard true ] -set2 [find_net -name $clk2 -ignore_case true \
      -no_wildcard true ] ]
    balance_route -set $tmppair
  }
  close $fileID
}
```

The following commands sort a list of net pairs in `nets_in.txt` by decreasing spine length, then balance routes the pairs, one-at-a-time.

```
clockSort nets_in.txt nets_sorted.txt
clockRoute nets_sorted.txt
```

Example input file:

```
eclk1 lclk1
eclk2 lclk2
eclk3 lclk3
```

Example output file (includes the optional calculated spine length for each net pair):

```
eclk2 lclk2 10.2
eclk1 lclk1 8.1
eclk3 lclk3 6.4
```

Related Information

Tcl Commands	`balance_route`

report_spine_nets

```
report_spine_nets
     [ -net {s_netName…} | -set d_setObj ]
```

Reports the spine parameters that were set using set_spine_nets for the given nets or nets in the set. By default, all nets that were processed by set_spine_nets are reported.

Arguments

`-net {s_netName…}`	Reports on nets in the list.
`-set d_setObj`	Reports on nets in the set.

Example

The following example shows how set_spine_nets settings can be reported using this command.

```
set_spine_nets -net net1 -direction vertical -trunk_type single_median -tap_type
steiner
report_spine_nets
net net1:
  vertical
  trunk type single_median
  tap type steiner
```

Related Information

Tcl Commands	set_spine_nets

route_taps

```
route_taps
     -set d_setObj
     [ -use_term_taper [false|true ] ]
     [ -gather_routes [ true | false ] ]
```

Routes tap connections for nets in the set. Use this command after routing only spines for the nets with one of the following:

- `balance_route -tie_pins_to_spine false`
- `spine_route -spine_only true`

Arguments

`-gather_routes [ true | false ] ]`

When set `true`, multiple taps are routed using a gathering method. Use this for bus-style routing.

Default: (`false`) Each tap is routed independent of the others.

`-set` *`d_setObj`*

Specifies the set of nets.

`-use_term_taper [ true | false ]`

Determines whether taps are connected using the taper rule on the terms.

Default: (`false`) Uses the taper rule for the net.

Example

The following example routes tap connections for the nets in the selected set, using the taper rule on the terms.

```
route_taps -set [get_selection_set] -use_term_taper
```

Related Information

Tcl Commands	balance_route spine_route

set_route_is_spine

```
set_route_is_spine
     -set d_setObj
```

Identifies which portion of a net is the trunk. This is particularly useful if the trunk is prerouted.

The contents of a trunk are labeled as *spines*. If any portion of the net is labeled as spines, it is assumed that the spine contains routes or terms that are also labeled as spines. If the trunk type is not `single_driver` or `single_median`, then the portion of the net that is connected to the driver is also considered to be a spine.

Arguments

`-set` *`d_setObj`*	Specifies a set of objects whose routes or terms will be labeled as spines.

Related Information

Tcl Commands	unset_route_is_spine

set_spine_nets

```
set_spine_nets
     {-net {s_netName…} | -set d_setObj}
     [ -bias_mode {middle | side | ratio} ]
     [ -clear [ true | false ] ]
     [ -direction {vertical | horizontal}
     [ -layers {s_layerName…} ]
     [ -max_cluster_dist f_distance ]
     [ -max_pins i_count]
     [ -optimize_cluster_ratio i_ratio]
     [ -tap_type {direct | short_steiner | steiner | tree}]
     [ -trunk_rule_spec s_routeSpecName ]
     [ -trunk_type
       { single_driver | single_median | single_middle |
       | single_in_channel | multi_in_channel
       | narrow_middle | narrow_median | wide | tree} ]
```

Sets parameters for spine routing. Usually, this is not required before running spine_route because the spine router can automatically tune the routing based on the configuration of the net. Use this command before running spine_route to force the spine router to use certain settings.

The spine router will normally use a `narrow_middle`, `narrow_median` or `wide` trunk type, depending on the aspect ratio of the net and the location of the receiver pins. If you issue this command without arguments, the spine router will use a `single_median` trunk type instead.

To check the spine parameters that were set by this command, use report_spine_nets.

Arguments

`-bias_mode {middle | side | ratio}`

(Applies only for `-trunk_type single_in_channel` and `-trunk_type multi_in_channel`) Specifies the preferred placement for an individual spine in a channel.

`middle`	Places the spine at the center of the associated channel.
`ratio`	(Default) Places the spine at a position relative to the ratio of the receiving pins on both sides of the channel. The distance from the spine to the edges of the sides have the same ratio as the pins on the two sides. The spine will be closer to the side with more pins.
`side`	Places the spine toward the side with more receiving pins.

`-clear [ true | false ]` Resets parameters that are set by this command to their default settings.

Default: `false`

`-direction {vertical | horizontal}`

(Applies only if the `trunk_type` is `single_driver`, `single_median`, `single_in_channel`, or `multi_in_channel`) Sets the spine direction. By default, the longer aspect of the bounding box of the net pins is used.

`-layers` **{***`s_layerName`***}** Sets preferred layers for the spine. The lowest layer that matches the spine's direction will be preferred. If only one layer is given, it will be used, even when the direction of the spine is perpendicular to the layer's routing direction for `single_driver` and `single_median` trunk types only.

Default: preferred or valid layers for the net

`-max_cluster_dist` *`f_distance`*

Specifies the maximum distance between clustered pins.

`-max_pins` *`i_count`* Specifies the maximum number of pins in a tap cluster.

`-net` **{***`s_netName…`***}**

Sets parameters only for nets in the list.

`-optimize_cluster_ratio` *`i_ratio`*

Used with `-max_pins` and `-max_cluster_dist` to fine-tune clustering. Suggested settings are greater than 2, and 3 is a good average. A value of 3 specifies that a pin will be clustered if the extra distance for clustering (connecting the pin to a neighbor pin) is less than 1/3 of the original distance from the pin to the spine.

`-set` *`d_setObj`*

Sets parameters only for nets in the set.

`-tap_type` *`s_tapType`*

Specifies the topology of the taps.

`direct`

Connects each tap directly to the trunk. This is the default when `trunk_type` is `single_driver` or `single_median`, and for narrow topologies.

`short_steiner`

This is a type of `steiner` tap that allows only local clustering. This is the default when `trunk_type` is not `single_driver` or `single_median`, and the net topology is nearly equal in height and width (has an aspect ratio near one).

`steiner`

Connects to the nearest receiver group or to the trunk, whichever is closest.

`tree`

The tap topology is determined automatically by spine_route based on the aspect ration of the bounding box of the net's receiver pins.

`-trunk_rule_spec` *`s_routeSpecName`*

Specifies the route spec to use for the trunk.

`-trunk_type` *`s_trunkType`*

Specifies the trunk style.

`multi_in_channel`

(Requires `-direction` to specify the direction of the trunk spines) Makes a trunk by placing spines in channels between instance rows containing receivers, then connecting the spines to a perpendicular main trunk and the drivers.

`narrow_median` Makes a trunk that is aligned with the median coordinate in the direction of the spine.

`narrow_middle` Makes a trunk that is aligned with the middle of the net’s bounding box.

`single_driver` Makes a single trunk that connects directly to the driver.

`single_in_channel`

(Requires `-direction` to specify the direction of the trunk) Makes a single trunk in a channel close to the center of the net’s bounding box.

`single_median` Makes a single trunk that is located at the median coordinate in the spine direction. This is the default.

`tree` The trunk placement is determined automatically by spine_route based on the aspect ratio of the net and the topology of the receivers and drivers.

`wide` Makes a trunk by clustering the receivers to identify representatives, then forming a tree where every driver-to-representative path is a shortest path. This is typically used when the net is approximately equal in height and width.

Example

The following command causes spine_route to route using `single_median` trunks instead of one of the normal trunk type defaults: `narrow_middle`, `narrow_median` or `wide`.

```
set_spine_nets
```

The following commands create a single vertical spine on the M2 layer that is connected to the driver.

```
set_spine_nets -net net1 -trunk_type single_driver -layers M2 -direction vertical
spine_route -net net1
```

Related Information

Tcl Commands	report_spine_nets spine_route

spine_route

```
spine_route
     {-net {s_netName…} | -set d_setObj}
     [ -fix_trunk [ true | false ] ]
     [ -spine_only [ true | false ] ]
     [ -tap_embedding {guide | global | detail} ]
     [ -use_existing_guides [ true | false ] ]
```

Routes a net or nets using spine-style routing. A primary trunk is routed from the output driver pin to the farthest receiver, then other receivers are connected in clusters or individually to the primary trunk.

By default, the spine router will use a `narrow_middle`, `narrow_median` or `wide` trunk type, depending on the configuration of the net. `Direct` taps will be used for narrow topologies and `short_steiner` connections will be used for all others. To override these defaults or set the spine direction, layers, or trunk rule spec, use set_spine_nets.

By default, all existing guides are removed before routing spines. Use `-use_existing_guides true` to keep existing guides.

Arguments

`-fix_trunk [ true | false ]`

Determines whether the trunk should be marked fixed.

`-net {`*`s_netName…`*`}` Spine routes nets in the given list.

`-set `*`d_setObj`* Spine routes nets in the given set.

`-spine_only [ true | false ]`

Chooses whether the entire net ((`false`/default) or only the primary trunk (`true`) is routed.

`-tap_embedding {guide | global | detail}`

Specifies the tap embedding type.

`-use_existing_guides [ true | false ]`

When `false` (default), removes existing guides and creates new guides for spine routing. When `true`, keeps existing guides (`true`) and uses them when routing spines.

Example

The following command routes the `clkA` net by automatically configuring the tap topology based on the locations of the driver and receivers.

```
spine_route -net clkA
```

The following commands ensure that a wide trunk topology with steiner taps will be used to route the net.

```
set_spine_nets -net wn1 -trunk_type wide -tap_type steiner
spine_route -net wn1
```

Related Information

Tcl Commands	set_spine_nets

unset_route_is_spine

```
unset_route_is_spine
     -set d_setObj
```

Re-labels a portion of a net that is incorrectly designated as a spine.

Arguments

`-set` *`d_setObj`*	Specifies a set of objects whose routes or terms must not be labeled as spines.

Related Information

Tcl Commands	set_route_is_spine

unset_spine_nets

```
unset_spine_nets
     {-set d_setObj | -net {s_netName…}}
```

Removes the spine designation for the spine nets in the set or in the list and resets any spine parameters that were associated with those nets.

Arguments

`-net {s_netName…}`	Operates on spine nets in the list.
`-set d_setObj`	Operates on spine nets in the set.

Related Information

Tcl Commands	set_route_is_spine

Star Routing Commands

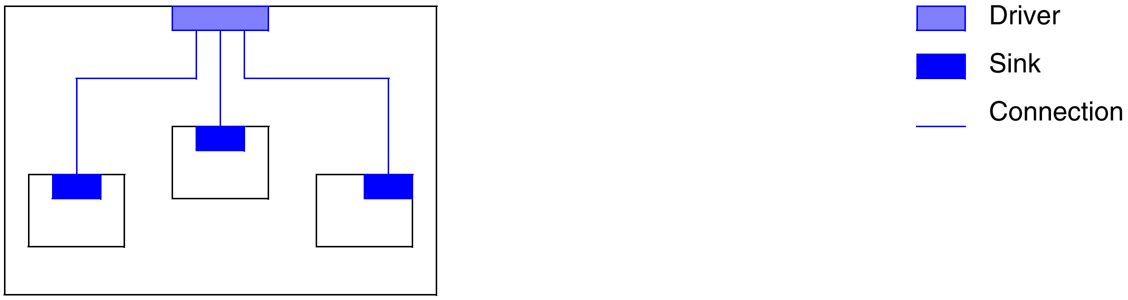
This section describes commands used to connect sinks to a single driver or sinks to multiple drivers.

star_route

```
star_route
     { -set d_setObj }
     [ -convert_terminals {sink | driver | none}]
     [ -multilevel [ true | false ] ]
     [ -multidriver [ true | false ] ]
     [ -driver_constraint [ true | false ] ]
     [ -driver s_termName ]
     [ -fromLayer s_layerName ]
     [ -instance s_instName ]
     [ -connect_drivers [ true | false ] ]
     [ -guides_only [ true | false ] ]
     [ -route_only [ true | false ] ]
     [ -incremental [ true | false ] ]
     [ -honor_to_pt [ true | false ] ]
     [ -allow_share [ true | false ] ]
     [ -allowViolations [ true | false ] ]
     [ -direction {horizontal | vertical} ]
     [ -include_term_set d_setObj ]
     [ -exclude_term_set d_setObj ]
     [ -include_term_names {s_termName…} ]
     [ -exclude_term_names {s_termName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -include_term_region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -exclude_term_region {f_xlo f_ylo f_xhi f_yhi} ]
```

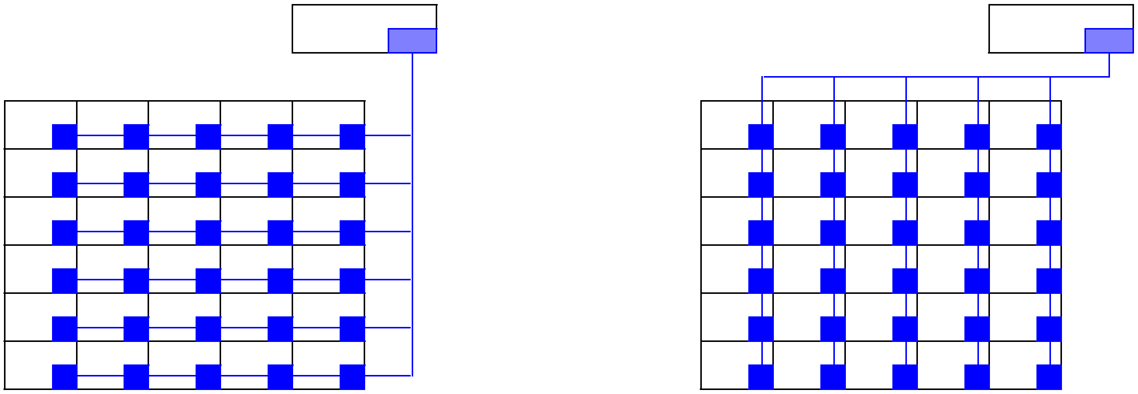
Routes nets that have one driver and multiple pins, or sinks. By default, each sink is connected directly to the driver as shown in Figure 9-2.

Figure 9-2 Single Level Star Routing



If a net has many sinks in a matrix-type layout of rows and columns, use `-multilevel` to connect the driver to multiple interconnected sinks. By default, when `-multilevel` is specified, sinks are connected horizontally in rows. To change this, specify `-direction vertical` to connect multilevel sinks vertically in columns. Figure 9-3 shows examples of a multi-level net star routed horizontally and vertically.

Figure 9-3 Multi-level Star Routing



Multi-level horizontal sink interconnect

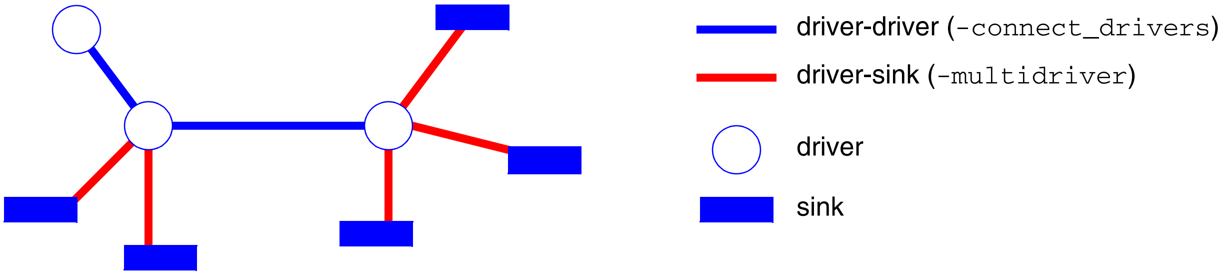
Multi-level vertical sink interconnect

To star route a net that has all sinks or all drivers, use `-convert_terminals` to convert each net to one driver and multiple sinks before routing. The driver for each net will be in the leftmost, rightmost, topmost or bottommost location relative to the sinks.

To star route a net that has multiple bidirectional instTerms, set the `isDriver` constraint to `true` on the driver instTerm, and use `-driver_constraint` to recognize the constraint setting.

If there are multiple drivers on a net, use `-multidriver` to connect sinks to the nearest driver and `-connect_drivers` to connect the drivers to each other. Figure 9-4 shows an example of this type of star routing with multiple drivers and sinks. By default, sinks are routed to only one driver on a net.

Figure 9-4 Star Routing Multiple Drivers to Multiple Sinks



By default, if a net has existing guides and/or route segments, `star_route` will not create additional guides. To create additional routes for a net, use create_fromto with

`-incremental true` to create the additional guides, then use `star_route` with `-route_only true -incremental true` to route them.

Arguments

`-allow_share [ true | false ]`

When set to `true`, route segments can be shared when connecting drivers to sinks. When set to `false`, route segments cannot be shared.

Default: `false`

`-allowViolations [ true | false ]`

When set to `true`, the new routing can include violations. When set to `false`, any new routing that causes a violation is discarded.

Default: `false`

`-connect_drivers [ true | false ]`

When set to `true`, connects routes between multiple drivers. When `false`, will not connect multiple drivers to each other.

Default: true

`-convert_terminals` *`s_type`*

Specifies the conversion type for star routing a net that consists of all sinks or all drivers. Each converted net will have only one driver at the leftmost, rightmost, topmost or bottommost position relative to the sinks.

`driver`	For each selected net, converts all but one driver to sinks before routing.
`none`	No conversion is done. This is the default.
`sink`	For each selected net, converts one sink to a driver before routing.

`-direction {horizontal | vertical}`

[Applies only when `-multilevel true`] Connects the driver to multiple interconnected sinks in the given direction.

Default: `horizontal`

`-driver` *`s_termName`* Star route only from drivers with this terminal name.

`-driver_constraint [ true | false ]`

When set to `true`, determines the drivers from the isDriver constraint on the instTerms. When `false`, determines the drivers from the isDriver property on the terms.

Default: `false`

`-exclude_term_names` **{***`s_termName…`***}**

Will not star route to the named terminals.

`-exclude_term_region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates for an area to be excluded from processing. Terminals in this area will not be considered.

`-exclude_term_set` *`d_setObj`*

Will not star route to the terminals in the given set.

`-fromLayer` *`s_layerName`*

Star routes only from drivers on this layer.

`-guides_only [ true | false ]`

When set to `true`, only guides are created. No routing is performed. The guides can be routed using `global_route` with `-create_guides false` or `star_route` with `-route_only true`.

Default: `false`

`-honor_to_pt [ true | false ]`

When set to `true`, star route will keep the current “to“ point for existing guides. When set to `false`, guides can be adjusted to make the shortest connection.

Default: `true`

`-include_term_names` **{***`s_termName…`***}**

Will star route to only the named terminals.

`-include_term_region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the lower-left (*f_xlo*, *f_ylo*) and the upper-right (*f_xhi*, *f_yhi*) bounding box coordinates for an area. Only terminals in this area will be considered.

`-include_term_set d_setObj`

Will star route only to the terminals in the given set.

`-incremental [ true | false ]`

Specifies whether additional guides can be routed for a net that has existing route segments. When set to `false`, additional guides cannot be created for a net with existing guides and/or route segments.

For incremental star routing, use create_fromto with `-incremental true` to create the additional guides, then use `star_route` with `-route_only true -incremental true` to route them.

Default: `false`

`-instance s_instName`

Star routing connects the driver to only the specified instance.

`-multidriver [ true | false ]`

When set to `true`, handles nets with multiple drivers by connecting sinks to their closest driver. When `false`, connections will be routed for only one driver for each net.

Default: `false`

`-multilevel [ true | false ]`

When set to `true`, permits interconnection of multiple sinks. When set to `false`, each sink is connected to only the driver. Default: `false`

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the lower-left (*f_xlo*, *f_ylo*) and the upper-right (*f_xhi*, *f_yhi*) bounding box coordinates for the area to be processed. Only terminals in this area are considered.

`-route_only [ true | false ]`

When set to `true`, guides are not created and only the existing guides are routed.

Default: `false`

`-set` *`d_setObj`* Specifies the set of nets to operate on.

Examples

The following command star routes the A1 net.

```
star_route -set [find_net -name A1]
```

Custom Topology Routing Commands

This section describes commands used to customize net topology.

create_fromto

```
create_fromto
     -fromSet d_setObj
     -toSet d_setObj
     [ -allow_duplicate [ true | false ] ]
     [ -constraintGroup s_groupName ]
     [ -fromLayer s_layerName ]
     [ -fromPoint { f_x f_y } ]
     [ -incremental [ true | false ] ]
     [ -must_connect [ true | false ] ]
     [ -toLayer s_layerName ]
     [ -toPoint { f_x f_y } ]
```

Creates a route guide between terminals. A constraint group can be assigned to the guide to specify constraints for routing, which will have precedence over constraints for the net, global net default, design and foundry rules. For terminals with more than one pin shape, only one pin will be connected by default, but the choice of pins can be restricted to a specific layer.

By default, if a net has existing guides and/or route segments, create_fromto will not create additional guides. When `-incremental` is `true`, guides can be added to a net that has existing guides and/or route segments. However, duplicate guides between a driver and sink will not be added.

Arguments

`-allow_duplicate [ true | false ]`

When set to `true`, duplicate guides can be created between a “from“ point and a “to“ point. When set to `false`, only one guide is created between each “from“ and “to“.

Default: `false`

`-constraintGroup` *`s_groupName`*

Assigns a constraint group to the new guides. When routed, constraints in the constraint group will be honored.

`-fromLayer` *`s_layerName`*

Considers only “from” pins on the given layer.

`-fromPoint` **{** *`f_x f_y`* **}** Specifies the “from“ point as an x and y coordinate.

`-fromSet` *`d_setObj`* Set containing the “from” terminal.

`-incremental [ true | false ]`

Specifies whether guides can be added to a net that has existing guides and/or route segments. When set to `false`, guides cannot be added to a net with existing guides and/or route segments.

Default: `false`

`-must_connect [ true | false ]`

When set to `true`, treats pins of the same terminal as must-connect by creating a guide from the driver directly to each pin on the terminal. When set to `false`, pins of the same terminal are treated as weak-connect and only one guide is created from the driver to the pins.

Default: `false`

`-toLayer` *`s_layerName`* Considers only “to” pins on the given layer.

`-toPoint` **{** *`f_x f_y`* **}** Specifies the “to“ point as an x and y coordinate.

`-toSet` *`d_setObj`* Set containing the “to” terminal.

Examples

Example 1—Create a guide between two terminals

The following example creates a guide between terminals `t1` and `t3`.

```
create_fromto -fromSet [find_terminal -name t1] -toSet [find_terminal -name t3]
```

Example 2—Route a guide between two terminals

In the following example, a guide is routed between two terminals of a net. First, a guide is created between the terminals, then the routing flow is run on the net ***without creating additional guides*** in the global route step.

```
# Specify the net to route
set myNet [find_net -name AVCC1]

# Create a guide between two terminals
create_fromto -fromSet [find_inst_term -instance_name u_40 -name VDD3] -toSet
[find_inst_term -instance_name u_f27 -name VDD_3P3V]

# Run the routing flow
global_route -set $myNet -create_guides false -mode full
croute -set $myNet
detail_route -set $myNet
```

Pair Routing Commands

For more information on how to use these commands in a flow, refer to “Examples for Pair Routing Scripts” on page 618.

pair_create_topology

```
pair_create_topology
     [ -set d_setObj ]
     [ -pref_regions ]
```

Creates netPairTerms at the closest location to each set of paired memberNet pins. A netPairRoute guide is added to connect the source and target netPairTerms.

Arguments

`-pref_regions`	Identifies regions with pins whose preferred layer routing direction is opposite to the preferred direction for the most efficient routing to or from the pins. Each *no preferred direction* region is marked with a rectangular boundary and the router is permitted to wrong-way route as needed within those regions. When pair_divide is run, the no preferred direction region markings are removed. By default, the router routes in the preferred direction for a layer.
`-set` *`d_setObj`*	Restricts processing to the memberNets in the given set. By default, all memberNets are processed.

Example

Refer to “Examples for Pair Routing Scripts” on page 618 for an example of how this command is used in a pair routing flow.

pair_divide

```
pair_divide
     [ -set d_setObj]
     [ -center_vias [ true | false ] ]
     [ -delete_violations [ true | false ] ]
     [ -preserve_topology [ true | false ] ]
     [ -routing_grid [ true | false ] ]
     [ -space_vias [ true | false ] ]
     [ -use_w2vpitch [ true | false ] ]
```

Divides the netPairRoutes into the memberRoutes by dividing all route segments and vias in the netPairRoutes. The resultant memberRoutes are connected to the memberNet source and target pins. The netPairTerms and the netPairRoutes are removed.

After running this command, use verify_connectivity to check for connectivity violations.

Arguments

`-center_vias [ true | false ]`

When `true`, if a via is wider than a connecting segment, then the center of the via is aligned with the centerline of the segment. In cases where there is minimum spacing between metal, adding center-aligned oversized vias can cause spacing violations. To fix these, specify `-space_vias true` with `-center_vias true`.

Default: (`false`) Oversized vias can be center-aligned or edge-aligned with connecting segments. The router will choose the best method to avoid spacing violations.

`-delete_violations [ true | false ]`

When set to `true`, will check for DRC violations, remove any violating segments and attempt to re-route them with the point-to-point router. Opens will be left for any violations that cannot be fixed and you can use detail_route to complete the routing. detail_route should be used with caution because it is more likely to disturb the balance in wiring with rip-ups and re-routing.

Default: (`false`) Checking is not performed.

`-preserve_topology [ true | false ]`

If set to `true`, will attempt to preserve the topology of the netPairRoutes while fixing DRC violations created by the divide. This is useful when matching lengths of groups of differential net pairs. By default (`false`), priority is given to fixing DRC violations instead of topology preservation.

`-routing_grid [ true | false ]`

When `true`, will place routing elements on the routing grid. When false, will place routing elements on the manufacturing grid.

Default: false

`-set` *`d_setObj`*

Restricts processing to the memberNets in the given set. By default, all memberNets are processed.

`-space_vias`

When `true`, will fix spacing violations for vias at pins. This is useful when you specify `-center_vias true` to center-align oversized vias with connecting metal, which can cause spacing violations. Typically, the violating oversized vias will be shifted to edge-aligned with the connecting metal, eliminating the violation.

Default: `false`

`-use_w2vpitch [ true | false ]`

When `true`, will use wire-to-via pitch when vias are oversized and `-center_vias true`. When `false`, will use via-to-via pitch when placing vias.

Default: `false`

Example

Refer to "Examples for Pair Routing Scripts" on page 618 for an example of how this command is used in a pair routing flow.

pair_report

`pair_report`

Outputs statistics to the Transcript area for net pairs. In addition, a `fat.rpt` file is created that reports net pair violations when the unpaired wire length exceeds 10*`minWidth` for the lowest metal layer.

In the statistics output, the following are reported:

- Summary
  - Number of opens and shorts after the composite nets are routed
  - Number of opens and shorts after the composite nets are divided into member nets
  - Number of opens and shorts for all nets
  - Unpaired percentage of total length of net pairs
  - Elapsed cpu time, in seconds
- Step Details
  - `start`
    - Number of Member Nets
    - Number of opens and shorts for the member nets
    - Total number of other nets
    - Number of opens and shorts for the other nets
  - `create_topology`, `route`
    - Number of opens and shorts for composite nets
    - Number of opens and shorts for other nets
    - Elapsed time, in seconds, for each step
    - Total elapsed time, in seconds
  - `divide`, `clean`
    - Number of opens and shorts for member nets
    - Number of opens and shorts for other nets
    - Total unpaired length and unpaired percentage of total length

- Number of net pair violations and maximum acceptable unpaired length
- Elapsed time, in seconds, for each step
- Total elapsed time, in seconds

In `fat.rpt`, the following information is reported:

- Fat Net Opens
  - Net name
  - Route
  - Length
- Pairing Violations
  - Net name
  - Number of Vias in net
  - Wire Length of net
  - Total distance between pin pairs for the net
  - Best Pairing (Wire Length - PinPair Distance/2)
  - paired wire length
  - Unpaired length and unpaired percentage of total length
- Pair Statistics (reported for each net pair)
  - For each net, number of vias for each layer and total vias
  - For each net, wire length on each layer and total wire length
  - For each net pair, difference in number of vias and wire length by layer

Arguments

None

Example

The following example shows output from `pair_report` for a routed net pair.

```
PairStats:         composite     divide (member nets)  final (all nets)
PairStats:         opens shorts  opens shorts unpair%  opens shorts unpair%  cpu
```

```
PairStats: Summary:    0      0      0      0       1    184      0       1  11.9
#= PAIR ROUTING REPORT =========================================================
#|
#|               Member                   Other
#| Step          Nets   Opens  Shorts     Nets    Opens  Shorts
#|---------------------------------------------------------------
#| start             2      2       0        198    184       0
#|
#|               Composite        Other Nets     Time(sec)
#| Step          Opens  Shorts     Opens  Shorts  Step   Total
#|---------------------------------------------------------------
#| create_topology     1       0       184       0   0.1      0.1
#| route               0       0       184       0  11.6     11.7
#|
#|          Member        Other         Unpaired       Violations  Time(sec)
#| Step       Opens Shorts  Opens Shorts  Length  Percent   (limit)  Step  Total
#|------------------------------------------------------------------------------
#| divide         0      0    184      0    7.04       1%  1 (3.60)   0.1   12.3
#| clean          0      0    184      0    7.04       1%  1 (3.60)   0.1   12.3
```

The following shows contents of the `fat.rpt` file created for the previous example.

```
# = FAT NET OPENS
================================================================================
# | Net                             | Route                           | Length |
# |---------------------------------|---------------------------------|--------|
# |---------------------------------|---------------------------------|--------|

# = PAIRING VIOLATIONS =========================================================
# | Unpaired length limit  3.60                                                |
# |                               |PinPair |     Pairing     |    Unpaired     |
# |  Net        | # Vias | Length |Distance| Best   |  | Length |Percent |
# |-------------|--------|--------|--------|--------|--------|--------|--------|
# |FLASH_1p0[2] |       5| 1156.69|   17.68| 1147.86| 1143.24|    4.62|     0.4|
# |-------------|--------|--------|--------|--------|--------|--------|--------|

# = PAIR STATISTICS ============================================================
# |----------------------------------------------------------------------------|
# | PAIR: 1
# |----------------------------------------------------------------------------|
# | Net: FLASH_1p0[2]
# |----------------------------------------------------------------------------|
# |    Metal1 | #Vias:    0 | Wire Length:    0.00
# |    Metal2 | #Vias:    0 | Wire Length:  452.88
# |    Metal3 | #Vias:    5 | Wire Length:  703.81
# |    Metal4 | #Vias:    0 | Wire Length:    0.00
# |    Metal5 | #Vias:    0 | Wire Length:    0.00
# |    Metal6 | #Vias:    0 | Wire Length:    0.00
# |    Metal7 | #Vias:    0 | Wire Length:    0.00
# |    Metal8 | #Vias:    0 | Wire Length:    0.00
# |    Metal9 | #Vias:    0 | Wire Length:    0.00
# | Total Vias:           5 | Total Wire Length: 1156.69
# |----------------------------------------------------------------------------|
# | Net: FLASH_1p0[1]
# |----------------------------------------------------------------------------|
# |    Metal1 | #Vias:    0 | Wire Length:    0.00
# |    Metal2 | #Vias:    0 | Wire Length:  452.80
# |    Metal3 | #Vias:    5 | Wire Length:  701.77
# |    Metal4 | #Vias:    0 | Wire Length:    0.00
# |    Metal5 | #Vias:    0 | Wire Length:    0.00
# |    Metal6 | #Vias:    0 | Wire Length:    0.00
# |    Metal7 | #Vias:    0 | Wire Length:    0.00
# |    Metal8 | #Vias:    0 | Wire Length:    0.00
# |    Metal9 | #Vias:    0 | Wire Length:    0.00
# | Total Vias:           5 | Total Wire Length: 1154.57
# |----------------------------------------------------------------------------
# | Differences per layer for this pair:
# |----------------------------------------------------------------------------|
# |    Metal1 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal2 | Delta Vias:    0 | Delta Wire Length:    0.08
# |    Metal3 | Delta Vias:    0 | Delta Wire Length:    2.04
# |    Metal4 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal5 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal6 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal7 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal8 | Delta Vias:    0 | Delta Wire Length:    0.00
# |    Metal9 | Delta Vias:    0 | Delta Wire Length:    0.00
```

Related Information

Tcl Command	pair_report_stats

pair_report_stats

```
pair_report_stats
```

Outputs statistics to the Transcript area for net pairs.

The following are reported:

- Number of opens on member nets
- Number of shorts on member nets
- Number of opens on other nets
- Number of shorts on other nets
- Unpaired total length
- Unpaired percentage of total length
- Number of net pair violations and maximum acceptable unpaired length

Arguments

None

Example

The following example shows output from `pair_report_stats` for a routed net pair.

```
|                       P A I R    R E P O R T                           |
|         Member        Other        Unpaired           Violations       |
|         opens  shorts opens shorts Length    Percent     (limit)       |
| ------- ------ ------ ----- ------ --------  -------  ----------  -----|
|              0      0   184      0    12.13        1    1 (3.60)       |
```

Related Information

Tcl Command	pair_report

Preparing for Pair Routing

To create pair routes, you begin by Defining Net Pairs. You can optionally change the required spacing between the pair routes by Customizing Gap Spacing.

Defining Net Pairs

In this step, you identify pairs of memberNets and create a netPairGroup for each pair. A netPairNet is associated with each netPairGroup.

```
create_group -name s_netPairNet -set d_setObjofNets -type net_pair
```

where *s_netPairNet* is the name of the composite net and *d_setObjofNets* is a set containing the memberNets to pair.

The following command creates the composite net, `Net01_pair`, from the net_pair grouping of memberNets `Net0` and `Net1:`

```
create_group -name Net01_pair -set [or_sets -set1 [find_net -silent -name Net0
-no_wildcard true -ignore_case false ] -set2 [find_net -silent -name Net1
-no_wildcard true -ignore_case false ] ] -type net_pair
```

Customizing Gap Spacing

To change the gap between the nets of a pair from the default spacing,

1. Create a constraint group.
2. Add the desired spacing for the layers to the constraint group.
3. Assign the constraint group as the reflexive constraint group for the composite net. The new spacing must be larger than or equal to the foundry `minSpacing`.

For example,

```
create_constraint_group -name GAP
set_layer_constraint -layer Metal1 -group GAP -constraint minSpacing -hardness hard
-Value 0.4
set_layer_constraint -layer Metal2 -group GAP -constraint minSpacing -hardness hard
-Value 0.4
set_layer_constraint -layer Metal3 -group GAP -constraint minSpacing -hardness hard
-Value 0.4
set_layer_constraint -layer Metal4 -group GAP -constraint minSpacing -hardness hard
-Value 0.4
set_constraint_group -reflexive GAP -net_group Net01_pair
```

Examples for Pair Routing Scripts

The following example scripts show the Tcl commands are used to perform pair routing.

```
# pairs.tcl
#
# Use this script to define net pairs.
# Identify net pairs for routing and group each pair using create_group with
# -type net_pair so that they will be recognized by the pair_* commands
# -name for the name of the netPairGroup and the associated composite net.

create_group -name Net01_pair -set [or_sets -set1 [find_net -silent -name Net0
-no_wildcard true -ignore_case false ] -set2 [find_net -silent -name Net1
-no_wildcard true -ignore_case false ] ] -type net_pair
.
.
```

```
# route.tcl
#
# Use this script to read in the design and perform pair routing.
#
# First load the design
read_db -lib sample -cell sample -view layout

# Source the tcl script that defines the net pairs using create_group -type net_pair
source pairs.tcl

# Create the guides for the member nets so that the netPair topology can be created.
# Ignore power, ground and clock nets
update_net_connectivity -all -ignore_types {power ground clock}

# Create the netPairTerms and guides that connect them for the netPairRoutes.
pair_create_topology

# Route the netPairNets and the non-paired nets.
global_route
local_route
croute
detail_route

# Divide the netPairRoutes into memberRoutes. The netPairRoutes and netPairTerms
# are removed.
pair_divide

# Check the connectivity for any remaining opens and shorts.
verify_connectivity -all
```

Related Information

Tcl Command	create_group croute detail_route global_route local_route pair_create_topology pair_divide update_net_connectivity verify_connectivity

Shield Routing Commands

For more information on how to use these commands in a flow, refer to “Examples for Shield Routing Scripts” on page 646.

add_shield_wires

```
add_shield_wires
     [ -net {s_netName…} | -set d_setObj | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -alternate_tie [ true | false ] ]
     [ -coaxial_tie_freq f_userunit ]
     [ -connect_shields_only [ true | false ] ]
     [ -fix_routes [ true | false ] ]
     [ -floating_shields [ true | false ] ]
     [ -report_shield_wires [ true | false ] ]
     [ -redundant_vias [ true | false ] ]
     [ -adj_layer_redundant_ties [ true | false ] ]
     [ -honor_valid_routing_layers [ true | false ] ]
     [ -no_router [ true | false ] ]
     [ -shield_min_length f_userunit ]
     [ -shield_terms [ true | false ] ]
     [ -shield_tie_frequency f_userunit ]
     [ -shield_around_vias [ true | false ] ]
     [ -shield_around_vias_all [ true | false ] ]
     [ -shield_enclose_vias [ true | false ] ]
     [ -shield_maintain_same_net [ true | false ] ]
     [ -tandem [ true | false ] ]
     [ -tie_shield [ true | false ] ]
     [ -ties_use_shield_width [ true | false ] ]
     [ -use_grid {route|mfg} ]
     [ -use_existing_pg_only [ true | false ] ]
     [ -via_width f_userunit ]
     [ -verbose [ true | false ] ]
```

Adds shield wires for the given nets that have been identified by shield_net. By default, vias are added to tie the new shield wires to their respective shield nets. You can optionally specify the maximum distance between ties, the minimum length for signal route segments to add shields to, and whether shielding can be added around vias. On completion, a shield coverage report is output. For more information on the shield coverage report, refer to "report_shield_wires" on page 631.

If you specified dual net parallel shielding using `shield_net -shield_net_alternate`, you can specify the placement of the two shield nets using the `-alternate_tie argument`, as shown in Figure 9-6 on page 629.

Tip

By default, shield wires that cause violations are not added. To identify the constraint setting that is preventing a shield from being added, set the `droute.shields_in_violation` environment variable to `true` before running add_shield_wires, then run the check commands described in Verify Commands.

Arguments

`-adj_layer_redundant_ties [ true | false ]`

By default and when `true`, adds redundant vias to tie shield wires to shield nets where the shield wires overlap their respective existing power/ground rails on adjacent layers only. For example, redundant vias will be added to tie Metal2 shield wires to respective Metal3 power/ground rails but not to Metal6 power/ground rails.

`-alternate_tie [ true | false ]`

(Applies only for dual net parallel shields) If `true`, for a two-point path, one shield net will be on one side of the signal wire, and the second shield net will be on the opposite side. If `false`, one shield net will be east and south of the signal, and the other shield net will be west and north of the signal wire. Refer to Figure 9-6 on page 629 for graphic examples.

Default: `false`

`-coaxial_tie_freq` *`f_userunit`*

Specifies the maximum distance between ties that must be inserted to tie the tandem shield wires and parallel shield wires for coaxial shielding.

`-connect_shields_only [ true | false ]`

If `true`, connects only same net shields. If `false`, shields are tied to the shield net when they are added.

Default: `false`

`-fix_routes [ true | false ]`

If `true`, sets the route status of the generated shields to `fixed`.

Default: The route status for the generated shields are set to `Unfixed`.

`-floating_shields [ true | false ]`

If `true`, preserves the floating shielding wire shapes when they cannot be tied.

Default: `false`

`-report_shield_wires [ true | false ]`

If `true`, reports the shielding coverage for the given design post tie shield.

Default: `false`

`-honor_valid_routing_layers [ true | false ]`

If `true`, shields will be tied off to shapes on valid routing layers only.

Default: `false`

`-net` **{***s_netName…***}**

Adds shield wires for the given nets that have been identified by shield_net.

`-no_router [ true | false ]`

If `true`, no wires will be added to connect shields to terminals but vias can be inserted to connect to shield nets. If `false`, wires and vias can be added to connect shield wires to the shield net and terminals.

Default: `false`

`-redundant_vias [ true | false ]`

Adds redundant vias to tie shield wires to shield nets at every location where the shield wires overlap their respective existing power/ground rails. If power/ground rails overlap the shield wires, this argument offers a cleaner solution for inserting multiple ties than `-shield_tie_frequency`. If none of `-redundant_vias`, `-adj_layer_redundant_ties`, or a tie frequency is specified, only the minimal connections are inserted to tie shield wires to their respective nets.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

Adds shield wires for nets in the area given by the lower left (*f_xlo f_ylo*) and upper right (*f_xhi f_yhi*) coordinates. Operates only nets in the region that have been identified by shield_net.

`-set` *d_setObj*

Adds shield wires for nets in the given set that have been identified by shield_net.

`-shield_around_vias [ true | false ]`

Determines whether shield wires will be added around vias. Refer to "Shielding Around Vias" on page 626 for an example.

Default: `true`

`-shield_around_vias_all [ true | false ]`

When `true`, adds shield wires all around vias even in the wrong direction. Refer to "Shielding Around Vias" on page 626 for an example.

Default: `false`

`-shield_enclose_vias [ true | false ]`

When `true`, adds shield wires all around vias in the preferred direction. Refer to "Shielding Around Vias" on page 626 for an example.

Default: `false`

`-shield_maintain_same_net [ true | false ]`

When `true`, maintains the minimum spacing between shield wires and pins of the same net. When `false`, shield wires are permitted to connect to pins of the same net as the shield.

Default: `true`

`-shield_min_length` *`f_userunit`*

Prevents shielding of any signal wire that is shorter than the given length. By default, Space-based Router and Chip Optimizer will attempt to shield all routes of the signal nets given by `-net`, `-set` or `-region` that have been identified for shielding by shield_net.

`-shield_terms [ true | false ]`

If `true`, will attempt to shield around shape terms.

Default: `false`

`-shield_tie_frequency` *`f_userunit`*

Specifies the maximum distance between ties that must be inserted to tie the new shield wires to their respective shield nets. If this argument is not given and the `redundant_via` argument is not specified, only the minimal connections are inserted to tie shield wires to their respective nets.

`-tandem [ true | false ]`

Adds tandem shields where possible.

Default: `true`

`-tie_shield [ true | false ]`

Specifies whether ties should be added to tie the new shield wires to the shield nets that they belong to. By default, shield ties are added.

If you choose to exclude shield ties (`-tie_shield false)`, you can later add them using route_shield_wires.

`-ties_use_shield_width [ true | false ]`

If `true`, will use the shield width as the width for the tie off connections.

Default: `false`

`-use_existing_pg_only [ true | false ]`

If `true`, uses only existing power/ground shapes for tie off connections.

Default: `false`

`-use_grid {route | mfg}`

Specifies the grid to use when adding shield wires.

Default: `mfg` (manufacturing)

`-verbose [ true | false ]`

By default, total shield coverage by layer is reported. For `-verbose true`, reports shield coverage by layer for each shielded net, in addition to total shield coverage by layer.

`-via_width` *`f_userunit`*

(Applies only to tandem split shields) Specifies the width, in user units, of the vias that will be inserted to tie tandem split shields.

Example

The following command adds shield wires for all nets in the entire design that have been identified for shielding by the shield_net command and ties the added shield wires to the shield net.

```
add_shield_wires
```

The following command adds shield wires for all nets that have been identified for shielding by the shield_net command but restricts the added shielding to those signal wires that are greater than or equal to the `shield_min_length` argument value of 20.

```
add_shield_wires -shield_min_length 20
```

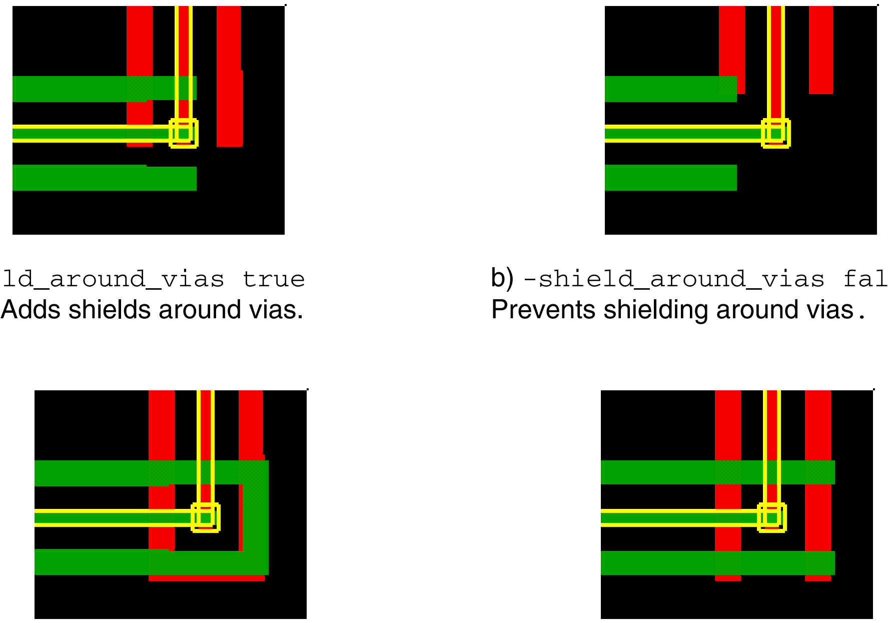
The following command adds shield wires for `net_1` and sets the route status for the added shields to `fixed`.

```
add_shield_wires -net net1 -fix_routes
```

Shielding Around Vias

Figure 9-7 on page 654 illustrates the difference in results when using `-shield_around_vias`, `-shield_around_vias_all`, and `-shield_enclose_vias`. The examples show the general placement of shield wires around the via but do not include shield ties.

Figure 9-5 Examples of Shielding around Vias



a) `-shield_around_vias true`
(Default) Adds shields around vias.

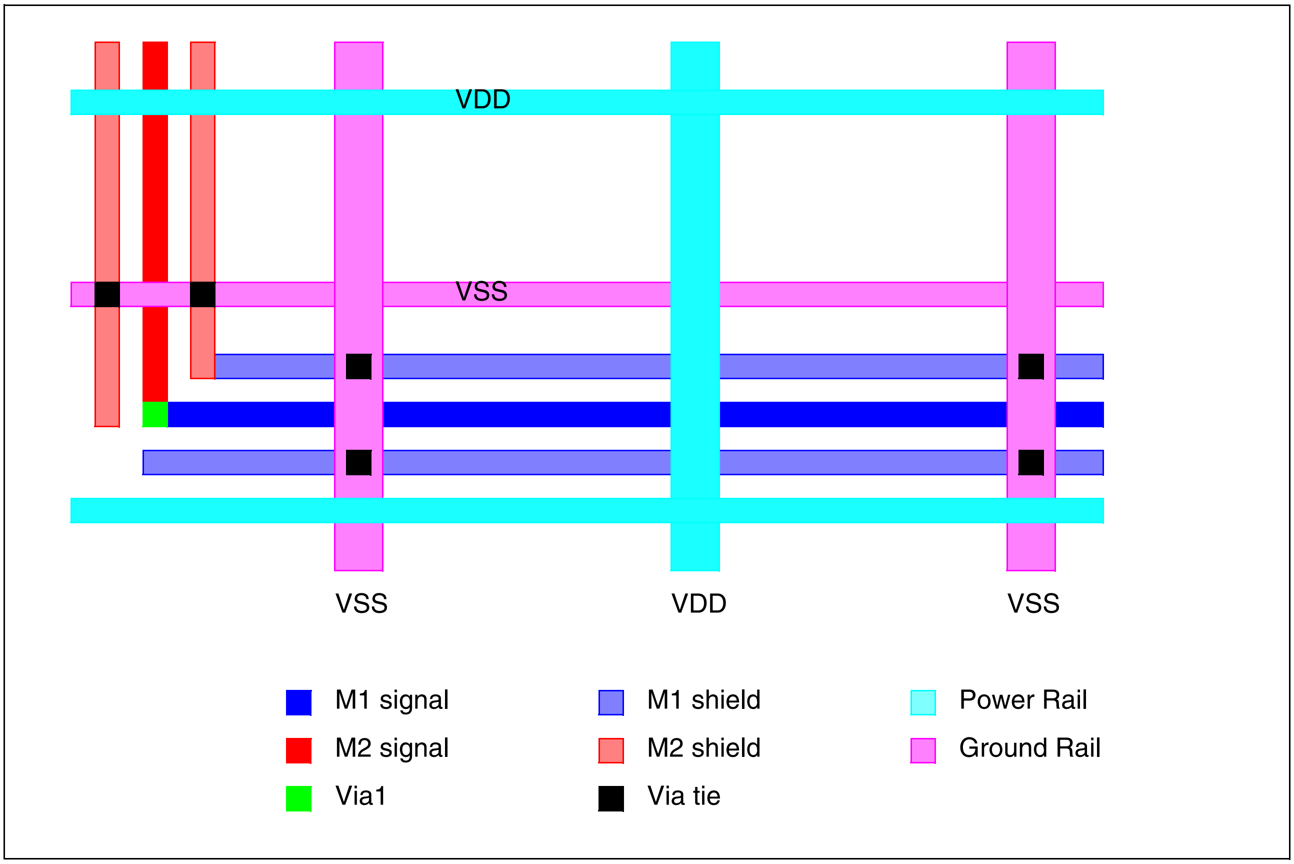
b) `-shield_around_vias false`
Prevents shielding around vias.

c) `-shield_around_vias_all true`
Shields all around vias on the same layer.

d) `-shield_enclose_vias true`
Adds shields all around vias in the preferred direction.

Inserting Shield Ties Using `-redundant_vias`

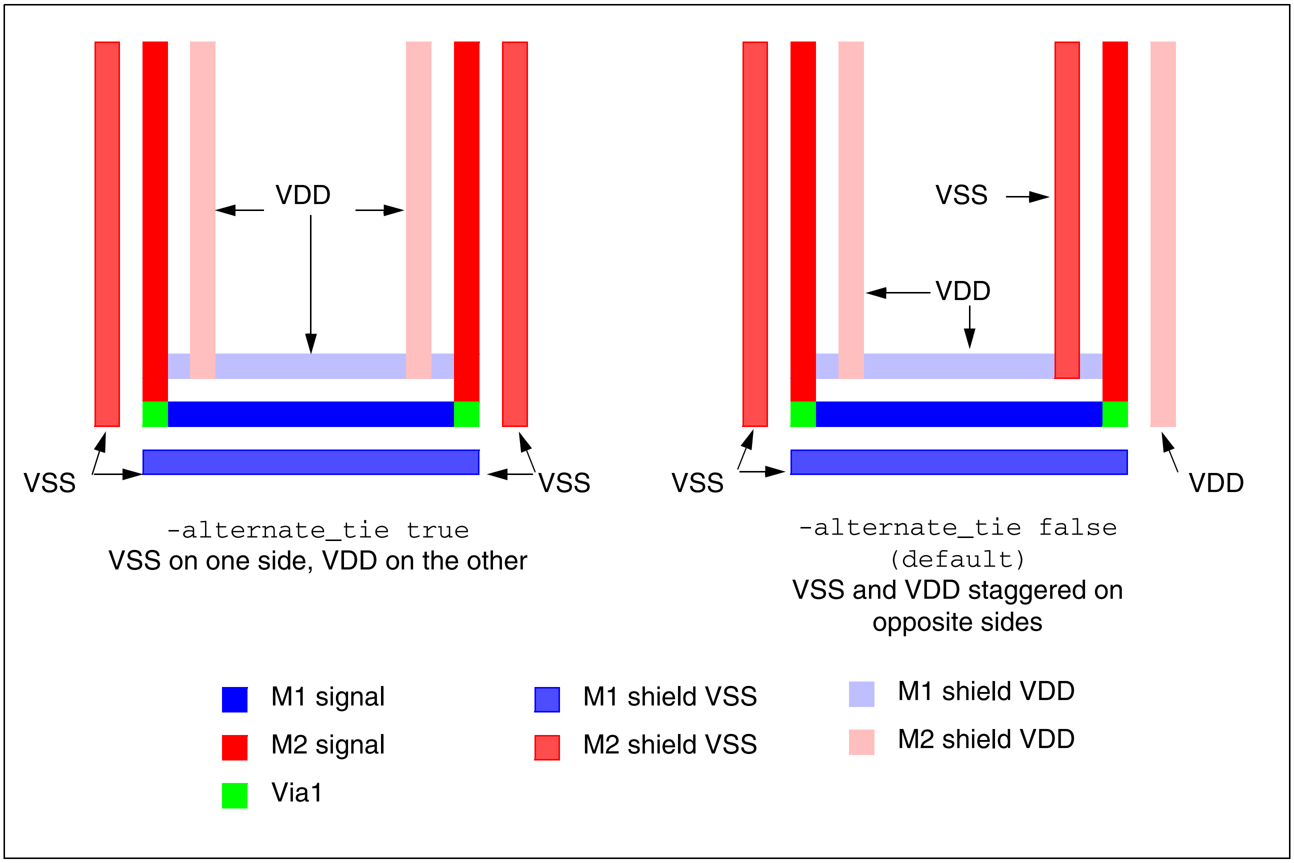
The following example shows how `-redundant_vias` can be used to add via ties at every overlap of the shield wires with their respective power or rail net. In this case, the shield net is VSS (ground).



Dual Net Parallel Shields

The following example shows the difference between the placement of dual net parallel shields when `-alternate_tie` is set `true` or `false`.

Figure 9-6 Dual Net Parallel Shield Placement



Related Information

Tcl Command	delete_shield_wires get_use_existing_shapes_for shielding route_shield_wires set_use_existing_shapes_for_shielding shield_net
Documentation	Refer to "Examples for Shield Routing Scripts" on page 646 for an example of how this command is used in a shield routing flow.

delete_shield_wires

```
delete_shield_wires
     {-all | -net {s_netName…} | -set d_setObj}
```

Removes shield routing (including vias ties) in the entire design, for specific signal nets, or for route segments in the given set.

Arguments

`-all`	Removes shield routing from the entire design.
`-net` **{***`s_netName`*…**}**	Removes shield routing for the signal nets in the list.
`-set` *`d_setObj`*	Removes shield routing for the route segments in the given set.

Example

The following command removes shield routing for `net2`.

```
delete_shield_wires -net net2
```

Related Information

Tcl Command	add_shield_wires

report_shield_wires

```
report_shield_wires
     [ -net {s_netName…} | -set d_setObj | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -verbose [ true | false ] ]
```

Outputs to the transcript area the shield coverage for the given nets, nets in the given set, or shielded nets in the given region. By default, the total shield length and percentage shielded is reported by layer. You can optionally include shield coverage by net and layer (`-verbose`).

Arguments

`-net {`*`s_netName…`*`}` Reports on the given nets.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Reports on shielded nets in the area given by the lower left (*`f_xlo f_ylo`*) and upper right (*`f_xhi f_yhi`*) coordinates.

`-set` *`d_setObj`* Reports on shielded nets in the given set.

`-verbose [ true | false ]`

By default, reports the total shield coverage by layer. For `-verbose true`, reports shield coverage by layer for each shielded net, in addition to the total shield coverage by layer.

Example

The following example reports the shield coverage for the given net.

```
report_shield_wires -net net1
```

route_shield_wires

```
route_shield_wires
     [ -gap_space_for_vias [ true | false ] ]
     [ -adj_layer_redundant_ties [ true | false ] ]
     [ -connect_shields_only [ true | false ] ]
     [ -floating_shields [ true | false ] ]
     [ -report_shield_wires [ true | false ] ]
     [ -honor_valid_routing_layers [ true | false ] ]
     [ -redundant_vias [ true | false ] ]
     [ -ties_use_shield_width [ true | false ] ]
```

Ties shield routes to the shield nets. Use this command following add_shield_wires `-tie_shield false` which adds the shield wires without connectivity to the shield nets and adds guides where connectivity is missing.

When spacing is tight and shield wires meet the gap spacing but the shield tie vias cannot be added in the simplest manner because they would violate shield gap spacing, you can specify `-gap_space_for_vias false`. The router will ignore the shield gap spacing and will add ties only when the minimum spacing for the layer can be met. This is can often be helpful when enclosing vias with preferred direction shields (add_shield_wires `-shield_enclose_vias true`).

Arguments

`-adj_layer_redundant_ties [ true | false ]`

By default and when `true`, adds redundant vias to tie shield wires to shield nets where the shield wires overlap their respective existing power/ground rails on adjacent layers only. For example, redundant vias will be added to tie Metal2 shield wires to respective Metal3 power/ground rails but not to Metal6 power/ground rails.

`-connect_shields_only [ true | false ]`

If `true`, connects only same net shields.

Default: When shields are added, they are tied to the shield net.

`-gap_space_for_vias [ true | false ]`

When `false`, ignores the gap spacing required between vias on shield wires and the wires being shielded. The minimum spacing for the layer must still be met.

Default: Gap spacing must be met when adding shield ties.

`-floating_shields [ true | false ]`

If `true`, preserves the floating shielding wire shapes when they cannot be tied.

Default: `false`

`-report_shield_wires [ true | false ]`

If `true`, reports the shielding coverage for the given design post tie shield.

Default: `false`

`-honor_valid_routing_layers [ true | false ]`

If `true`, shields will be tied off to shapes on valid routing layers only. (Default: `false)`

`-redundant_vias [ true | false ]`

Adds redundant vias to tie shield wires to shield nets at every location where the shield wires overlap their respective existing power/ground rails. If power/ground rails overlap the shield wires, this argument offers a cleaner solution for inserting multiple ties than `-shield_tie_frequency`. If none of `-redundant_vias`, `-adj_layer_redundant_ties`, or a tie frequency is specified, only the minimal connections are inserted to tie shield wires to their respective nets.

`-ties_use_shield_width [ true | false ]`

If `true`, will use the shield width as the width for the tie off connections.

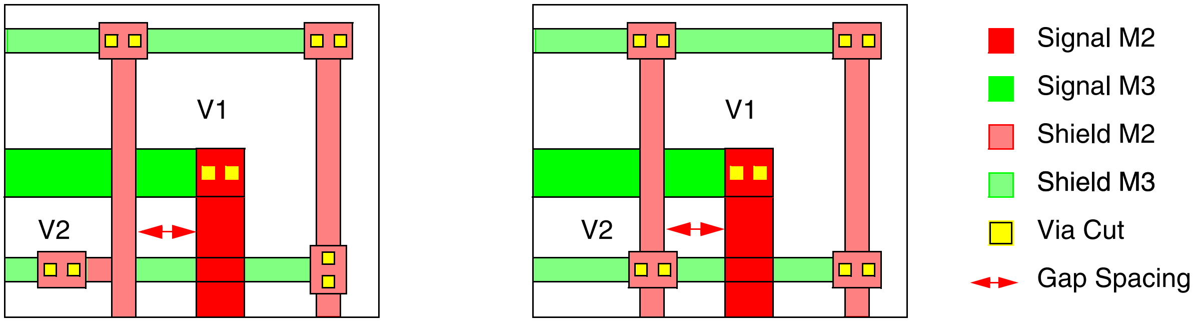
Default: `false`

Example

Usually, you will use add_shield_wires to route shields and add shield ties. Occasionally this might result in extra routing when adding the shield ties, as shown in Figure 9-7 on page 635. In situations like this, you can remove the shields and re-route without adding ties, then use route_shield_wires to add the ties using minimum spacing checking instead of gap spacing.

```
delete_shield_wires -net netA
add_shield_wires -net netA -shield_enclose_vias true -tie_shield false
route_shield_wires -gap_space_for_vias false
```

Figure 9-7 Example for route_shield_wires Ignoring Gap Spacing



a) Shield ties added during add_shield_wires must meet gap spacing requirements. In this example, additional wiring is needed to place shield tie V2 when preferred direction shields are used to enclose Via V1.

b) When shield ties are added using `route_shield_wires -gap_space_for_vias false`, shield tie V2 can be placed without extra wiring, as long as the minimum spacing is met. (minimum spacing <= gap spacing)

Related Information

Tcl Command	add_shield_wires

set_route_noshield

```
set_route_noshield
     -set d_setObj
     -no_shield [ true | false ] ]
```

Enables or disables shielding on a routes in the set. This command must be issued before running add_shield_wires.

Arguments

`-no_shield [ true | false ]`	
	If set to `false`, prevents shields from being added to the routes in the set. By default and when set to `true`, enables shielding on the routes in the set. This argument sets the `noShield` property on the route.
`-set` *`d_setObj`*	Operates on routes in the set.

Example

The following example prevents shields from being added for the routes in the set.

```
set_route_noshield -set [get_selection_set] -noShield true
```

Related Information

Tcl Command	add_shield_wires

shield_net

```
shield_net
     {-net {s_netName…} | -set d_setObj}
     -shield_net s_netName
     [ -routes [ true | false ] ]
     [ -shield_net_alternate s_netName ]
     [ -silent [ true | false ] ]
     [ -type {parallel | tandem | coaxial | tandem_split} ]
```

Assigns a shield net to a signal net, and optionally excludes shielding on specific layers by setting constraints. This command flags the signal nets for which the router must reserve additional clearance. By default, the shields will be parallel, unless a specific type is given.

You can choose to use two different nets as parallel shields (`-shield_net_alternate`), for example, power and ground. In this case, one parallel shield will be power and the other will be ground. When two nets are used for parallel shields, you cannot use the existing shapes for shielding or shield sharing. To control the position of the shield nets, use `add_shield_wires -alternate_tie`.

Before you run this command, you must set the constraints needed for shielding. For more information on this procedure, refer to “Controlling Shield Options” on page 641.

When this command is run, a composite net comprising the signal net and the shield net is created.

A warning message is issued by this command if the taper constraint group assigned to the net is the same as the net’s default constraint group.

```
    shielded <netName> has the same default and taper constraint groups.
```

If this condition exists, tapering will not be performed. If routing fails, you can change the name of either the taper or the default constraint group and re-route. For example, if both constraint groups are defaulting to the `LEFDefaultRouteSpec`, use this procedure:

```
create_constraint_group -name temp_spec -transient true
set_constraint_group -shield s_shieldCG -default temp_spec -net s_netName
```

Because the taper and default constraint group names are now different, the tapering software will be enabled.

Use find_shielded_nets to get a set of nets that are or will be shielded, optionally limited to a shield type.

Arguments

`-net` **{***s_netName…***}**

Flags the given signal nets for shielding.

`-routes [ true | false ]`

When set to `true`, the `shieldNet` property information is added to each route on the net. Use this with imported data to identify the shield nets.

Default: `false`

`-set` *`d_setObj`*

Flags the signal nets in the given set for shielding.

`-shield_net` *`s_netName`*

Specifies the name of the net to use for shielding. For dual net parallel shields, this net is used for one of the shields and the net given by -`shield_net_alternate` is used for the shield on the other side.

`-shield_net_alternate` *`s_netName`*

Specifies the second net for dual net parallel shields.

Important

When you set this argument, you cannot share shields (`shareShields` constraint must be `false` or not set), and you must disable the use of existing shapes.

```
set_use_existing_shapes_for_shielding -share false
```

`-silent [ true | false ]`

If set to `true`, informational messages will be suppressed.

Default: `false`

`-type` *`s_shieldType`*

Specifies the type of shield to create for the given signal nets.

Note: You can also set the shield type by setting the msShieldStyle constraint. For more information, see Setting the Shield Type.

`coaxial`	Both tandem and parallel shields to surround the signal wire on four sides
`parallel`	Planar shield wires parallel to the signal wire
`tandem`	Shield wires on a given layer above and below the signal wire

`tandem_split`	Coaxial shields with tandem shields above and below the parallel shields

Example

The following command flags `net2` for parallel shielding by the `GND` net. The shield wires will be added when the add_shield_wires command is issued.

```
shield_net -net net2 -shield_net GND
```

Related Information

Tcl Command	add_shield_wires
Documentation	Refer to "Examples for Shield Routing Scripts" on page 646 for an example of how this command is used in a shield routing flow.

unshield_net

```
unshield_net
     {-net {s_netName…} | -set d_setObj}
```

Removes the shield net flag from one or more signal nets.

Note: This command does not remove existing shielding around a net, but prevents add_shield_wires from adding shielding to the given nets. Use delete_shield_wires to remove shielding.

Arguments

`-net` **{***`s_netName`***…}**	Removes the shield net flag from the named signal nets.
`-set` *`d_setObj`*	Removes the shield net flag from the signal nets in the given set.

Example

The following command removes the shield net flag for `net2` and `net3`.

```
unshield_net -net {net2 net3}
```

Related Information

Tcl Command	shield_net

Controlling Shield Options

The following constraints are used when shielding nets:

Constraint Name	Constraint Group	Description (Constraint Type, Value Type)
`minWidth`	Shield	(Layer, Value) Width of parallel shield wires and tandem split shields
`minSpacing`	Shield	(Layer, Value) Spacing between the parallel shield wires and the net it is shielding and between tandem shields and tandem split shields
`tandemWidth`	Shield	(Layer, Value) Width of tandem shield wires
`tandemLayerAbove`	Shield	(Layer, LayerValue) Layer above the given signal layer to use for tandem shields
`tandemLayerBelow`	Shield	(Layer, LayerValue) Layer below the given signal layer to use for tandem shields
`shareShields`	Design	(Simple, BoolValue) Determines whether shield wires can be shared between nets
`ignoreShieldingOnLayers`	Default	(Simple, LayerArrayValue) Prevents shields from being added on the given layers
`msShieldStyle`	Shield	(Simple, IntValue) Determines the type of shielding to route around a net.

The following sections describe how to configure shields and set shielding options:

- Configuring Shield Wires
- Using Existing Power and Ground Shapes for Shielding
- Sharing Shields
- Preventing Shielding on Specific Layers
- Disabling Shielding on a Route

- Shielding by Route
- Setting the Shield Type

Configuring Shield Wires

To uniquely define the shield constraints for a net, do the following:

1. Create a new constraint group:

   ```
   create_constraint_group -name s_shieldGroupName
   ```

2. Assign the new constraint group as the shield group for the nets that you want to shield.

   ```
   set_constraint_group -shield s_shieldGroupName -net s_netName
   ```

3. Add constraints for the appropriate layers to the new constraint group.

   ❑ Specify `minWidth` and `minSpacing` if you are adding **parallel shields** and/or **tandem split shields**.

   ```
   set_layer_constraint -layer s_layerName -constraint minWidth -Value f_value
   -group s_shieldGroupName
   ```

   ```
   set_layer_constraint -layer s_layerName -constraint minSpacing -Value f_value
   -group s_shieldGroupName
   ```

   ❑ Specify `tandemWidth`, `tandemLayerAbove`, and `tandemLayerBelow` if you are adding **tandem shields**.

   ```
   set_layer_constraint -layer s_layerName -constraint tandemWidth -Value
   f_value -group s_shieldGroupName
   ```

   ```
   set_layer_constraint -layer s_layerName -constraint tandemLayerAbove
   -LayerValue s_layerName -group s_shieldGroupName
   ```

   ```
   set_layer_constraint -layer s_layerName -constraint tandemLayerBelow
   -LayerValue s_layerName -group s_shieldGroupName
   ```

   ❑ Specify tandem **and** parallel shield constraints, given above, if you are adding **coaxial shields**.

If constraints are not set in the shield group for the net, the shields will inherit the constraint settings from the signal net, the default, design, and foundry group, in that order. You can also specify unique shield options for individual routes in the net as described in "Shielding by Route" on page 644.

Using Existing Power and Ground Shapes for Shielding

By default, existing power and ground shapes are used to shield nets. You can disable this feature with the following command:

```
set_use_existing_shapes_for_shielding -share false
```

When this feature is disabled, the router will reserve space around nets for shield wires where required.

To get the current status, use the following command:

```
get_use_existing_shapes_for_shielding
```

Sharing Shields

Usually, the router will attempt to individually shield each net. If there is a channel between two nets that is not sufficiently large to add individual shields for each net and `shareShields` is set, one shield wire will be added between the two nets and shared by the nets.

To permit nets in a net group to share shield routes whenever possible, use the following:

```
create_group{-set d_setOfNets | -net {s_netName…}} -name s_groupName -type group
set_constraint -constraint shareShields -BoolValue true -group s_groupName
```

To permit all shielded nets to share shield routes whenever possible, use the following:

```
set_constraint -constraint shareShields -BoolValue true
```

This setting makes the `shareShields` constraint a global setting, applied to the design route spec (`catenaDesignRules`). When using the global setting, you should not specify a different group.

Preventing Shielding on Specific Layers

In some cases, you might want to exclude shielding on some layers. For example, on more congested layers it might be difficult to add shielding. To prevent Space-based Router and Chip Optimizer from shielding on those layers, you can set the `ignoreShieldingOnLayers` constraint with the layers that should be ignored.

```
set_constraint -constraint ignoreShieldingOnLayers -group s_groupName
-LayerArrayValue {s_layerName…}
```

where *`s_groupName`* is the name of the design constraint group or the default constraint group for the net. For example, the following sets `ignoreShieldingOnLayers` for the `single` constraint group and prevents the router from adding shields on the `M1` and `M2` layers for the nets with the `single` constraint group assigned as the default constraint group.

```
set_constraint -name ignoreShieldingOnLayers -group single -LayerArrayValue {M1
M2}
```

Disabling Shielding on a Route

To prevent shielding on a route of a net to be shielded, use set_route_noshield before adding shield wires.

Shielding by Route

In most cases, you will configure shields by assigning a constraint group to the net to be shielded. Space-based Router and Chip Optimizer also lets you specify unique shield settings for routes of the net, allowing you to change options, such as shield spacing or width, on specific routes by creating another constraint group and assigning it those routes. This is especially helpful when shielding cannot be added along some routes due to insufficient space. Reducing width or spacing requirements along those routes might allow them to be shielded, while keeping stricter requirements on the rest of the net. You can also use this method to create thicker shields for specific routes for signal integrity issues.

To configure a net with different shield settings on some routes,

1. Create a constraint group and assign it to the net to be shielded as described in "Configuring Shield Wires" on page 642.
2. Create another constraint group.

   ```
   create_constraint_group -name s_shieldGroupName2
   ```
3. Assign the constraint group from step 2 to the routes that should have shield settings that are different from the rest of the net.

   One way to do this is to set the Routing Object Granularity to *Entire Route*, then interactively select the route, or use `find_by_area` to add the route to the selection set.

   ```
   set_active -object { "route" } -active true
   set_active -object { "net" "connected_shapes" } -active false
   replace_set -set1 [find_by_area -single -region {f_xlo f_ylo f_xhi f_yhi}] \
     -set2 [get_selection_set]
   set_constraint_group -shield s_shieldGroupName2 -set [get_selection_set]
   ```
4. Add constraints for the routes to the new constraint group as in step 3 of "Configuring Shield Wires" on page 642.

   The router will apply the appropriate constraints for the net and individual routes when adding the shield wires.

Setting the Shield Type

There are two ways to control the shield type:

- Set the `shield_net` `-type` argument when assigning the shield net to the net to be shielded.

  ```
  shield_net {-net s_netName | -set d_setID} -shield_net s_shield -type s_type
  ```

- Set the msShieldStyle constraint for the shield constraint group.

  ```
  shield_net {-net s_netName | -set d_setID} -shield_net s_shield -type s_type
  set_constraint -constraint msShieldStyle -group s_shieldGroup -IntValue
  i_shieldStyle
  ```

  Use this method for interoperability with Cadence® Innovus™ flows and when you want to specify a shield type that cannot be set using `shield_net` `-type`. You must first run `shield_net` to assign the shield net to the net to be shielded, then set the shield type for the shield constraint group by using the msShieldStyle constraint. For an example, see Tandem Above Shield Routing Script.

Important

When you set the shield type by using the msShieldStyle constraint, you must set the correct `shield_net` `-type`, according to the following table:

Table 9-1 Mapping msShieldStyle to shield_net -type

msShieldStyle	shield_net -type	Description
1	`parallel`	Only parallel (sides)
2	`tandem`	Only tandem below
3	`coaxial`	Tandem below and parallel
4	`tandem`	Only tandem above
5	`coaxial`	Tandem above and parallel
6	`tandem`	Tandem above and below
7	`coaxial`	Coaxial (Tandem above and below, parallel)
15	`tandem_split`	Tandem split (Split, tandem above and below, parallel)

Nets are shielded according to the shield type that was last set by either method before adding the shields.

Examples for Shield Routing Scripts

Parallel Shield Routing Script

The following example script shows the Tcl commands used to perform parallel shield routing.

```
# Use this script to read in the design and perform shield routing.

read_db -lib sample -cell sample -view layout

# Set shield constraints by creating a new constraint group

create_constraint_group -name single_shield

# assign the new constraint group as the 'shield' and 'default' constraint groups
#for the nets to be shielded

set_constraint_group -shield single_shield -default single -net net_1
set_constraint_group -shield single_shield -default single -net net_3

# Set minWidth and minSpacing for shields

set_layer_constraint -layer M1 -constraint minWidth -Value .2 -group
single_shield
set_layer_constraint -layer M2 -constraint minWidth -Value .2 -group
single_shield
set_layer_constraint -layer M3 -constraint minWidth -Value .2 -group
single_shield
set_layer_constraint -layer M1 -constraint minSpacing -Value .16 -group
single_shield
set_layer_constraint -layer M2 -constraint minSpacing -Value .2 -group
single_shield
set_layer_constraint -layer M3 -constraint minSpacing -Value .2 -group
single_shield

# Permit nets to share shields by setting the shareShields constraint
# to true

set_constraint -constraint shareShields -BoolValue true

# Identify the signal nets and their respective shield nets.
# This must be done to flag the signal nets for which the router must allot
# additional clearance. When no shield type (-type) is given, the type defaults to
# parallel shields.

shield_net -net net_1 -shield_net GND
shield_net -net net_3 -shield_net GND

# Route the design.

global_route
local_route
croute
detail_route

# Add the parallel shield wires and, by default, shield ties.

add_shield_wires
```

```
# Check the connectivity.
verify_connectivity -all
```

Coaxial Shield Script

The following example script shows the Tcl commands used to perform coaxial shield routing.

```
# Use this script to read in the design and perform coaxial shield routing.

read_db -lib sample -cell sample -view layout

# Set shield constraints by creating a new constraint group

create_constraint_group -name double_shield

# assign the new constraint group to the nets to be shielded
# the 'double' constraint group is already defined

set_constraint_group -shield double_shield -default double -net net_1
set_constraint_group -shield double_shield -default double -net net_3

# Set parallel shield constraints

set_layer_constraint -layer M1 -constraint minWidth -Value .2 -group
double_shield
set_layer_constraint -layer M2 -constraint minWidth -Value .2 -group
double_shield
set_layer_constraint -layer M3 -constraint minWidth -Value .2 -group
double_shield
set_layer_constraint -layer M4 -constraint minWidth -Value .4 -group
double_shield
set_layer_constraint -layer M5 -constraint minWidth -Value .4 -group
double_shield
set_layer_constraint -layer M1 -constraint minSpacing -Value .16 -group
double_shield
set_layer_constraint -layer M2 -constraint minSpacing -Value .2 -group
double_shield
set_layer_constraint -layer M3 -constraint minSpacing -Value .2 -group
double_shield
set_layer_constraint -layer M4 -constraint minSpacing -Value .2 -group
double_shield
set_layer_constraint -layer M5 -constraint minSpacing -Value .2 -group
double_shield

# Set tandem shield constraints

set_layer_constraint -layer M1 -constraint tandemWidth -Value .6 -group
double_shield
set_layer_constraint -layer M2 -constraint tandemWidth -Value .6 -group
double_shield
set_layer_constraint -layer M3 -constraint tandemWidth -Value .6 -group
double_shield
set_layer_constraint -layer M4 -constraint tandemWidth -Value .4 -group
double_shield
set_layer_constraint -layer M5 -constraint tandemWidth -Value .4 -group
double_shield
set_layer_constraint -layer M1 -constraint tandemLayerAbove -LayerValue M2 -group
```

```
double_shield
set_layer_constraint -layer M2 -constraint tandemLayerAbove -LayerValue M3 -group
double_shield
set_layer_constraint -layer M2 -constraint tandemLayerBelow -LayerValue M1 -group
double_shield
set_layer_constraint -layer M3 -constraint tandemLayerAbove -LayerValue M4 -group
double_shield
set_layer_constraint -layer M3 -constraint tandemLayerBelow -LayerValue M2 -group
double_shield
set_layer_constraint -layer M4 -constraint tandemLayerAbove -LayerValue M5 -group
double_shield
set_layer_constraint -layer M4 -constraint tandemLayerBelow -LayerValue M3 -group
double_shield
set_layer_constraint -layer M5 -constraint tandemLayerBelow -LayerValue M4 -group
double_shield

# Identify the signal nets and their respective shield nets.
# This must be done to flag the signal nets for which the router must allot
# additional clearance.

shield_net -net net_1 -shield_net GND -type coaxial
shield_net -net net_3 -shield_net GND -type coaxial

# Route the design.

global_route
local_route
croute
detail_route

# Add the parallel and tandem shield wires and, by default, shield ties.

add_shield_wires

# Check the connectivity.
verify_connectivity -all
```

Tandem Above Shield Routing Script

The following example script shows the Tcl commands used to perform tandem shield routing only on the above layer.

```
# Use this script to read in the design and perform shield routing.

read_db -lib sample -cell sample -view layout

# Set shield constraints by creating a new constraint group

create_constraint_group -name top_shield

# assign the new constraint group to the nets to be shielded
# the 'single' constraint group is already defined

set_constraint_group -shield top_shield -default single -net net_1

# Set tandem above shield constraints

set_layer_constraint -layer M1 -constraint tandemWidth -Value .6 -group top_shield
set_layer_constraint -layer M2 -constraint tandemWidth -Value .6 -group
top_shield
set_layer_constraint -layer M3 -constraint tandemWidth -Value .6 -group
top_shield
set_layer_constraint -layer M4 -constraint tandemWidth -Value .4 -group
top_shield
set_layer_constraint -layer M5 -constraint tandemWidth -Value .4 -group
top_shield
set_layer_constraint -layer M1 -constraint tandemLayerAbove -LayerValue M2 -group
top_shield
set_layer_constraint -layer M2 -constraint tandemLayerAbove -LayerValue M3 -group
top_shield
set_layer_constraint -layer M3 -constraint tandemLayerAbove -LayerValue M4 -group
top_shield
set_layer_constraint -layer M4 -constraint tandemLayerAbove -LayerValue M5 -group
top_shield

# Identify the signal net and its shield net.
# This must be done to flag the signal nets for which the router must allot
# additional clearance.

shield_net -net net_1 -shield_net GND -type tandem

# IntValue 4 is for Tandem above only.
# Setting the msShieldStyle constraint sets the shield style, replacing the
# shield_net -type setting. IMPORTANT: The shield_net -type must match the shield
# type category of the msShieldStyle value, as shown in Table 9-1.

set_constraint -constraint msShieldStyle -hardness hard -IntValue 4 -group
top_shield

# Route the design.

global_route
local_route
croute
detail_route

# Add the tandem shield wires and, by default, shield ties.

add_shield_wires

# Check the connectivity.
verify_connectivity -all
```

Related Information

Tcl Command	add_shield_wires create_constraint_group croute detail_route global_route local_route set_constraint set_constraint_group set_layer_constraint shield_net verify_connectivity

Matched Length Routing Commands

Matched length routing is used to control net lengths to meet timing and delay requirements.

check_length

```
check_length
     [ -do_composite [ true | false ] ]
     [ -file s_fileName ]
     [ -match_paths [ true | false ] ]
     [ -min_segment_length f_micron ]
     [ -report ]
     [ -set d_setObj ]
     [ -sink_driver [ true | false ] ]
```

Reports length violations based on the settings of the following constraints:

- `routeMinLength`

  When this constraint is set, errors are reported on nets with routes that are shorter than `routeMinLength`.

- `matchTolerance`

  (Applies only for net groups of type `net_match`) Errors are reported on nets that are shorter than the longest net of the group minus the `matchTolerance`. Lengths are based on the net's routes. If both `matchTolerance` and `msTolerance` are set, `matchTolerance` is used. However, for interoperability, you should only use `msTolerance`, not `matchTolerance`. The default setting is two times the pitch or four times the gap space is used for the tolerance.

- `msMatchPerLayer`

  For net groups of type `net_match`, when this constraint is set to `true`, length checks are performed by layer.

- `msTolerance`

  (Applies only for net groups of type `net_match`) Errors are reported on nets that are shorter than the longest net of the group minus the `msTolerance` percentage. Lengths are based on the net's routes. If both `matchTolerance` and `msTolerance` are set, `msTolerance` is ignored. However, for interoperability, you should only use `msTolerance`, not `matchTolerance`. The default setting is two times the pitch or four times the gap space is used for the tolerance.

  For example, if `msTolerance` is `10`, or 10%, and the longest net `is` `20` microns, then a net length of 18.2 microns would be acceptable because it is greater than the longest net minus 10% of 20 (2 microns). A net length less than 18 microns would fail.

The following information is given in the report:

Group	Name of the net’s net group
Net	Name of the net
Layer	Name of the layer (`msMatchPerLayer` checks only)
Delta	Difference between net’s route length and either the `routeMinLength` (if set), or the route length of the longest net of the group (Constraint-Length)
Length	The net’s route length
Constraint	`routeMinLength` (if set), or the route length of the longest net of the group
Tolerance	Absolute tolerance from `matchTolerance` (if set) or `msTolerance` (if set and `matchTolerance` is not set), otherwise two times the routing pitch or four times the gap space is used

Important

Nets must be detail routed before using this command.

Arguments

`-do_composite [ true | false ]`

If set to `true`, checking is performed at the composite level for net pair groups.

Default: `false`

`-file` *`s_fileName`* Outputs results to the named file.

`-match_paths [ true | false ]`

When `true`, will report errors on route paths of the same net between terminals.

Default: `false`

`-min_segment_length` *`f_micron`*

Only checks segments that are longer than this length.

`-report` Reports on all nets that are evaluated.
By default, only violating nets are reported.

`-set` *`d_setObj`* Specifies the set of nets or composite nets to operate on. By default and when this argument is not given, all nets in the design will be processed.

`-sink_driver [ true | false ]`

When `true`, will report errors on routes of the same net between sink-driver terminals.

Default: `false`

Example

The following example creates a net group of all nets whose names start with `MATCHB`, then detail routes the nets and checks the lengths of the nets against each other.

```
update_net_connectivity -all
set MATCHBset [find_net -name MATCHB*]
create_group -name MATCHB -set $MATCHBset -type net_match

# create a constraint group for the net_match group and initialize the new
# constraint group with the constraints from the LEFDefaultRouteSpec
create_constraint_group -name GM -type group2group
set_constraint_group -default GM -net_group MATCHB
copy_constraint -group LEFDefaultRouteSpec -to_group GM

# route the nets
detail_route
```

```
# check lengths
check_length -set $MATCHBset
```

The following example shows reported errors:

```
LENGTH ERROR: Group MATCHB Net MATCHB1 Delta 33.00 Length 39.50 Constraint 72.50
Tolerance 2.40
LENGTH ERROR: Group MATCHB Net MATCHB0 Delta 7.10 Length 65.40 Constraint 72.50
Tolerance 2.40
CHECK LENGTH: 2 Errors
2
```

Related Information

Tcl Commands	fix_length

fix_length

```
fix_length
     [ -set d_setObj]
     [ -accordion_height f_micron ]
     [ -detour [ true | false ] ]
     [ -do_composite [ true | false ] ]
     [ -match_paths [ true | false ] ]
     [ -min_segment_length f_micron ]
     [ -multi_layer [ true | false ]  [ -triple_layer [ true | false ] ] ]
     [ -report [ true | false ] ]
     [ -reset_max_length f_micron ]
     [ -shift_pattern [ true | false ] ]
     [ -sink_driver [ true | false ] ]
     [ -start_distance f_micron ]
     [ -tune_pattern [ true | false ] ]
```

Controls the length of individual nets and/or matches the length of groups of nets, including differential pair nets, or groups of routes within a tolerance.

Important

Nets must be detail routed before using this command.

Two types of length fixing are supported and each has specific requirements:

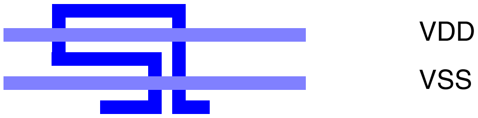
- To fix lengths on individual nets, you must first set the `routeMinLength` constraint.
- To match net lengths relative to each other, you must first create the net group of nets and, optionally, set the `msTolerance` (or `matchTolerance`) and `msMatchPerLayer` constraints. Wire is added to the shorter nets to match the longest net of the group.

To match lengths at the composite level for groups of net pairs, use `-do_composite true -multi_layer true.`

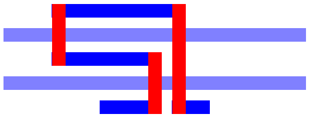
Elongation methods can also be customized:

- To prevent lengthening on a specific layer, set the `lengthPatternOff` constraint to `true` for the layer.
- To restrict the use of an elongation pattern, set the constraint for the pattern (`lengthPatternEndRun`, `lengthPatternAccordion`, `lengthPatternRWAccordion`, or `lengthPatternTrombone`) to `false`. Otherwise, all of these patterns can be used.
- To permit elongation by adding dangles, set lengthPatternDangle to the desired dangle style. If this constraint is not set, dangles are not used for elongation.

- To use more than one metal layer for elongation patterns, use `-multi_layer true`. This can offer flexibility for routing around existing wires, as shown in the following figure. By default, a single layer is used for elongation patterns. Use `-multi_layer true -triple_layer true` to use up to three metal layers for elongation.

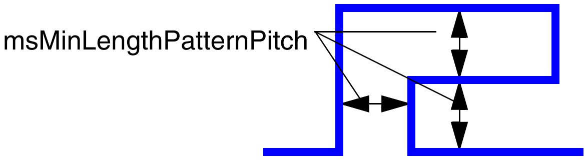


By default, a single layer is used for an elongation pattern. In this case, the elongation would not be routed because power rails on the same layer would obstruct the path of the elongation pattern.

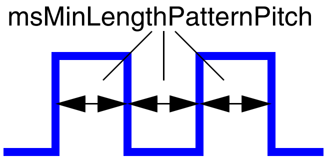


When `-multi_layer` is set to `true`, multiple layers can be used to route the elongation pattern.

- To specify the same net spacing for elongation patterns, use the msMinLengthPatternPitch constraint.



msMinLengthPatternPitch applied to Trombone-style elongation



msMinLengthPatternPitch applied to Accordion-style elongation

Arguments

`-accordion_height` *`f_micron`*

Controls the height (in microns) of an accordion pattern, measured centerline to centerline.

`-detour [ true | false ]`

If set to `true`, will attempt to use advanced techniques for lengthening. This can result in more detouring of wiring, and should only be used when normal lengthening techniques (`-detour false`) have failed.

Default: `false`

`-do_composite [ true | false ]`

If set to `true`, length matching is performed at the composite level for net pair groups. To be effective, this must be done before running pair_divide.

Default: `false`

`-match_paths [ true | false ]`

When `true`, will attempt to match routes on the same net between terminals.

Default: `false`

`-min_segment_length` *`f_micron`*

Prevents lengthening of segments that are shorter than or equal to this length.

`-multi_layer [ true | false ]`

If set to `true` with `-triple_layer false`, then elongation patterns can be generated in up to two metal layers. This is useful when wires, such as the power mesh, have already been routed and can obstruct elongation routing on a single layer. When set to `false`, elongation patterns can use only a single metal layer.

Default: `false`

`-report`

When set to `true`, reports on all nets that are evaluated.

Default: (`false`) Only violating nets are reported.

`-reset_max_length` *`f_micron`*

By default, the algorithm is iterative and permits length to be added to the longest net in order to converge. This argument limits the added length, past the initial routed length of the longest net, to the given value.

`-set` *`d_setObj`*

Operates on the net groups or nets in the set. If a net in the set is a member of a net group, all nets in the net group are operated on. By default,

- If this argument is not given and net groups of type net_match exist, all nets in the net match groups are processed.
- All nets for which routeMinLength is set will be processed.

`-shift_pattern [ true | false ]`

When `true`, permits patterns to be shifted to avoid minimum spacing violations with oversized vias.

Default: `false`

`-sink_driver [ true | false ]`

When `true`, will attempt to match routes on the same net between sink-driver terminals.

Default: `false`

`-start_distance` *`f_micron`*

Prevents elongation patterns from being inserted within this distance of pins and macros.

`-triple_layer [ true | false ]`

If set to `true` with `-multi_layer true`, then elongation patterns can be generated in up to three metal layers. This is useful when existing routing is congested and obstruct elongation routing. When set to `false`, elongation patterns cannot use three metal layers.

Default: `false`

`-tune_pattern [ true | false ]`

When `true`, post-process will attempt to meet length requirements by elongating accordion bumps. When `false`, post-processing is not performed.

Default: `true`

Example

The following scripts are included in this section:

- Relative Length Matching of Nets in a Set
- Controlling the Length of a Single Net Using `routeMinLength`
- Controlling the Lengths of Nets in a Set Using `routeMinLength`
- Relative Length Matching of Net Pair Groups
- Matching Lengths of Net Pairs Using Minimum and Maximum Route Lengths

Relative Length Matching of Nets in a Set

The following example creates a net group of all nets whose names start with `MATCHB`, then detail routes the nets, and finally matches the lengths of the nets by elongating the shorter nets, within a tolerance of 1.0.

```
update_net_connectivity -all
set MATCHBset [find_net -name MATCHB*]
create_group -name MATCHB -set $MATCHBset -type net_match

# create a constraint group for the net_match group and initialize the new
# constraint group with the constraints from the LEFDefaultRouteSpec
create_constraint_group -name GM -type group2group
set_constraint_group -default GM -net_group MATCHB
copy_constraint -group LEFDefaultRouteSpec -to_group GM

# Set the tolerance. If this constraint is not set, the default tolerance is used.
set_constraint -group GM -constraint matchTolerance -Value 1.0

# route the nets
detail_route

# match/fix lengths
fix_length -set $MATCHBset
```

Controlling the Length of a Single Net Using `routeMinLength`

The following example sets `routeMinLength` and `routeMaxLength` constraints for `netA`, checks the length of `netA`, then lengthens it, without allowing elongation on the `Metal2` layer.

```
update_net_connectivity -all
detail_route
create_constraint_group -name GA
set_constraint_group -default GA -net netA
copy_constraint -group LEFDefaultRouteSpec -to_group GA
set_layer_constraint -layer Metal2 -group GA -constraint lengthPatternOff
-BoolValue true
set_constraint -group GA -constraint routeMinLength -Value 75.0
set_constraint -group GA -constraint routeMaxLength -Value 77.0
check_length
fix_length
```

Controlling the Lengths of Nets in a Set Using `routeMinLength`

The following example sets `routeMinLength` and `routeMaxLength` constraints for the nets in the selected set, checks the lengths of the nets, then lengthens them, as needed.

```
update_net_connectivity -all
detail_route
create_constraint_group -name GA
set_constraint_group -default GA -set [get_selection_set]
copy_constraint -group LEFDefaultRouteSpec -to_group GA
set_constraint -group GA -constraint routeMinLength -Value 75.0
set_constraint -group GA -constraint routeMaxLength -Value 77.0
check_length -set [get_selection_set]
fix_length -set [get_selection_set]
```

Relative Length Matching of Net Pair Groups

The following example performs relative length matching at the composite level for two net pairs.

```
# define the net pairs
set c1nets [or_sets -set1 [find_net -name clk1n] -set2 [find_net -name clk1p]]
set c2nets [or_sets -set1 [find_net -name clk2n] -set2 [find_net -name clk2p]]
create_group -name c1_pair -set $c1nets -type net_pair
create_group -name c2_pair -set $c2nets -type net_pair

# create a net group for matching the net pairs
create_group -name clkGroup -set [find_group -name c*_pair -type net_pair] -type
net_match

# create the constraint group
create_constraint_group -name GC

# assign the constraint group to the pairs
set_constraint_group -default GC -net_group c2_pair
set_constraint_group -default GC -net_group c1_pair
set_constraint_group -default GC -net_group clkGroup

# set constraint values by copying from existing, then setting indiv constraints
copy_constraint -group LEFDefaultRouteSpec -to_group GC

set_constraint -group GC -constraint lengthPatternRWAccordion -BoolValue false
set_constraint -group GC -constraint lengthPatternAccordion -BoolValue false
# set the allowed Tolerance: all nets must be no more than 10% shorter than the
# longest net length
set_constraint -group GC -constraint msTolerance -FltValue 10

# set same net spacing to power pitch
set_layer_constraint -layer M8 -group GC -constraint msMinLengthPatternPitch
-hardness hard -Value 9.0
set_layer_constraint -layer M7 -group GC -constraint msMinLengthPatternPitch
-hardness hard -Value 9.0

# disable elongation on all but M7 and M8
foreach layer {M1 M2 M3 M4 M5 M6 M9 MD} {
  set_layer_constraint -layer $layer -constraint lengthPatternOff -BoolValue true
-group GC}
set_constraint -group GC -constraint validRoutingLayers -hardness hard
-LayerArrayValue {M7 M8}
set_constraint -group GC -constraint limitRoutingLayers -hardness hard
-LayerArrayValue {M7 M8}

# Create the net pair topology
pair_create_topology

# route the composite nets
global_route
croute
detail_route

# match lengths on nets relative to the length of the longest net
fix_length -do_composite -multi_layer true

# divide composite nets, preserving the topology of the composite net
pair_divide -preserve_topology true
```

Matching Lengths of Net Pairs Using Minimum and Maximum Route Lengths

The following example performs length matching at the composite level for two net pairs whose lengths must be greater than or equal to `routeMinLength` and less than or equal to `routeMaxLength`.

```
# define the net pairs
create_group -name c1_pair -set [or_sets -set1 [find_net -name clk1n -set2
[find_net -name clk1p] -type net_pair
create_group -name c2_pair -set [or_sets -set1 [find_net -name clk2n -set2
[find_net -name clk2p] -type net_pair

# create the constraint group
create_constraint_group -name GC

# assign the constraint group to the pairs
set_constraint_group -default GC -net_group c2_pair
set_constraint_group -default GC -net_group c1_pair

# set constraint values by copying from existing, then setting indiv constraints
copy_constraint -group LEFDefaultRouteSpec -to_group GC

# set absolute min and max lengths
set_constraint -group GC -constraint routeMinLength -Value 1472
set_constraint -group GC -constraint routeMaxLength -Value 1490
set_constraint -group GC constraint lengthPatternRWAccordion -BoolValue false
set_constraint -group GC constraint lengthPatternAccordion -BoolValue false

# set same net spacing to the power pitch
set_layer_constraint -layer M8 -group GC -constraint msMinLengthPatternPitch
-hardness hard -Value 9.0
set_layer_constraint -layer M7 -group GC -constraint msMinLengthPatternPitch
-hardness hard -Value 9.0

# disable elongation on all but M7 and M8
foreach layer {M1 M2 M3 M4 M5 M6 M9 MD} {
  set_layer_constraint -layer $layer -constraint lengthPatternOff -BoolValue true
-group GC}
set_constraint -group GC -constraint validRoutingLayers -hardness hard
-LayerArrayValue {M7 M8}
set_constraint -group GC -constraint limitRoutingLayers -hardness hard
-LayerArrayValue {M7 M8}

# Create the net pair topology
pair_create_topology

# route the composite nets
global_route
croute
detail_route

# match lengths on nets using absolute routeMinLength and routeMaxLength
fix_length -do_composite -multi_layer true

# divide composite nets, preserving the topology of the composite net
pair_divide -preserve_topology true
```

Related Information

Tcl Commands	check_length

Strand Routing Commands

Strand routing commands in this section are used to do the following:

- Operate on wide nets that have already been routed with the normal flow (split_net)
- Route nets using multiple strands between pins (strand_route)

split_net

```
split_net
     {-all | net {s_netName…} | -set d_setObj}
     -width f_userunit
```

Splits a routed wide width net into individual segments of the given width. This is useful when the routed wire width is greater than the maximum width for the design or when strand routing can give a better result for current load and spacing requirements. The net is split to maximize the number of strands within the wide wire bounds.

Arguments

`-all`	Splits all nets in the design with wire widths greater than the given width (`-width`).
`-net` **{***s_netName…***}**	Splits all nets in the list with wire widths greater than the given width (`-width`).
`-set` *d_setObj*	Splits all nets in the set with wire widths greater than the given width (`-width`).
`-width` *f_userunit*	Splits nets with wire widths greater than this value into multiple strands of this width.

Example

The following command splits nets wider than 0.3 microns into stranded routes with wires of width 0.3 microns.

```
split_net -all -width 0.3
```

strand_route

```
strand_route
     { -all [ true | false ]
     | -set d_setObj
     | -nets {s_netName…} }
     [ -num_strands i_count ]
     [ -strand_spacing f_distance ]
     [ -strand_width f_width ]
     [ -max_cluster_distance f_clusterDistance ]
     [ -enable_bus_route [ true | false ] ]
     [ -strap_strands [ true | false ] ]
```

Routes all nets, specified nets, or nets in a set using multiple strands (fingers) between pins. In addition, stranded routing supports all connection types, including fat-to-fat, one-to-many (or many-to-one), and many-to-many, in all patterns, including straight, L, and Z patterns. For pattern path-find method, currently stranded routing only works for many-to-many connection type. For other connection types, a message is displayed.

Note: Stranded routing provides the pattern path-finding method to generate connections if it does not work with straight, L, or Z patterns.

If any of the nets to be routed are in a `net_strand` constraint group, then `numStrands`, `strandSpacing`, `strandWidth`, `maxClusterDistance,` and `clusterDistance` constraints, if set for the constraint group, are used for strand routing. Alternatively, the number of strands can be explicitly specified using command arguments, or automatically maximized based on the strand width (`minWidth` for the layer) and the strand spacing (via-to-via pitch). Command arguments can be used to override constraint settings.

Note: When both `clusterDistance` and `maxClusterDistance` constraints are defined, the `clusterDistance` constraint is used. However, when only one of them is defined, the one that is defined is used for strand routing.

Strand routing produces parallel strands that are equal in width and evenly spaced. New wiring that fails this criterion (for example, due to obstructions) is discarded. If `-enable_bus_route` is `true` and strand routing fails, the router attempts to route strands as a bus. While this method can be successful, the resulting strands might not be parallel and evenly spaced.

Arguments

`-all [ true | false ]`

Strand routes all nets in the design.

`-enable_bus_route [ true | false ]`

When `true`, will route strands as a bus if search-based pattern route fails.

Default: `false`

`-max_cluster_distance` *`f_clusterDistance`*

Specifies the maximum distance, in microns, between adjacent pins in a cluster.

Default: For nets in a `net_strand` group, the `maxClusterDistance` constraint value is used, if set; otherwise, 10 * maximum via-to-via pitch for valid routing layers is used.

`-nets {`*`s_netName`*`…}`

Strand routes the specified nets.

`-num_strands` *`i_count`*

Routes the specified number of strands.

Default: For nets in a `net_strand` group, the `numStrands` constraint value will be used, if set; otherwise, the maximum possible strands is based on `-strand_spacing` and `-strand_width`.

`-set` *`d_setObj`*

Strand routes the nets in the given set.

`-strand_spacing` *`f_distance`*

Specifies the exact spacing, in microns, between strands.

Default: For nets in a `net_strand` group, the `strandSpacing` constraint value is used, if set; otherwise, the strand spacing is the via-to-via pitch.

`-strand_width` *`f_width`*

Specifies the width, in microns, for individual strands.

Default: For nets in a `net_strand` group, the `strandWidth` constraint value will be used, if set; otherwise, the strand width is the `minWidth` for the layer.

`-strap_strands [ true | false ]`

This is useful for multi-layer strands. For graphic examples, see Figure 9-8.

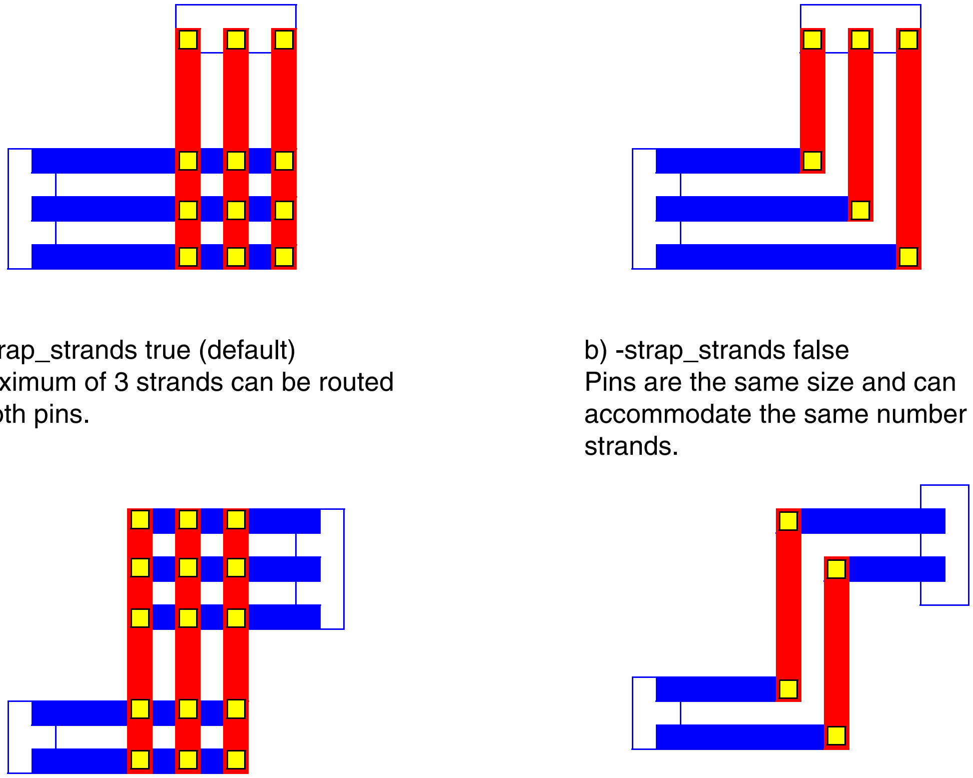
`true`	(Default) If the number of strands is not specified, then the number of strands routed is dependent on the size of the larger pin.
`false`	If the number of strands is not specified, then the number of strands routed is dependent on the size of the smaller pin.

Examples

In this example, the fat pin-to-fat pin nets in the selected set are strand routed using the maximum number of strands that can be accommodated.

```
create_group -set [get_selection_set] -type net_strand -name StrandGroup
strand_route -set [get_selection_set]
```

Figure 9-8 Example of strap_strands Argument



a) -strap_strands true (default)
A maximum of 3 strands can be routed for both pins.

b) -strap_strands false
Pins are the same size and can accommodate the same number of strands.

c) -strap_strands true
The maximum number of strands that can be routed is dependent on the size of the larger pin.

d) -strap_strands false
The maximum number of strands that can be routed is dependent on the size of the smaller pin.

Related Information

Tcl Commands	create_group

Preparing for Strand Routing

To strand route nets, do the following:

1. (Optional) Identify the nets and create a `net_strand` group of those nets using `create_group.`

    ❑ For nets in a set, use

    ```
    create_group -name s_groupName -type net_strand -set d_setObj
    ```

    ❑ To specify the nets by name, use

    ```
    create_group -name s_groupName -type net_strand -net {s_netName…}
    ```

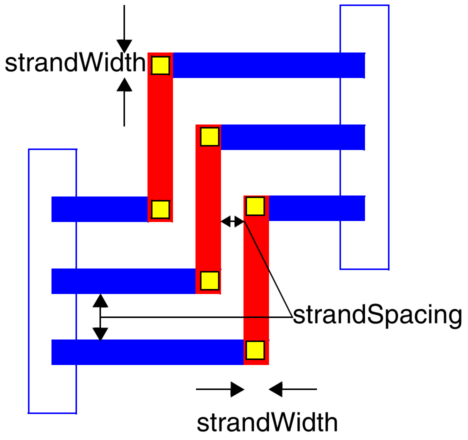
Important

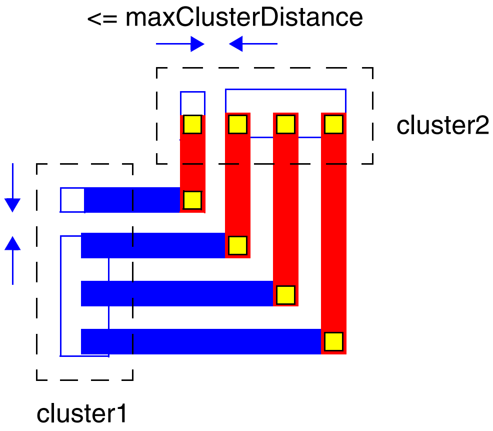
This step is required only if strand constraints will be used.

2. (optional) Create a constraint group containing strand routing constraints for the nets.

Table 9-2 Strand Routing Constraints

Constraint Name	Description
numStrands	This optional constraint specifies the number of strands that must be routed for strand nets. If this value is not given, the strand router will attempt to route the maximum number of strands based on the `strandSpacing` and `strandWidth` constraints.
strandSpacing	This optional constraint specifies the required exact spacing between strands, in microns, for the layer. If this constraint is not set, via-to-via pitch is used.
strandWidth	This optional constraint specifies the required width, in microns, for individual strands on the layer. If this constraint is not set, `minWidth` for the layer is used.
maxClusterDistance	This optional constraint determines the maximum distance, in microns, between two adjacent pins in a cluster. Strand routing can only route two clusters per net. Nets with greater than two clusters will not be routed. If this constraint is not set, 10*via-to-via pitch (maximum for all valid routing layers) is used.
clusterDistance	This optional constraint specifies that pins spaced less than or equal to the distance from each other on a multi-pin net are clustered together for strand routing.
limitRoutingLayers	Limits routing to the specified layers. Set this constraint for single layer strands.





3. Route the nets using strand_route.

Example for Strand Routing Using Constraints

In this example,

- The `net_strand` group, `StrandGroup`, is created for the selected nets.
- The `strandRuleSpec` constraint group is created.
- The strand constraints are set and added to `strandRuleSpec`.
- `strandRuleSpec` is assigned to the `StrandGroup` net group.
- The selected nets are strand routed with 3 strands per net.

```
create_group -set [get_selection_set] -type net_strand -name StrandGroup
create_constraint_group -name strandRuleSpec -type userdefined
set_constraint -constraint numStrands -IntValue 3 -group strandRuleSpec
set_layer_constraint -layer M2 -constraint strandWidth -Value 0.1 \
  -group strandRuleSpec
set_layer_constraint -layer M2 -constraint strandSpacing -Value 0.2 \
  -group strandRuleSpec
set_layer_constraint -constraint maxClusterDistance -layer M2 -Value 0.5 \
  -group strandRuleSpec
assign_constraint_group -net_group StrandGroup -group strandRuleSpec
strand_route -set [get_selection_set]
```

10

ECO Route Commands

Cadence® Space-based Router and Chip Optimizer supports Engineering Change Order (ECO) operations to modify and re-route sections of the design. ECOs are commonly used to move cells, remaster cells, and add or delete buffers to improve timing and signal integrity. Space-based Router and Chip Optimizer lets you run *what-if* scenarios by performing a sequence of ECO commands, and saving or discarding the changes based on your evaluation of the results.

For examples of typical ECO operations in this chapter, refer to “ECO Routing Examples” on page 698, which includes methods for Routing the ECO Changes.

The ECO commands are presented in alphabetic order:

- eco_begin on page 675
- eco_commit on page 676
- eco_connect_inst_term on page 677
- eco_connect_term on page 678
- eco_copy_implementation on page 679
- eco_create_instance on page 680
- eco_create_net on page 682
- eco_destroy_instance on page 683
- eco_destroy_net on page 684
- eco_disconnect_inst_term on page 685
- eco_disconnect_term on page 686
- eco_move_instance on page 687
- eco_remaster_instance on page 688
- eco_repair_net on page 689

eco_begin

```
eco_begin
     [ -no_undo [ true | false ] ]
```

Starts an ECO command sequence and returns the ECO control for the sequence. The ECO control is passed by other ECO commands to identify the command sequence that they apply to. An eco_commit or eco_undo terminates the command sequence.

Arguments

`-no_undo [ true | false ]`

Determines whether changes in this ECO command sequence can be undone.

Default: `false` (Changes can be undone.)

Value Returned

d_ecoObj	Is the ECO control for the command sequence.

Example

The following example sets the `eco` variable to the ECO control value for a new command sequence. This variable is passed to ECO commands to identify the sequence to add the command to.

```
set eco [eco_begin]
```

Related Information

Tcl Commands	eco_undo

eco_commit

```
eco_commit
     -eco_control d_ecoObj
     [ -no_repaint ]
```

Saves the work associated with the given ECO control and terminates the command sequence.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-no_repaint`	Prevents the artwork from being automatically updated as `eco` commands in the process are run.

Example

The following command saves the work associated with the `eco` control variable and terminates the command sequence.

```
eco_commit -eco_control $eco
```

Related Information

Tcl Commands	eco_undo

eco_connect_inst_term

```
eco_connect_inst_term
     -eco_control d_ecoObj
     -term_name s_termName
     -net_name s_netName
     { -inst_name s_instName | -object d_instObj }
     [ -no_repaint ]
```

Connects the given net to the given instance terminal.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-inst_name` *`s_instName`*	Is the name of the instance.
`-net_name` *`s_netName`*	Is the name of the net to connect the instance terminal to.
`-no_repaint`	Prevents the artwork from being run when this command is run.
`-object` *`d_instObj`*	Is the object identifier for the instance.
`-term_name` *`s_termName`*	Is the name of the terminal to connect to the net.

Example

The following example connects the instance terminal given by the terminal name `A` and instance name `I2` to the net `n2`.

```
eco_connect_inst_term -eco_control $eco -inst_name I2 -term_name A -net_name n2
```

Related Information

Tcl Commands	eco_connect_term eco_disconnect_inst_term eco_disconnect_term

eco_connect_term

```
eco_connect_term
     -eco_control d_ecoObj
     -net_name s_netName
     { -term d_termObj
     | -term_name s_termName -lib s_libName -cell s_cellName -view s_viewName }
     [ -no_repaint ]
```

Connects a net to a terminal.

Arguments

`-cell` *`s_cellName`*	Specifies the cell name.
`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-lib` *`s_libName`*	Specifies the library name.
`-net_name` *`s_netName`*	Is the name of the net to connect the terminal to.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-term` *`d_termObj`*	Specifies the object identifier for the terminal to connect.
`-term_name` *`s_termName`*	Is the name of the terminal to connect to the net.
`-view` *`s_viewName`*	Specifies the view name.

Example

Related Information

Tcl Commands	eco_connect_inst_term eco_disconnect_inst_term eco_disconnect_term

eco_copy_implementation

```
eco_copy_implementation
     -eco_control d_ecoObj
     -from_net s_netName
     -to_net s_netName
     [ -no_repaint ]
```

Copies all routes in one net to another net.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-from_net` *`s_netName`*	Copies routes from the named net.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-to_name` *`s_netName`*	Copies routes to the named net.

Example

The following command copies all routes from net `n1` to net `n2`.

```
eco_copy_implementation -eco_control $eco -from_net n1 -to_net n2
```

Related Information

Tcl Commands	eco_repair_net

eco_create_instance

```
eco_create_instance
     -eco_control d_ecoObj
     -lib s_libName -cell s_cellName -view s_viewName
     -origin {f_x f_y}
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
     [ -name s_instName ]
     [ -no_repaint ]
```

Creates a new instance of the master given by `lib/cell/view` and the origin for the instance.

Arguments

`-cell` *`s_cellName`*	Specifies the cell name for the master.
`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-lib` *`s_libName`*	Specifies the library name for the master.
`-name` *`s_instName`*	Is the name for the instance. The default is "".
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90}`	Specifies the orientation for the instance. The default is `R0`.
`-origin` **{***`f_x f_y`***}**	Specifies the x and y coordinates for the instance origin.
`-view` *`s_viewName`*	Specifies the view name for the master.

Example

The following command creates an instance of `des_lib/BUF2/abstract`.

```
eco_create_instance -eco_control $eco -lib des_lib -cell BUF2 -view abstract
-origin {0 390} -name new_buf
```

Related Information

Tcl Commands	eco_connect_inst_term eco_destroy_instance

eco_create_net

```
eco_create_net
     -eco_control d_ecoObj
     -name s_netName
     [ -no_repaint ]
```

Creates a new net. To establish connectivity, you must use eco_connect_inst_term and eco_connect_term.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-name` *`s_netName`*	Is the name for the net.
`-no_repaint`	Prevents the artwork from being updated when this command is run.

Example

The following command creates a net `n2`.

```
eco_create_net -eco_control $eco -name n2
```

Related Information

Tcl Commands	eco_destroy_net

eco_destroy_instance

```
eco_destroy_instance
     -eco_control d_ecoObj
     { -name {s_instName…} | -object d_instObj | -set d_setObj }
     [ -no_repaint ]
```

Removes an instance and all routes incident to it.

Arguments

`-eco_control` *d_ecoObj*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-name` **{***s_instName…***}**	Specifies the names of the instances to remove.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-object` *d_instObj*	Is an object identifier for the instance to be removed.
`-set` *d_setObj*	Is a set identifier for the set of instances to be removed.

Example

The following command removes an instance `I2`.

```
eco_destroy_instance -eco_control $eco -name I2
```

Related Information

Tcl Commands	eco_create_instance

eco_destroy_net

```
eco_destroy_net
     -eco_control d_ecoObj
     {-name {s_netName…} | -object d_netObj | -set d_setObj}
     [ -no_repaint ]
```

Removes a net by removing all routes and steiners.

Important

You must disconnect terminals from the net before using `eco_destroy_net`.

Arguments

`-eco_control d_ecoObj`	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-name {s_netName…}`	Is the name of the nets to be removed.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-object d_netObj`	Is an object identifier for the net to be removed.
`-set d_setObj`	Is a set identifier for the set of nets to be removed.

Example

The following commands disconnect terminals from a net, then destroy the net.

```
eco_disconnect_inst_term -eco_control $eco -inst_name B -term_name A
eco_disconnect_inst_term -eco_control $eco -inst_name I2 -term_name Y
eco_destroy_net -eco_control $eco -name n2
```

Related Information

Tcl Commands	eco_disconnect_inst_term eco_disconnect_term

eco_disconnect_inst_term

```
eco_dicconnect_inst_term
     -eco_control d_ecoObj
     -term_name s_termName
     { -inst_name s_instName | -object d_instObj }
     [ -no_repaint ]
```

Disconnects the given instance terminal from its net.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-inst_name` *`s_instName`*	Is the name of the instance.
`-no_repaint`	Prevents the artwork from being run when this command is run.
`-object` *`d_instObj`*	Is the object identifier for the instance.
`-term_name` *`s_termName`*	Is the name of the terminal to disconnect from the net.

Example

The following command disconnects the `Y` terminal from the instance `I2`.

```
eco_disconnect_inst_term -eco_control $eco -inst_name I2 -term_name Y
```

Related Information

Tcl Commands	eco_disconnect_term

eco_disconnect_term

```
eco_disconnect_term
     -eco_control d_ecoObj
     { -net_name s_netName -term_name s_termName
     | -term d_termObj }
     [ -no_repaint ]
```

Disconnects a terminal from its net.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-net_name` *`s_netName`*	Disconnects the given terminal from this net.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-term` *`d_termObj`*	Specifies the object identifier for the terminal to be disconnected.
`-term_name` *`s_termName`*	Used with `-net_name`, specifies the name of the terminal to be disconnected from the net.

Example

The following command disconnects the `rst_n` term from net `reset`.

```
eco_disconnect_term -eco_control $eco -term_name rst_n -net_name reset
```

Related Information

Tcl Commands	eco_disconnect_inst_term

eco_move_instance

```
eco_move_instance
     -eco_control d_ecoObj
     { -object d_instObj | -name s_instName }
     [ -origin {f_x f_y} ]
     [ -orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
     [ -no_repaint ]
```

Moves an instance to the given location and/or orientation.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-name` *`s_instName`*	Is the name of the instance to be moved.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-object` *`d_instObj`*	Is the object identifier for the instance to be moved.
`-orient {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90}`	
	Specifies the new orientation of the instance. The default is `the` current orientation.
`-origin` **{***`f_x f_y`***}**	Specifies the new x and y coordinates for the instance origin. The default is the current origin.

Example

The following command moves instance `I2` to a new location with the same orientation.

```
eco_move_instance -eco_control $eco -name I2 -origin {0 396}
```

Related Information

Tcl Commands	eco_create_instance

eco_remaster_instance

```
eco_remaster_instance
     -eco_control d_ecoObj
     -lib s_libName -cell s_cellName -view s_viewName
     { -name s_instName | -object d_instObj }
     [ -no_repaint ]
```

Changes the master of an instance.

Arguments

`-cell` *`s_cellName`*	Specifies the cell name for the new master.
`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-lib` *`s_libName`*	Specifies the library name for the new master.
`-name` *`s_instName`*	Is the name of the instance to be remastered.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-object` *`d_instObj`*	Specifies the object identifier for the instance to be remastered.
`-view` *`s_viewName`*	Specifies the view name for the new master.

Example

The following command changes the master of instance `I2`.

```
eco_remaster_instance -eco_control $eco -lib macroLib -cell BUFX1 -view abstract
-name I2
```

Related Information

Tcl Commands	eco_create_instance

eco_repair_net

```
eco_repair_net
     -eco_control d_ecoObj
     -name s_netName
     [ -no_repaint ]
```

Repairs a net by updating connectivity to create guides and trimming routes appropriately.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-name` *`s_netName`*	Is the name of the net to be repaired.
`-no_repaint`	Prevents the artwork from being updated when this command is run.

Example

The following command repairs net `n2`.

```
eco_repair_net -eco_control $eco -name n2
```

Related Information

Tcl Commands	eco_copy_implementation

eco_route

```
eco_route
     [ -set d_setObj | -net {s_netName …} ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]}
     [ -fix_errors [ true | false ] ]
     [ -pre_route_repair [ true | false ] ]
     [ -remove_violating_fills [ true | false ] ]
     [ -threads i_threads ]
     [ -verbose [ true | false ] ]
```

Identifies nets with opens in the target group of nets (in the entire top cellview, in a given set, or in a given list), and completes routing on those nets according to the design rules. By default, this process includes the following steps:

- Pre-ECO route repair

  Resolves existing shorts and spacing violations by removing violating routes and repairing connectivity on the target nets.

- Connectivity update

  Identifies target nets with opens as ECO nets.

- ECO route

  Routes ECO nets identified in the previous step.

- Fix errors

  Resolves existing shorts and spacing violations on the ECO nets.

- Remove violating fill shapes

  Removes fill shapes (floating fill used for metal density correction and connected fill used for power and ground optimization) that cause DRC violations on the ECO nets, and removes `gapFill` shapes that are no longer physically connected to their respective nets.

During processing, status messages are output to the Transcript area and include the following information:

- The number of pre-route repairs performed
- The number of nets in the design and the number of target nets
- The number of nets changed as a result of ECO routing at each stage.

Arguments

`-exclude_net` **{***`s_netName`* …**}**

Excludes the named nets from processing.

`-exclude_set` *`d_setObj`*

Excludes nets in the given set from processing.

`-exclude_type` **{**`[power][ground][clock]`**}**

Excludes nets of the specified types from processing.

`-fix_errors [ true | false ]`

Runs `fix_errors` only on nets affected by ECO routing performed by this command.

Default: `true`

`-net` **{***`s_netName`*…**}**

Operates only on nets in the list. If neither `-set` nor `-net` is given, all nets in the top cell view with opens are processed.

`-pre_route_repair [ true | false ]`

Prepares the design for ECO routing: deletes pre-existing violating routes, updates and repairs connectivity, removes loops and dangles.

Default: `true`

`-remove_violating_fills [ true | false ]`

When set `true`, removes fill shapes that cause DRC violations for ECO nets and removes `gapFill` shapes that are no longer physically connected to their respective net.

Default: `false`

`-set` *`d_setObj`*

Operates only on nets in the set. If neither `-set` nor `-net` is given, all nets in the top cell view with opens are processed.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-verbose [ true | false ]`

When set `true`, outputs additional information, listing the nets that are modified at each stage of ECO routing.

Example

The following example routes nets in the selected set that have opens.

```
eco_route -set [get_selection_set]
```

eco_set_net_preferred_layers

```
eco_set_net_preferred_layers
     { -all | -net s_netName | -set d_setObj }
     -layers {s_layerName…}
     [ -eco_control d_ecoObj ]
     [ -no_repaint ]
```

Sets the preferred layers for one or more nets.

Arguments

`-all`	Sets the preferred layers for all nets.
`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-layers` **{***`s_layerName…`***}**	Specifies the preferred layers.
`-net` *`s_netName`*	Is the name of the net for which the preferred layers are set.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-set` *`d_setObj`*	Sets the preferred layers for nets in the given set.

Example

The following command sets the preferred layers for net `n2`.

```
eco_set_net_preferred_layers -eco_control $eco -net n2 -layers {M2 M3 M4}
```

Related Information

Tcl Commands	eco_create_net

eco_set_net_rule

```
eco_set_net_rule
     { -all | -net s_netName | -set d_setObj }
     -route_spec s_routeSpec
     [ -eco_control d_ecoObj ]
     [ -no_repaint ]
```

Assigns the given route spec to one or more nets.

Arguments

`-all`	Assigns the route spec to all nets.
`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command. This command is processed immediately but results are not saved until eco_commit is issued.
`-net` *`s_netName`*	Assigns the route spec to the given net.
`-no_repaint`	Prevents the artwork from being updated when this command is run.
`-route_spec` *`s_routeSpec`*	
	Specifies the name of the route spec.
`-set` *`d_setObj`*	Assigns the route spec to nets in the given set.

Example

The following command assigns the `double_wide` route spec to net `n2`.

```
eco_set_net_rule -eco_control $eco -route_spec double_wide -net n2
```

Related Information

Tcl Commands	eco_create_net

eco_undo

```
eco_undo
     -eco_control d_ecoObj
     [ -no_repaint ]
```

Reverses all operations done by the command sequence given by the ECO control and terminates the sequence.

Arguments

`-eco_control` *`d_ecoObj`*	Specifies the command sequence ECO control that was assigned by the originating command.
`-no_repaint`	Prevents the artwork from being updated when this command is run.

Example

```
eco_undo -eco_control $eco
```

Related Information

Tcl Commands	eco_commit

get_changed_area

```
get_changed_area
     [ -halo f_micron ]
     [ -tcl_list [ true | false ] ]
```

Creates annotations for the rectangular areas within which wiring has changed during an ECO command sequence. You must first issue an `eco_begin` command before making changes that you want to track, then use this command to create annotations for the changed areas as layer `annotation:viaOpt` objects that will be listed by layer as `Changed area:ECO changed area` annotations in the Optimizations page of the Annotation Browser.

Arguments

`-halo` *`f_micron`*

Expands changed regions by the halo value and creates a single annotation where expanded regions merge and overlap, instead of individual annotations for those regions.

Default: 0

`-tcl_list [ true | false ]`

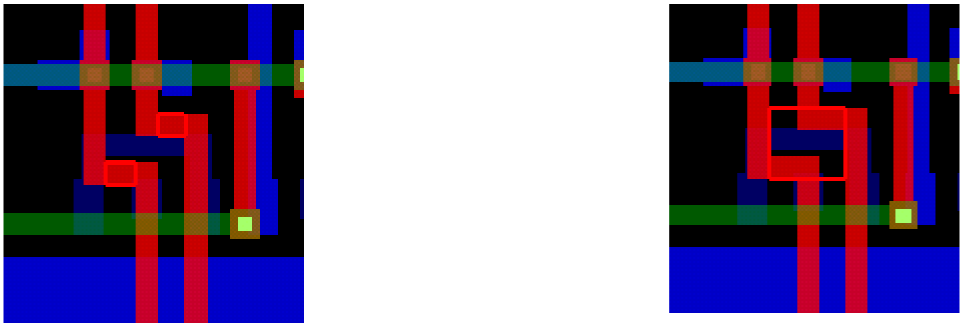
Returns a Tcl list that specifies the areas changed for specific layers in the following format:

```
{{{s_layerName} {f_xlo f_ylo f_xhi
     f_yhi}…}…}
```

Example

Figure 10-1 on page 697 shows the difference in results when the -halo argument is specified.

Figure 10-1 Difference in Results When -halo Is Specified



Case 1
Two annotations indicate where two segments were added.

Case 2
When -halo is specified that causes expanded regions to overlap, one annotation is created for a merged, overlapping region.

Related Information

Tcl Command	check_litho_errors eco_begin read_check_area read_litho_errors

ECO Routing Examples

The following examples illustrate how the ECO commands can be used to perform netlist changes and commands to use to finish routing:

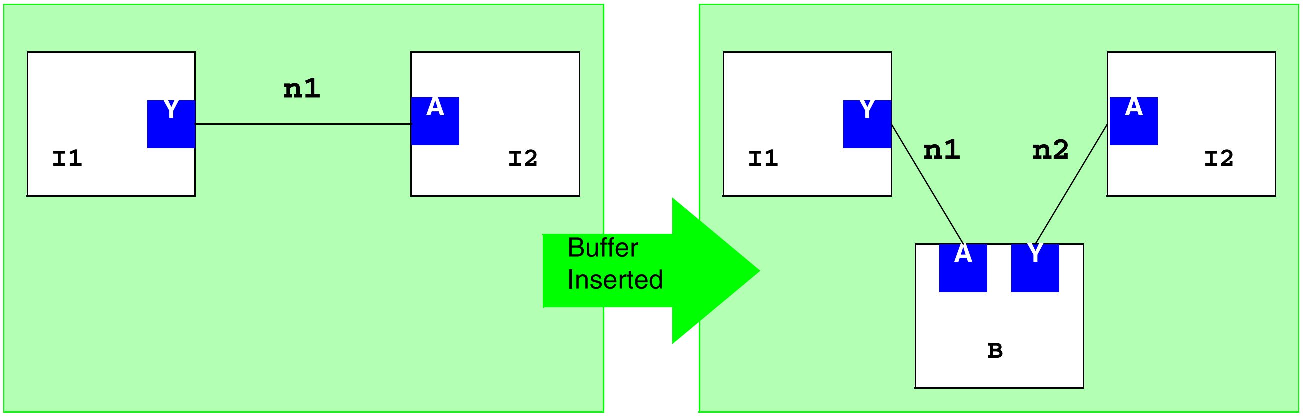
- Adding a Buffer
- Copying Routes
- Deleting a Buffer
- Remastering an Instance
- Routing the ECO Changes
- Saving the ECO Changes
- Saving the ECO Changes

Adding a Buffer

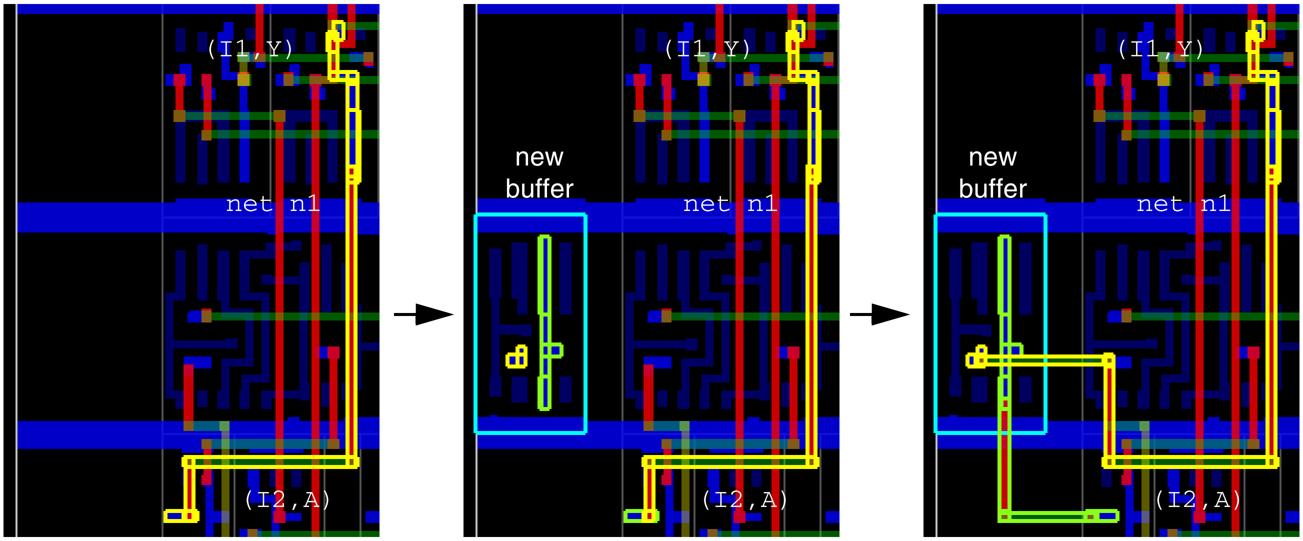
In this example, ECO commands are used to buffer a net. The net `n1` is a two-pin net from pin (`I1`, `Y`) to pin (`I2`, `A`). In this case, the pins are named as instance-pin pairs. The added buffer `B` will be an instance of master `MB`. The output pin of `MB` is `Y` and the input pin is `A`. The `POWR` and `GRND` pins of the new instance are connected to `VDD` and `VSS`, respectively.

```
set eco [eco_begin]
eco_create_instance -eco_control $eco -cell MB -view abstract -lib libname -name B
-origin {x y}
eco_create_net -eco_control $eco -name n2
eco_disconnect_inst_term -eco_control $eco -inst_name I2 -term_name A
eco_connect_inst_term -eco_control $eco -inst_name B -term_name A -net_name n1
eco_connect_inst_term -eco_control $eco -inst_name B -term_name Y -net_name n2
eco_connect_inst_term -eco_control $eco -inst_name I2 -term_name A -net_name n2
eco_connect_inst_term -eco_control $eco -inst_name B -term_name POWR -net_name VDD
eco_connect_inst_term -eco_control $eco -inst_name B -term_name GRND -net_name VSS
```

The netlist change is graphically shown in the following figure.



The following figure shows an example of a buffer that is added to a net using these commands.



Original net n1 highlighted in yellow

Buffer added
New net n2 terminals highlighted in green

Nets routed

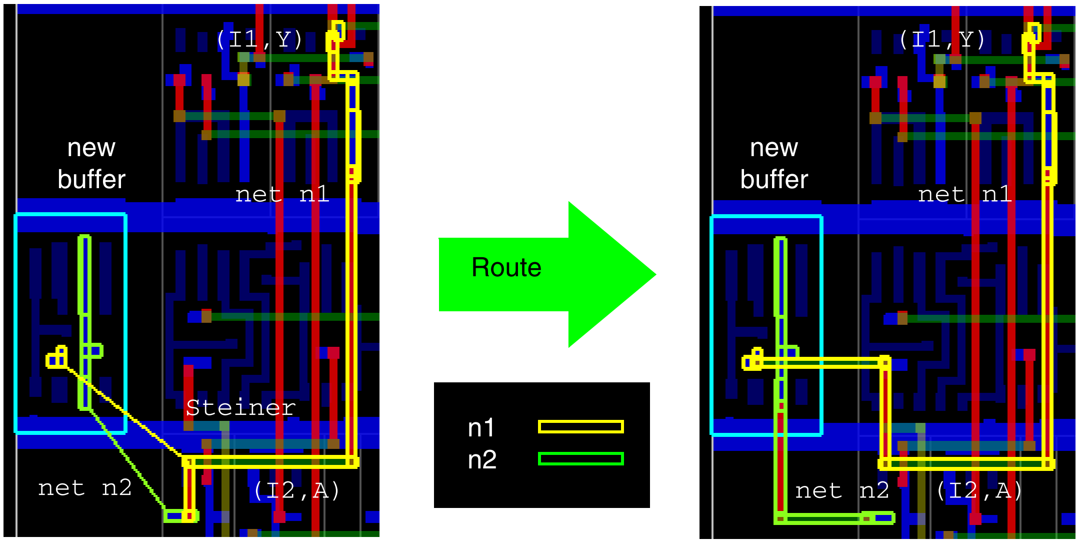
Copying Routes

In the previous section, Adding a Buffer, little was done to the routes present in net `n1`. The net was disconnected from pin (`I2`, `A`), but all route segments and vias would still be present. Ideally, if the buffer is inserted in an already-routed net, it would be useful if the original net n1 would be split near the buffer. This operation can be simulated with the command sequence:

```
eco_copy_implementation -eco_control $eco -from_net n1 -to_net n2
eco_repair_net -eco_control $eco -name n1
eco_repair_net -eco_control $eco -name n2
```

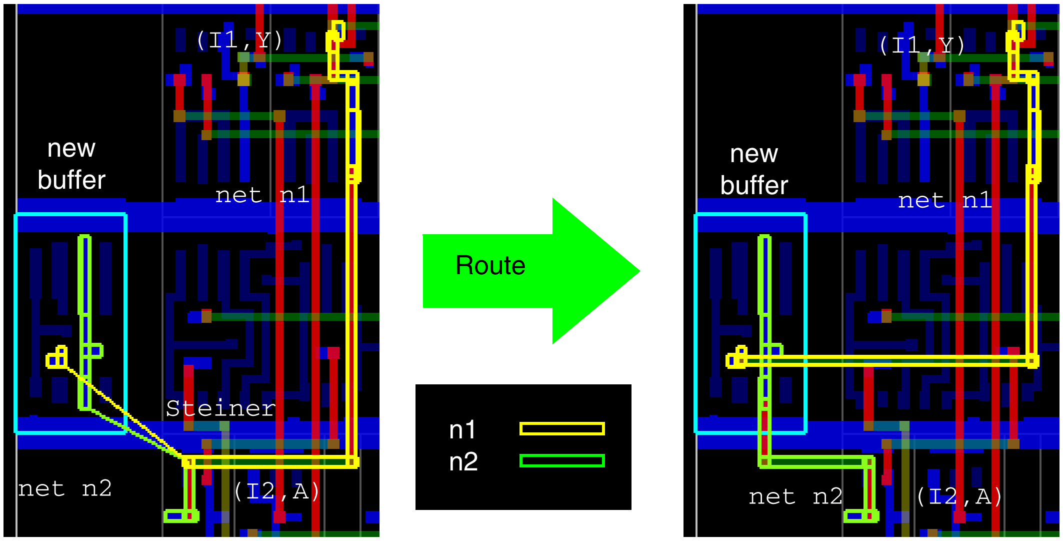
When these commands are issued in the example, the routes in net `n1` are copied to net `n2`, the connectivity is updated, and the nets `n1` and `n2` are trimmed appropriately by the `eco_repair_net` command.

The following figure shows how nets are routed when the copy/repair commands are not used.



Guides are created (1) between the terminals for the new net `n2`, and (2) from the closest point of the existing net `n1` to its new terminal on the new buffer. This creates a steiner in `n1`, dividing `n1` into two routes. The route to the disconnected `n1` terminal is discarded and the guides are routed without changes to the remaining route.

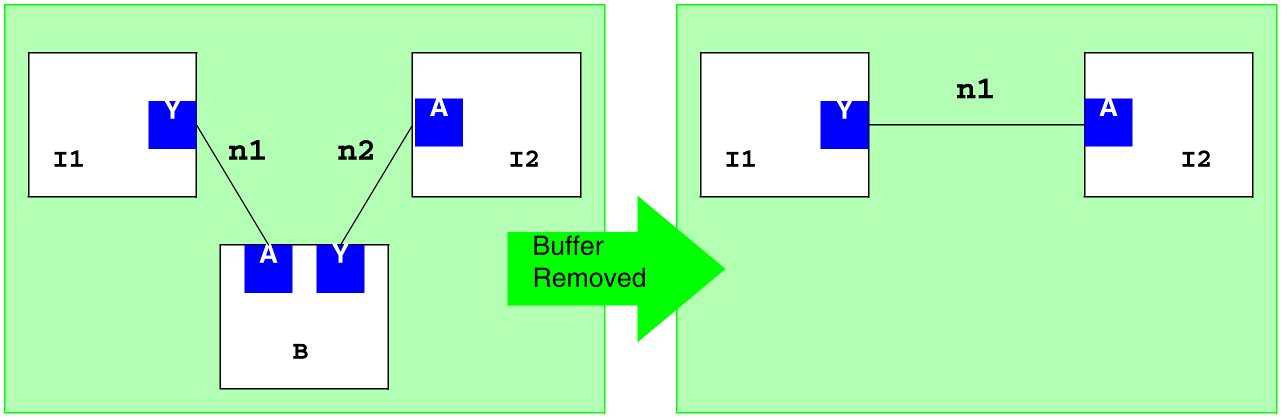
In contrast, the following figure shows the same two nets when copy/repair commands are used.



In this case, the route from net `n1` is copied to the new net `n2`. The `eco_repair_net` command updates the connectivity, creating guides from the closest points on the nets to the new terminals. The routes to the disconnected terminals for both nets are discarded. The guides are routed, and the nets are trimmed where needed. This results in a more direct connection between terminals for `n1`, compared with the previous example which did not use copy/repair.

Deleting a Buffer

In the previous example, a buffer was added to net `n1` and a new net `n2` was created. In this example, the buffer and `n2` will be removed, and net `n1` will be restored to its original state.



```
set eco [eco_begin]
eco_copy_implementation -eco_control $eco -from_net n2 -to_net n1
eco_disconnect_inst_term -eco_control $eco -inst_name B -term_name A
eco_disconnect_inst_term -eco_control $eco -inst_name B -term_name Y
eco_disconnect_inst_term -eco_control $eco -inst_name I2 -term_name A
eco_disconnect_inst_term -eco_control $eco -inst_name B -term_name POWR
eco_disconnect_inst_term -eco_control $eco -inst_name B -term_name GRND
eco_connect_inst_term -eco_control $eco -inst_name I2 -term_name A -net_name n1

eco_destroy_net -eco_control $eco -name n2
eco_destroy_instance -eco_control $eco -name B

eco_repair_net -eco_control $eco -name n1
```

In the first step, the implementation of net `n2` is copied to net `n1`. It is expected that the merged net will combine the routes from both `n2` and `n1`. The net connectivity is established, and net `n2` and the buffer are destroyed. The new net `n1` is repaired and then routed.

Important

> `eco_destroy_net` will remove all routes in the net but cannot remove terminals. In order to remove the net, you must disconnect terminals from the net before using `eco_destroy_net`.

Remastering an Instance

In this example, instance `I1` is remastered, changing the master from `A` to `B`.

```
set eco [eco_begin]
eco_remaster_instance -eco_control $eco -lib mlib -cell B -view abstract -name I1
eco_move_instance -eco_control $eco -name I -origin {$x $y}
```

Since remastered instances often do not match the footprint of the original instance, remastered instances might need to be moved to a legal location. This can be done with the `eco_move_instance` command.

Note: Space-based Router and Chip Optimizer does not check the location of the instance placement.

Routing the ECO Changes

When you are ready to close the opens that remain after ECO changes, use one of the following methods:

- Using the ECO Router
- Using the Point-to-Point Router
- Using the Detail Router
- Using the Global Router

Using the ECO Router

The ECO router will identify nets with opens and complete routing on those nets according to the design rules. This is the *preferred* method for finishing ECO routing. Various techniques are used, depending on the length of the guides that represent the opens. For information on the command arguments, refer to "eco_route" on page 690.

Using the Point-to-Point Router

The point-to-point router can quickly close opens without disturbing surrounding nets. When used with an ECO command sequence, the routing can be undone.

To use the point-to-point router to close opens in the design, use one of the following methods,

➤ Select the guides interactively then run the point-to-point router using:

```
p2p_route -set [get_selection_set] -show_failures
```

➤ Select all guides and use the point-to-point router using this script:

```
# make only guides active, make all other layers and objects inactive
set_active -lpp [get_layers] -active false
set_active -lpp { "user_guides" "guides" "instance:boundary" "instance:label"\
"term:label" annotations:violations" "annotations:dimensions" \
"annotations:others" "highlights:HL1" "grids:routing" "grids:manufacturing" \
"grids:placement" "grids:snap" "grids:axis"} -active false
set_active -lpp guides -active true
view_layer -lpp guides -visible true

# set routing granularity to shapes or vias
set_active -object {"net" "route" "connected_shapes"} -active false

# select all guides (only things active)
select_all

set num_guides [set_count -set [get_selection_set]]
if {$num_guides !=0} {
   p2p_route -set [get_selection_set] -show_failures
}
```

Caution

The point-to-point router is not ideal for longer guides. Care should be taken to exclude longer and non-ECO guides.

Using the Detail Router

The detail router is useful for closing opens for a larger number of nets and the work can be undone when used with an ECO command sequence. Unlike the point-to-point router, there is a increased chance that the detail router will cause disruptions to surrounding nets because the detail router can rip up areas to allow difficult routes to converge. The entire top cell will be operated on If you do not select and specify the guides.

To use the detail router to close all opens in the design, use the following script:

```
# make only guides active, make all other layers and objects inactive
set_active -lpp [get_layers] -active false
set_active -lpp { "user_guides" "guides" "instance:boundary" "instance:label"\
"term:label" annotations:violations" "annotations:dimensions" \
"annotations:others" "highlights:HL1" "grids:routing" "grids:manufacturing" \
"grids:placement" "grids:snap" "grids:axis"} -active false
set_active -lpp guides -active true
view_layer -lpp guides -visible true

# set routing granularity to shapes or vias
set_active -object {"net" "route" "connected_shapes"} -active false

# select all guides (only guides are active)
select_all
detail_route -set [get_selection_set]
```

Caution

To route longer guides, use the complete router flow for quicker processing.

For example,

```
global_route -mode eco -set [get_selection_set]
croute
detail_route -mode eco -set [get_selection_set]
```

Using the Global Router

The global router can be used to automatically connect the ECO changes made using an ECO command sequence. If the global router has not been used in the session, it must be initialized before starting an ECO command sequence using:

```
global_route -mode init
```

When `eco_begin` is run, the global router will activate its design observers which will watch all netlist changes and add the affected nets to a list of nets to route. The routing changes are made using:

```
global_route -mode eco
```

The detail router must also be run to change the global routes to `wire:detail`.

Using the Search and Repair Method

In addition to using the detail router to close the opens, the search and repair method runs the checker to search for same net and different net spacing violations, and attempts to repair the violations that are found. The entire top cell will be operated on if you do not select and specify the guides.

To use the search and repair method to close all opens in the design, use the following commands:

```
# make only guides active, make all other layers and objects inactive
set_active -lpp [get_layers] -active false
set_active -lpp { "user_guides" "guides" "instance:boundary" "instance:label"\
"term:label" annotations:violations" "annotations:dimensions" \
"annotations:others" "highlights:HL1" "grids:routing" "grids:manufacturing" \
"grids:placement" "grids:snap" "grids:axis"} -active false
set_active -lpp guides -active true
view_layer -lpp guides -visible true

# set routing granularity to shapes or vias
set_active -object {"net" "route" "connected_shapes"} -active false

# select all guides (only guides are active)
```

```
select_all
search_and_repair -set [get_selection_set]
```

Saving the ECO Changes

After you have completed the changes and closed the opens, if you are satisfied with the result, you must commit the changes to have them saved using the `eco_commit` command.

The following example shows the global router commands added to the buffer insertion sequence and the `eco_commit` command added at the end to save the changes.

```
# The global router initialization is only needed if the global router
# has not been run in this session. For example, if you loaded a pre-routed design.
```

global_route -mode init

```
set eco [eco_begin]
eco_create_instance -eco_control $eco -cell MB -view abstract -lib libname -name B
-origin {$x $y}

eco_create_net -eco_control $eco -name n2

eco_disconnect_inst_term -eco_control $eco -inst_name I2 -term_name A

eco_connect_inst_term -eco_control $eco -inst_name B -term_name A -net_name n1
eco_connect_inst_term -eco_control $eco -inst_name B -term_name Y -net_name n2
eco_connect_inst_term -eco_control $eco -inst_name I2 -term_name A -net_name n2
eco_connect_inst_term -eco_control $eco -inst_name B -term_name POWR -net_name VDD
eco_connect_inst_term -eco_control $eco -inst_name B -term_name GRND -net_name VSS
```

global_route -mode eco

```
eco_commit -eco_control $eco
```

Using the remastering example, the following sequence remasters an instance, routes the changes with the point-to-point router, then saves the change after the results are checked.

```
set eco [eco_begin]
eco_remaster_instance -eco_control $eco -lib mlib -cell B -view abstract -name I1
eco_move_instance -eco_control $eco -name I -origin {$x $y}
eco_repair_net -eco_control $eco -name n1
eco_repair_net -eco_control $eco -name n2

# Interactively select the guides for the disconnects before proceeding

p2p_route -set [get_selection_set] -show_failures

# Analyze the changes; check timing

eco_commit -eco_control $eco
```

Undoing ECO Changes

After the global router is done, you can analyze or do further work on the design before deciding whether you want to accept the changes. For example, if analysis shows that timing has degraded, you can issue `eco_undo` to return the design to the state it was at when `eco_begin` was invoked, which started the command sequence. This allows you to check *what-if* scenarios.

Using the previous remastering example, this sequence remasters an instance, routes the changes with the point-to-point router, then reverses the change after the results are checked.

```
set eco [eco_begin]
eco_remaster_instance -eco_control $eco -lib mlib -cell B -view abstract -name I1
eco_move_instance -eco_control $eco -name I -origin {$x $y}
eco_repair_net -eco_control $eco -name n1
eco_repair_net -eco_control $eco -name n2

# Interactively select the guides for the disconnects before proceeding

p2p_route -set [get_selection_set] -show_failures

# Analyze the changes; check timing

eco_undo -eco_control $eco
```

11

Route Commands

This chapter describes the AutoRouter commands. These commands are used for routing signal nets after power routing (described in "Power Route Commands" on page 463) and other specialty routing, such as clock nets and buses (described in "Specialty Route Commands" on page 549).

The AutoRouter flow includes the following steps:

- Global Route replaces all opens with global routes and re-routes to reduce congestion.
- Local Route adds pin escapes.
- Conduit Route lays down as many wires as possible along routing conduits.
- Detail Route completes the routing of all nets and resolves violations.
- Post-Route Refinement fixes some violations, re-routes short connections, and removes unnecessary vias.

For more information on the routing flow and the Virtuoso Space-based Router Graphical User Interface, refer to "Routing Your Design" in *Virtuoso Space-based Router User Guide*.

The router commands are presented in alphabetic order:

- extend_wire_to_pin_edge on page 742
- fix_errors on page 743
- global_route on page 747
- local_route on page 754
- optimize_routes on page 755
- p2p_route on page 758
- route_optimize on page 759
- show_congestion on page 782
- wrongway_pin_escape on page 785

Additional information is given to help you get started with the router.

- Preparing the Routing Environment on page 788
- A Routing Example on page 811
- Evaluating Router Results on page 820

congestion_analysis

```
congestion_analysis
     [ -gcell_width i_tracks ]
     [ -wrong_way ]
```

Performs a congestion analysis on the active design and outputs a congestion summary to the Transcript area. If necessary, the congestion analysis builds the *gcell* grid, comprised of uniform square areas.

The congestion summary indicates the number of gcells of each type for each layer and statistics for each group including:

- The number and percentage of overcongested gcells
- The percentage of gcells grouped by the percentage of resources they are using
- The percentage of gcells that have no capacity due to blockages (`Blk%`)

In general, it is not necessary to make an explicit `congestion_analysis` call during global routing. The `global_route` command automatically calls `congestion_analysis` with the default parameters. However, if you want to use a different gcell size, then you must invoke `congestion_analysis` with the desired gcell width, prior to `global_route`.

For a pictorial view of the congestion map, use show_congestion.

Arguments

`-gcell_width` *`i_tracks`*

Specifies the width of the cell gcell, in tracks. If this argument is not given, Space-based Router and Chip Optimizer automatically determines the size of the gcells. If the specified `gcell_width` is too small for the size of the design and would create more gcells than can be represented, then the smallest possible gcell width will automatically be used instead.

`-wrong_way`

Allows global wrong-way routing in the entire design. By default, wrong-way global routing is not allowed.

This option is useful if portions of a design are left unrouted because some pins require wrong-way access, and when routing channels are overcongested and could benefit if some portion switched its preferred routing direction. However, more searching is allowed if wrong-way routing is permitted, which increases the processing time.

Example

The following example sets the gcell width to 10 tracks, then runs the global router.

```
congestion_analysis -gcell_width 10
global_route
```

The following is an example congestion summary.

```
     Gcell summary
Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
----------------------------------------------------------------------------------------
metal1:cell     65         5   ( 7.69%)  9   6   1   9   3   3   3  23  13  20    0    0
metal1:edge     87         7   ( 8.05%) 34   8   1   8   0   0   0  16  12  11    0    0
metal2:cell     65         1   ( 1.54%)  9  10  12  15   4   6   1  15  15   7    0    0
metal2:edge     87         4   ( 4.60%) 32  10  14  10   0   1   0  14  11   0    0    0
metal2:down-via 65         0   ( 0.00%)
----------------------------------------------------------------------------------------
```

- `cell` gcells occupy an area of the design.
- `edge` gcells represent the region between two cell gcells on the same layer.
- `down-via` gcells represent the region between two cell gcells on different layers.
- `#Gcell` is the number of gcells of the given type for the given layer.

- `#Overcon` is the number of layer/type gcells that are overcongested.
- `%Overcon` is the percentage of layer/type gcells that are overcongested.
- Each percentage grouping represents the number of layer/type gcells with congestion that meets the given heading percentage. In the example above, nine (9) `metal1:cell` gcells have less than 10% congestion, while twenty (20) are at least 90% congested.

Related Information upper-right

Tcl Command	show_congestion

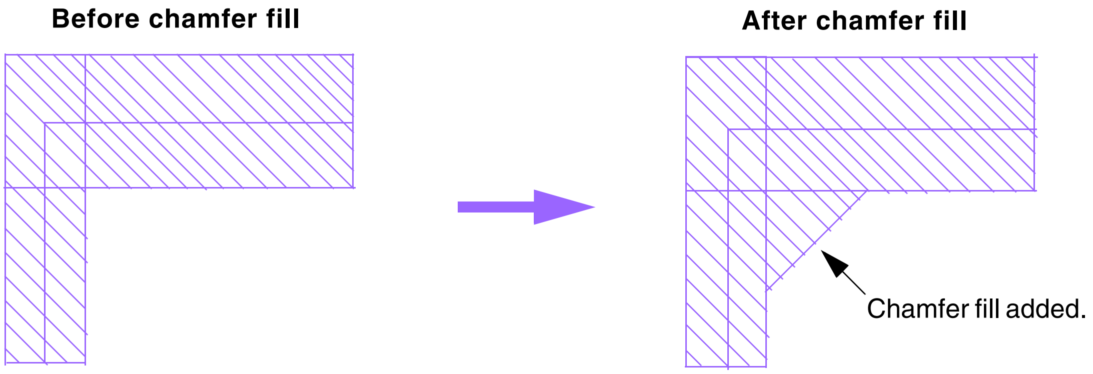
create_chamfer_fill

```
create_chamfer_fill
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -layer {s_layerName…} ]
     [ -chamfer_filter {s_filter…} ]
     [ -length_threshold f_threshold ]
     [ -chamfer_value1 {f_chamfer1 | { f_inner1 f_outer1}} ]
     [ -chamfer_value2 {f_chamfer2 | { f_inner2 f_outer2}} ]
     [ -extend_via_chamfer [ true | false ] ]
     [ -allow_violation [ true | false ] ]
     [ -output_purpose s_purposeName ]
     [ -material_removal_purpose s_purposeName ]
```

Creates chamfer fill on wires (pathSegs). The chamfer fill mechanism removes 90-degree corners by filling the corners of wires that form T-junctions, L-shaped wires, wire-to-via, wire-to-rectangle, and wire-to-polygon connections. T-junctions and L shapes created using only rectangles or polygons are not considered for chamfer fill.

An example of a chamfer fill shape on an L-shaped wire is shown in Figure 11-1.

Figure 11-1 Chamfer Fill on an L-Shaped Wire



An example of chamfer fill for a wire that connects to a pin (rectangle) is shown in Figure 11-2.

Figure 11-2 Chamfer Fill on a T-Junction Where a Wire Connects to a Pin

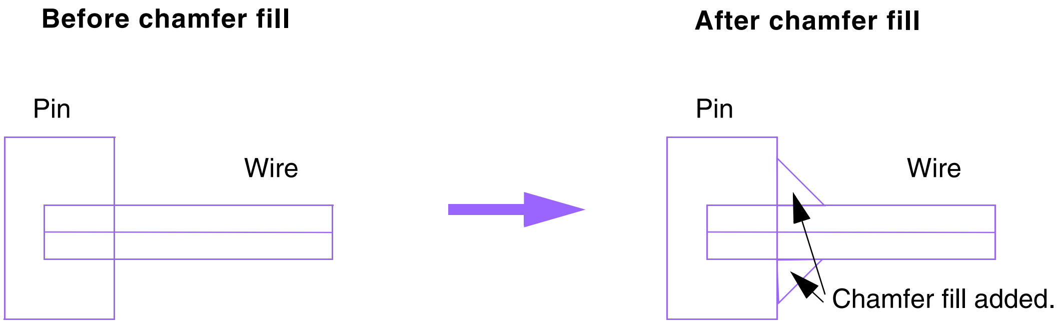
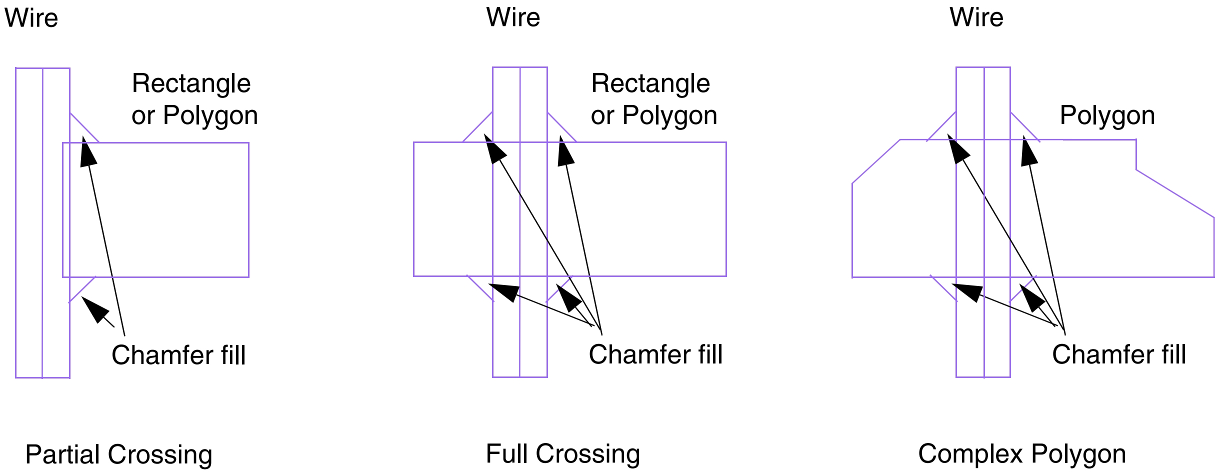


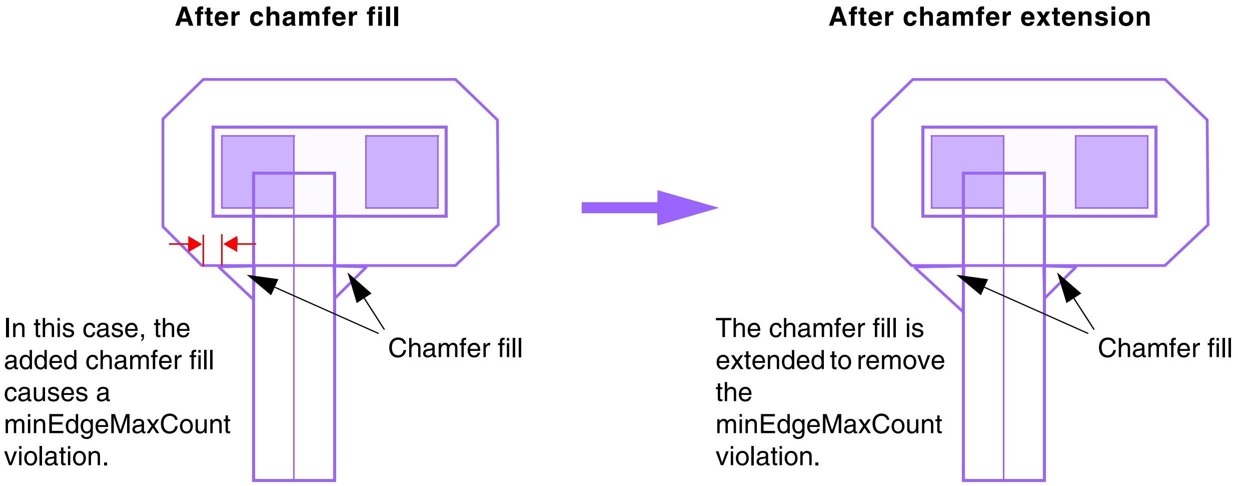
Figure 11-3 shows examples of chamfer fill shapes added at the intersections of wires and rectangles or polygons.

Figure 11-3 Chamfer Fill Where a Rectangle or Polygon Crosses a Wire



If the chamfer fill added by this command causes a `minEdgeMaxCount` violation, the chamfer fill edges are extended to fix the violation, as shown in Figure 11-4.

Figure 11-4 Chamfer Fill Extended to Prevent a minEdgeMaxCount Violation



In this case, the added chamfer fill causes a minEdgeMaxCount violation.

The chamfer fill is extended to remove the minEdgeMaxCount violation.

Arguments

`-all`

Creates chamfer fill on all the wires of the design. This is the default.

`-allow_violation [ true | false ]`

When `true`, adds chamfer fill even if it causes design rule violations. By default, chamfer fill that causes violations will not be added.

`-chamfer_filter {s_filter…}`

Filters the wires on which chamfer fill is created based on specified criteria. For example:

- To create chamfer fill only on wires with maximum voltage greater than 100V, specify
  `{"voltage > 100.0"}`
- To create chamfer fill only on wires with width greater than or equal to 10 microns, specify
  `{"width >= 10.0"}`
- To create chamfer fill only on wires with width greater than or equal to 8 microns, and voltage less than 45V, specify
  `{"width > 8.0" "voltage < 45.0"}`

`-chamfer_value1 {f_chamfer1 | {f_inner1 f_outer1}}`

Specifies the chamfer value to be applied if `length1` or `length2` is less than the `length_threshold` value. See Example 2—Chamfer fill situation where length_threshold is considered for an example.

- If a single value is specified, it applies to both the inner and outer chamfers.
- If two values are specified, the first value is the inner chamfer value and the second value is the outer chamfer value.

`-chamfer_value2 {f_chamfer2 | {f_inner2 f_outer2}}`

Specifies the chamfer value to be applied if `length1` and `length2` are greater than or equal to the `length_threshold` value. See Example 2—Chamfer fill situation where length_threshold is considered for an example.

- If a single value is specified, it applies to both the inner and outer chamfers.
- If two values are specified in a list, the first value is the inner chamfer value and the second value is the outer chamfer value.

`-extend_via_chamfer [ true | false ]`

When `false`, chamfer fill will not be added to wire-to-via junctions.

(Default) When `true`, adds chamfer fill on wire-to-via junctions.

`-layer` **{***`s_layerName…`***}** Creates chamfer fill only on the layers specified in the list. By default, all the routing layers are included.

`-length_threshold` *`f_threshold`*

Specifies the threshold value that determines whether `chamfer_value1` or `chamfer_value2` is applied.

If `length1` or `length2` is less than the `length_threshold` value, `chamfer_value1` is applied; else, `chamfer_value2` is applied. See Example 2—Chamfer fill situation where length_threshold is considered for an example of how `length1` and `length2` are measured.

`-material_removal_purpose` *`s_purposeName`*

Adds a chamfer shape on the specified purpose to be processed during stream out or fracturing to remove the 90-degree corner of an L-shaped wire.

`-output_purpose` *`s_purposeName`*

Specifies the purpose for the added chamfer fill shapes. The default is `gapFill`.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

	Creates chamfer fill only on the wires in the specified region. By default, the entire design is processed.
`-set` *`d_setObj`*	Operates on route segments in the set.

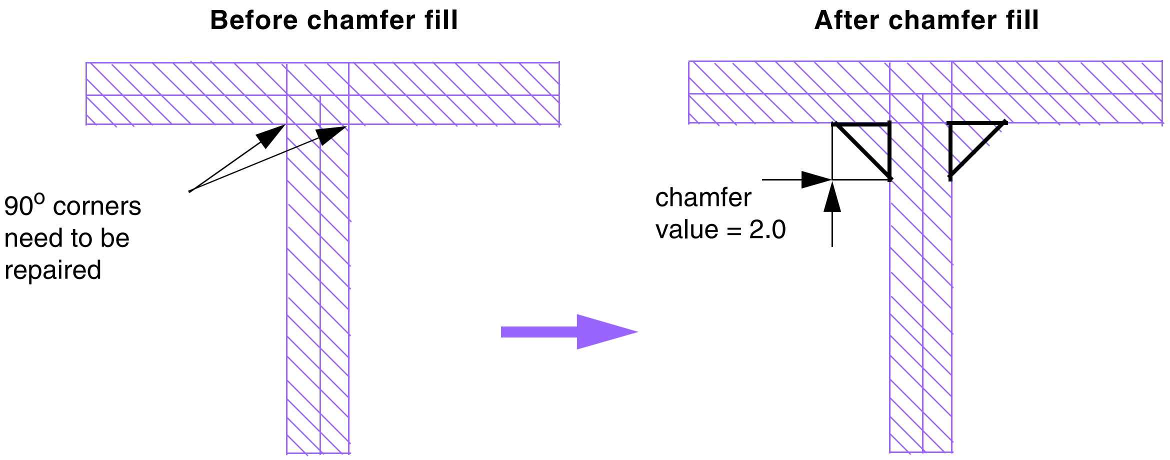
Examples

Example 1—Chamfer fill situation where length_threshold is not considered

```
create_chamfer_fill -all -length_threshold 0.0 -chamfer_value1 2.0 -chamfer_value2
2.0
```

Creates chamfer fill of 2 microns on all routing layers for all 90-degree corners at T-junctions, L-shaped wires, and wire-to-via connections. When applied to the example in Figure 11-5, chamfer fill of 2 microns is created on both sides of the T-junction.

Figure 11-5 Chamfer Fill Situation Where length_threshold Is Not Considered



Example 2—Chamfer fill situation where length_threshold is considered

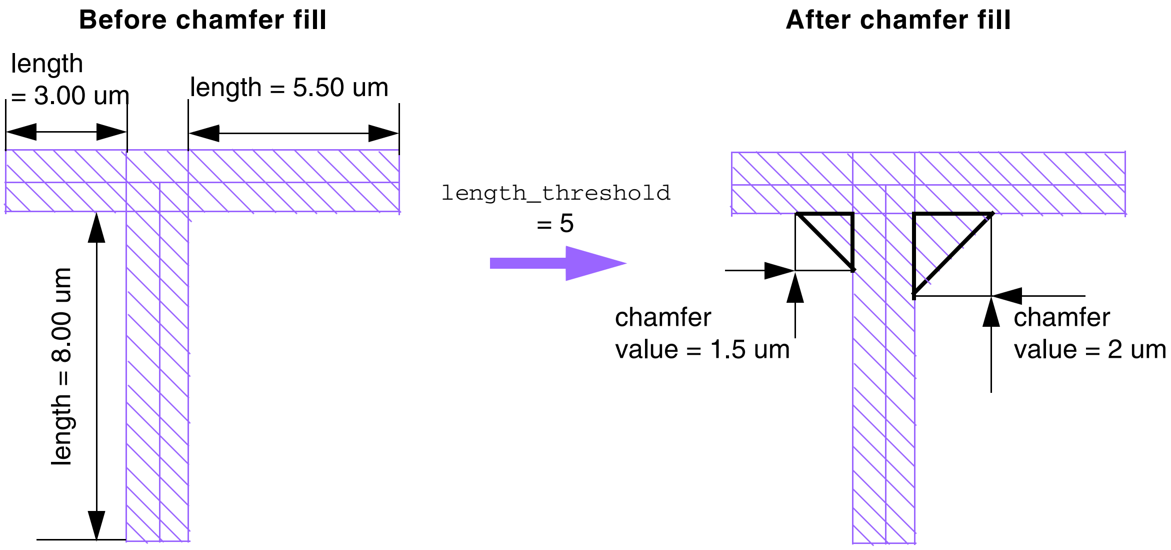
```
create_chamfer_fill -all -layer M2 -length_threshold 5.0 -chamfer_value1 1.5
-chamfer_value2 2.0
```

Creates chamfer fill on the `M2` layer according to the following conditions:

- If one of the two segments is less than 5.0 microns in length, then the chamfer fill will be 1.5 microns.
- If both segments are greater than or equal to 5.0 microns in length, then the chamfer fill will be 2.0 microns.

Figure 11-6 illustrates this with a T-junction. The left corner has segments of 3.0 and 8.0 microns in length, so its chamfer fill is 1.5 microns. Both segments (5.50 and 8.0 microns) for the right corner are greater than or equal to the `length_threshold` of 5.0 microns, so its chamfer is 2 microns.

Figure 11-6 Chamfer Fill Situation Where length_threshold Is Considered

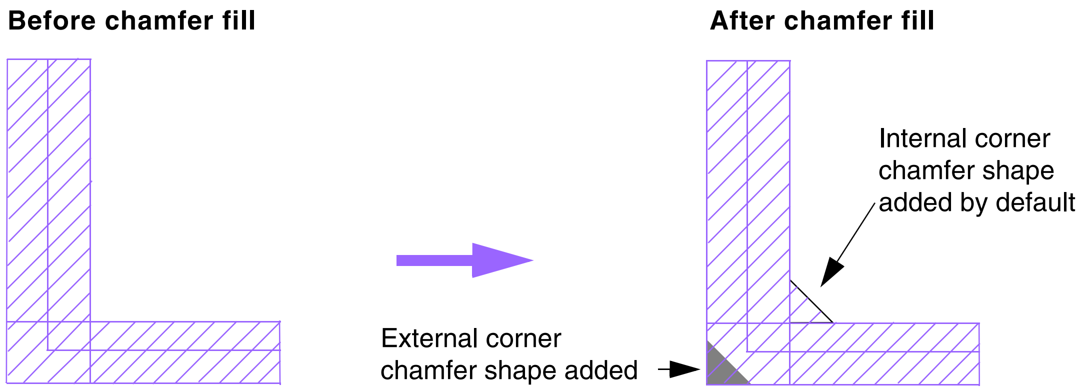


Example 3—Using material_removal_purpose to chamfer the external corner of an L-shaped wire

```
create_chamfer_fill -all -layer M2 -material_removal_purpose "noDrawing"
```

Removes the 90-degree corners by adding chamfer fill on the `M2` layer at T-junctions, L-shaped wires, and wire-to-via connections. In addition, chamfer fill on the `noDrawing` purpose is added to overlap the 90-degree corners of L-shaped wires as shown in Figure 11-7 on page 721.

Figure 11-7 Using material_removal_purpose to Chamfer the External Corner of an L-Shaped Wire

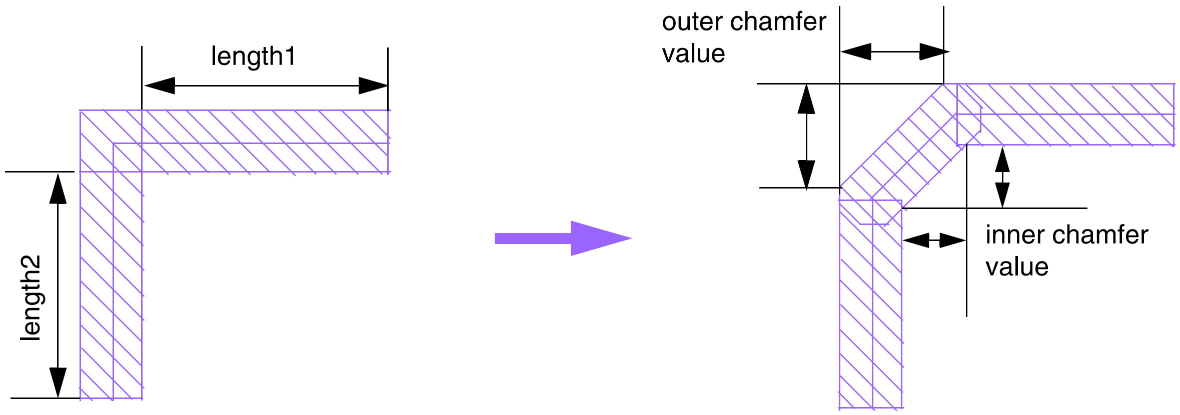

Before chamfer fill
After chamfer fill
Internal corner chamfer shape added by default
External corner chamfer shape added

create_wire_chamfer

```
create_wire_chamfer
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -layer {s_layerName…} ]
     [ -chamfer_filter {s_filter…} ]
     [ -length_threshold f_threshold ]
     [ -chamfer_value1 {f_chamfer1 | { f_inout1 f_endOfStripe1}} ]
     [ -chamfer_value2 {f_chamfer2 | { f_inout2 f_endOfStripe2}} ]
     [ -allow_violation [ true | false ] ]
     [ -inner_chamfer [ true | false ] ]
     [ -chamfer_end_of_stripe [ true | false ] ]
```

Creates chamfer on wires (pathSegs). The wire chamfer mechanism replaces 90-degree wires with 45-degree wires. An example of a wire chamfer is shown in Figure 11-8.

Figure 11-8 Wire Chamfer on a 90-degree Wire Turn



Arguments

`-all`

Creates wire chamfers on all the wires of the design. This is the default.

`-allow_violation [ true | false ]`

When `true`, creates wire chamfers even when they cause design rule violations. By default, chamfers that cause violations will not be created.

`-chamfer_end_of_stripe [ true | false ]`

When `true`, creates wire chamfers on the ends of wires to align with chamfered via arrays. By default and when `false`, ends of wires are not chamfered.

`-chamfer_filter` **{***`s_filter…`***}**

Filters the wires on which chamfers are created based on specified criteria. For example:

- To create chamfers only on wires with maximum voltage greater than 100V, specify
  `{"voltage > 100.0"}`
- To create chamfers only on wires with width greater than or equal to 10 microns, specify
  `{"width >= 10.0"}`
- To create chamfers only on wires with width greater than or equal to 8 microns, and voltage less than 45V, specify
  `{"width > 8.0" "voltage < 45.0"}`

`-chamfer_value1 {`*`f_chamfer1`* `|` **{***`f_inout1 f_endOfStripe1`***}**`}`

Specifies the chamfer value to be applied if `length1` or `length2` is less than the `length_threshold` value. See Figure 11-10 for an example.

- If a single value is specified, it is the inner, outer, and end-of-stripe chamfer.
- If two values are specified, the first value is the inner and outer chamfer, and the second value is the end-of-stripe value.

`-chamfer_value2 {`*`f_chamfer2`* `|` **{***`f_inout2 f_endOfStripe2`***}**`}`

Specifies the chamfer value to be applied if `length1` and `length2` are greater than or equal to the `length_threshold` value. See Figure 11-9 for an example.

- If a single value is specified, it is the inner, outer, and end-of-stripe chamfer.
- If two values are specified, the first value is the inner and outer chamfer, and the second value is the end-of-stripe value.

`-inner_chamfer [ true | false ]`

If set to `false`, `chamfer_value1` or `chamfer_value2` applies to the outer corner of the wire. The default value of `true` applies `chamfer_value1` or `chamfer_value2` to the inner corner of the wire.

`-layer` **{***s_layerName…***}** Creates wire chamfer only on the layers specified in the list. By default, all the routing layers are included.

`-length_threshold` *`f_threshold`*

Specifies the threshold value that is used to determine if `chamfer_value1` or `chamfer_value2` is applied.

If `length1` or `length2` is less than the `length_threshold` value, `chamfer_value1` is applied; otherwise `chamfer_value2` is applied. Figure 11-8 shows how `length1` and `length2` are measured.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

Creates chamfer only on the wires in the specified region. By default, the entire design is processed.

`-set` *`d_setObj`* Operates on route segments in the set.

Examples

Example 1—Inner wire chamfer

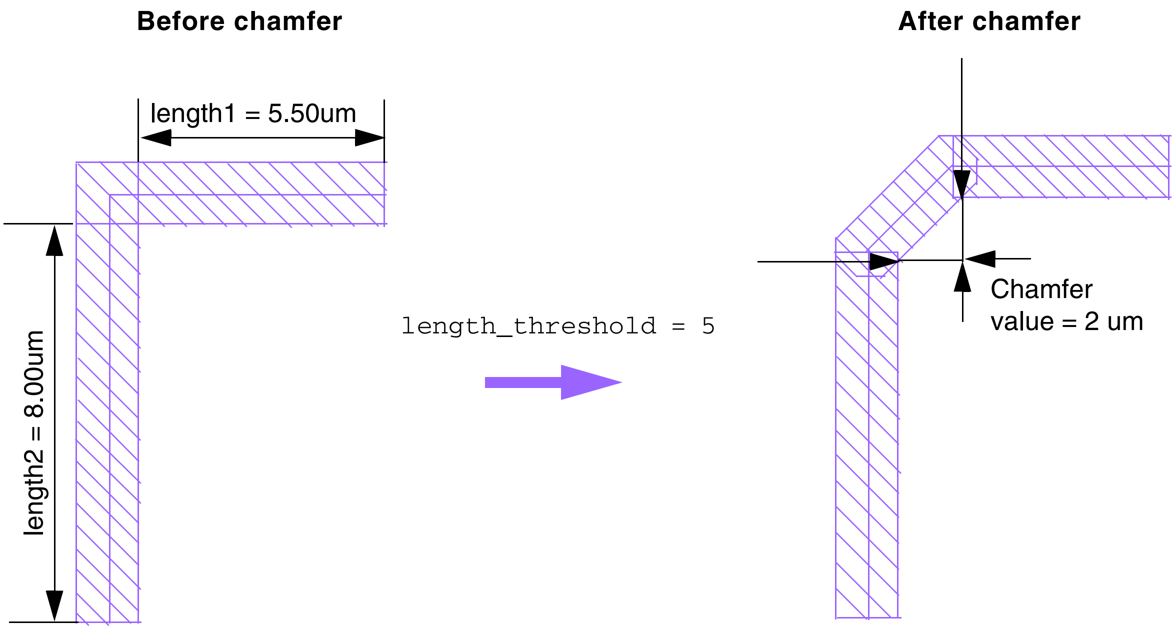
```
create_wire_chamfer -all -layer M2 -length_threshold 5.0 -chamfer_value1 1.5
-chamfer_value2 2.0 -inner_chamfer true
```

Creates wire chamfer on the M2 layer according to the following conditions:

- If one of the two segments is less than 5 microns in length, then the wire chamfer will be 1.5 microns, measured from the inner side of the wire.
- If both segments are greater than or equal to 5 microns in length, then the wire chamfer will be 2.0 microns, measured from the inner side of the wire.

In Figure 11-9, a wire chamfer of 2 microns is created on the inner side of the wire, because the lengths of both segments (5.5 and 8 microns) are greater than `length_threshold` (5 microns).

Figure 11-9 Inner Wire Chamfer Situation Where length1 and length2 Are Greater Than length_threshold



Example 2—Outer wire chamfer

```
create_chamfer_wire -all -layer M2 -length_threshold 6.0 -chamfer_value1 2.0
-chamfer_value2 3.0 -inner_chamfer false
```

Creates wire chamfer on the M2 layer according to the following conditions:

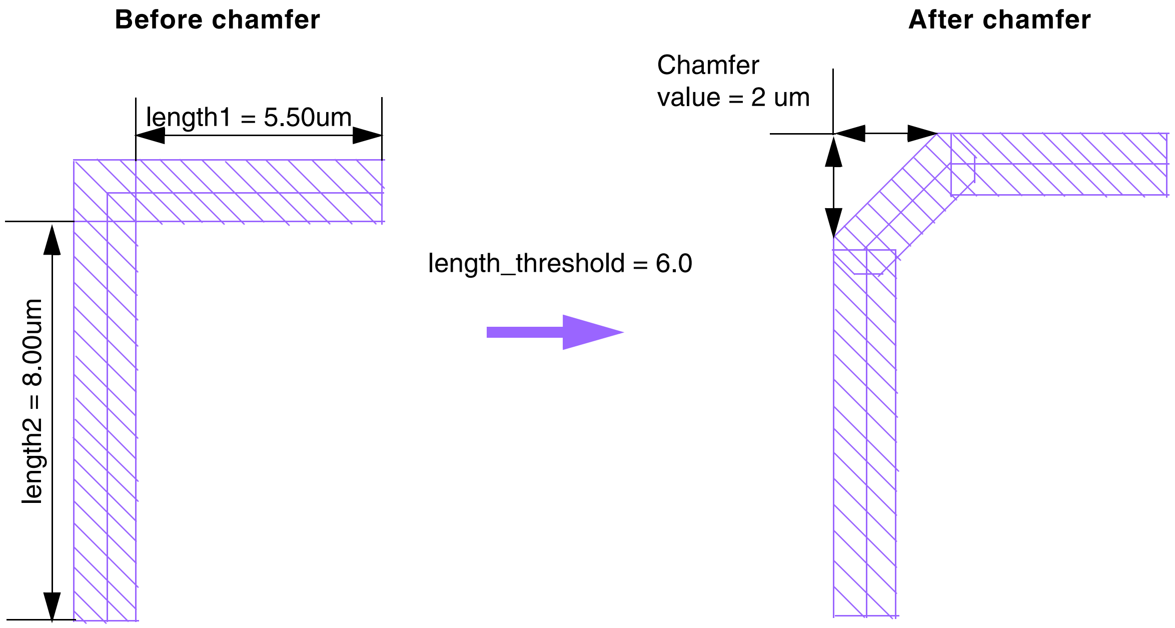
- If one of the two segments is less than 6 microns in length, then the wire chamfer will be 2.0 microns, measured from the outer side of the wire.
- If both segments are greater than or equal to 6 microns in length, then the wire chamfer will be 3.0 microns, measured from the outer side of the wire.

The following command using dual values for `-chamfer_value1` and `-chamfer_value2` has the same result:

```
create_chamfer_wire -all -layer M2 -length_threshold 6.0 -chamfer_value1 {2.0 0}
-chamfer_value2 {3.0 0} -inner_chamfer false
```

In Figure 11-10, a wire chamfer of 2 microns is created on the outer side of the wire, because the length of one segment (5.5 microns) is less than `length_threshold` (6 microns).

Figure 11-10 Outer Wire Chamfer Situation Where length1 Is Less Than length_threshold



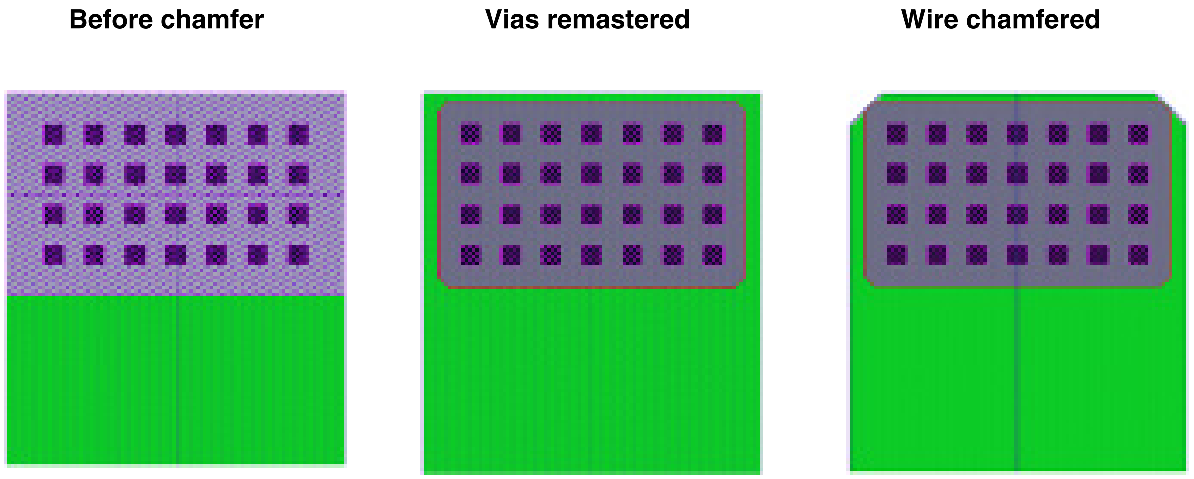
Example 3—Chamfering end of stripe wires

```
create_chamfer_wire -all -layer M2 -length_threshold 6.0 -chamfer_value1 {2.5 2.0}
-chamfer_value2 {3.5 3.0} -chamfer_end_of_stripe true
```

Chamfers the ends of stripes by 2.0 if the length of the wire is less than 6.0, or by 3.0 if the length of the wire is greater than or equal to 6.0.

Figure 11-11 illustrates the effect of specifying the `chamfer_end_of_stripe` argument, which chamfers power routing stripes to align with chamfered via arrays.

Figure 11-11 Chamfering End of Stripe Wires Using `chamfer_end_of_stripe`



Before chamfer

Vias remastered

Wire chamfered

croute

```
croute
     [ -replace_globals [ true | false ] ]
     [ -set d_setObj ]
     [ -skip_layers {s_layerName …} ]
     [ -threads i_threads ]
     [ -use_grid {mfg | route} ]
```

Assigns tracks for the globally routed design and, guided by the global routes, lays down as many wires as possible along routing conduits. Wires can be pushed to fix spacing violations. When this command is done, guides will indicate where connections need to be completed in the detail route stage. The router strives to make these guides short in length and ensure that violations can be corrected in the detail route step.

Arguments

`-replace_globals [ true | false ]`

Replaces the remaining global routes with guides. Defaults to true.

`-set` *`d_setObj`* Operates on nets in the set. By default, all the nets in the entire design are processed.

`-skip_layers` **{***`s_layerName`* **…}**

Excludes the given layers from processing. By default, all layers are included.

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

`-use_grid {mfg | route}`

Specifies the grid to use for placement of the routing conduits from the following choices:

`mfg`	Selects the manufacturing grid.
`route`	Selects the routing grid. this is the default.

delete_conflicts

```
delete_conflicts
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -check_mode {hard | soft} ]
     [ -crossing [ true | false ] ]
     [ -threads i_threads ]
```

Finds spacing violations in a routed design and removes shapes (not nets) to eliminate the violations, replacing the removed shapes with guides to represent the opens. Can also detect and remove same net crossing violations. Can be run in multi-threading mode.

Arguments

`-check_mode {hard | soft}`

Determines whether this command checks for adherence to `hard` or `soft` constraints (rules).

Default: `hard`

`-crossing [ true | false ]`

When set to `true`, checks for same net crossings and removes violating shapes.
Default: `false`

`-exclude_net {`*`s_netName…`*`}`

Excludes the given nets from processing.

`-exclude_set` *`d_setObj`*

Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Excludes the given types of nets from processing.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Processes routes in the area given by the lower-left (*`f_xlo f_ylo`*) and upper-right (*`f_xhi f_yhi`*) coordinates. By default, the entire top cellview is processed.

`-set` *`d_setObj`*

Processes nets in the given set. If this argument is not specified, the entire top cell is routed.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

delete_routing

```
delete_routing
     [ -floating_route [ true | false ] ]
     [ -global_only [ true | false ] ]
     [ -partition_delete_type [contained_nets | intersecting_nets |
     contained_shapes] ]
     [ -set d_setObj ]
     [ -type trim_extend ]
```

Removes all the routes in the current design, including the floating routes that are not on a net. You can optionally keep the floating routes that are not on a net, remove only global routes, remove only routes with a specific `routeType` property, and restrict processing to nets in a set.

Arguments

`-floating_route [ true | false ]`

When set to `false`, keeps floating routes that are not placed on a net.

Default: `true` (removes floating routes that are not on a net)

`-global_only [ true | false ]`

When set to `true`, removes only global routes.

Default: `false`

`-partition_delete_type [ contained_nets | intersecting_nets | contained_shapes ]`

This option is ignored unless there is a concurrent design partition active. In such cases, the nets are removed according to the specified setting.

`contained_nets`	Only nets with all its terminals within or on the border of the partition are deleted.
`intersecting_nets`	Only nets with at least one terminal completely within the partition are deleted.
`contained_shapes`	Any segments or vias completely within the partition are deleted.

`-set` *`d_setObj`*

When set to `true`, processes only the nets in the set.

Default: `false` (processes all the nets in the design)

`-type trim_extend`

Removes only trim metal, bridge metal, and the routes with the `TrimExtend routeType` property (for example, where pins are extended by extend_pins).

Examples

Removes all the routes of all the nets in the current design and the floating routes that are not on a net.

```
delete_routing
```

Removes only the global routes in the design.

```
delete_routing -global_only
```

Removes only the routes with the `TrimExtend` `routeType` property that are on nets in `myset`.

```
delete_routing -set myset -type trim_extend
```

detail_route

```
detail_route
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -critic [ true | false ] ]
     [ -mode {clean | ECO} ]
     [ -pause_pass i_count ]
     [ -start_pass i_count ]
     [ -stop_pass i_count ]
     [ -collect_same_net_errors [ true | false ] ]
     [ -collect_diff_net_errors [ true | false ] ]
     [ -check_antenna [ true | false ] ]
     [ -maximize_cuts [none | useMinRule | useMaxRule | useViaDef] ]
     [ -optimize_pin_escaped_vias [ true | false ] ]
     [ -threads i_threads ]
     [ -use_double_cut_vias [ true | false ] ]
```

Finishes routing according to design rules by running multiple passes in cycles. Resolves violations that were created during conduit routing and some phases of detail routing. In the first cycle, any remaining opens are routed and error types such as different net violations, weak connect violations, and off-grid errors are addressed by rerouting. Subsequent cycles deal with any remaining DRC violations including same net violations, minimum width, minimum area and minimum enclosed area violations.

During processing, the router outputs status to the Transcript area including the following:

- The number of guides before and after the command was run
- The number of passes within each cycle and the elapsed time for each pass
- The number of connections being routed in a pass (rips) which is comprised of the following counts:
    - ❑ Opens or guides to be routed (unroutes)
    - ❑ Existing violations (errors)
    - ❑ New violations introduced during *violation mode* routing (violatees)
    - ❑ Weak connection violations (weak)
    - ❑ Off-grid wires (off-grid)

Arguments

`-check_antenna [ true | false ]`

(Applies only when `-mode clean` is specified) If true, checks new routes from clean mode for antenna violations and, if found, the original route connections are kept. Default is `false`.

`-collect_diff_net_errors [ true | false ]`

If false, different net errors will not be collected during cycles greater than 1. Default is `true`.

`-collect_same_net_errors [ true | false ]`

If false, same net errors are not collected during cycles greater than 1. Default is `true`.

`-critic [ true | false ]`

Straightens wires where possible after routing. Defaults to `false`.

`-exclude_net` **{***`s_netName`…***}**

Excludes the given nets from processing. Nets that are not fixed or locked in this list can be shifted while routing other nets.

`-exclude_set` *`d_setObj`*

Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Excludes the given types of nets from processing. Nets that are not fixed or locked in this list can be shifted while routing other nets.

`-maximize_cuts [none | useMinRule | useMaxRule | useViaDef]`

When connecting to wide wires, will maximize the number of cuts per via after routing, where possible, according to the specified setting:

`none` (Default) No post-routing effort to maximize cuts.

`useMaxRule` Uses the maximum of via parameters from the applicable oaStdViaDef and existing via rules to maximize cuts.

`useMinRule` Uses constraint values for the cut layer and if not found, allows these constraints to be derived from the default oaStdViaDef.

`useViaDef` Uses via parameters from the applicable oaStdViaDef instead of using constraint values for the cut layer. No vias will be created that violate existing via rules.

`-mode {clean|ECO}` Specifies a special purpose routing mode to run for this step.

`clean` By default, this step is not included.

`ECO` Closes opens in nets given by `-set`. This mode is effective for ECO routing.

Caution

To route longer guides, use the complete router flow for quicker processing.

For example:

```
global_route -mode eco -set [get_selection_set]
croute
detail_route -mode eco -set [get_selection_set]
```

`-optimize_pin_escaped_vias [ true | false ]`

When `true` and re-routing single-cut vias, also optimize pin-escaped vias with double-cut vias where possible. To use this option, you must also specify `-use_double_cut_vias true`.

Typically, pin escapes can only use single-cut vias because the vias must be fully enclosed on the pin.

Default: `false`

`-pause_pass` *`i_count`* Pauses detail_route after the specified pass, if the specified pass is necessary.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Processes routes in the area given by the lower-left (*f_xlo f_ylo*) and upper-right (*f_xhi f_yhi*) coordinates. By default, the entire top cellview is processed.

-set *d_setObj*

Processes nets in the given set. If this argument is not specified, the entire top cell is routed.

-start_pass *i_count*

Starts the detail router at the given pass.

-stop_pass *i_count*

Stops the detail router after the given pass is completed, if the given pass is necessary.

-strictly_in_region **{***f_xlo f_ylo f_xhi f_yhi***}**

Processes only routes with both endpoints in the area given by the lower left (*f_xlo f_ylo*) and upper right (*f_xhi f_yhi*) coordinates. By default, the entire top cellview is processed.

-threads *i_threads*

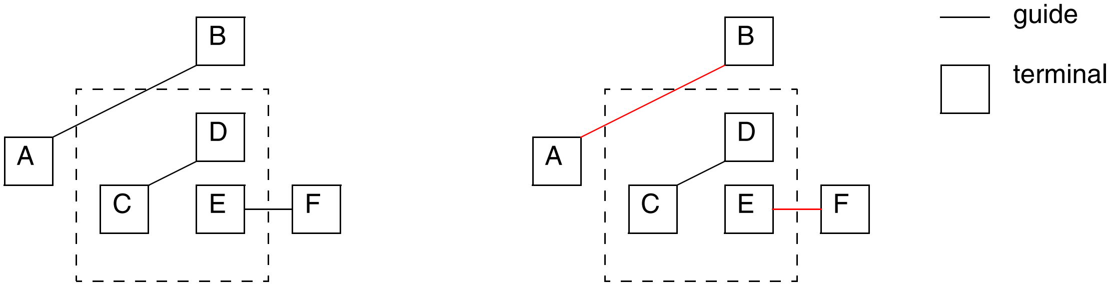
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

-use_double_cut_vias [true | false]

When true, during post-route, re-route with double-cut vias where possible.

Default: false

Figure 11-12 Illustration of the Difference between -region and -strictly_in_region for detail_route



a) When the dotted box represents the `-region` area, all three routes will be detail routed.

b) When the dotted box represents the `-strictly_in_region` area, only route CD will be detail routed because guides for the other two routes (AB, EF) are not fully contained in the given area.

Violation Summary Output

When you include the `-print_summary` argument, violation checks are performed and reported after routing. The following is an example of the violation summary:

```
==============================================================================
The following reported errors are for the passed set of nets only!!
Total number of opens from verify connectivity: 0.
Total number of shorts from verify connectivity: 0.
..............................................................................
The following violations include: minSpacing, minNumCut, maxStack and via ext
   Number of route diff net violations: 0.
   Number of route same net violations: 0.
   Total number of route violations: 0.
..............................................................................
The following reported errors are for the entire design, not only for the set!
Number of minWidth violations: 0.
Number of minEdge violations: 0.
Number of minArea (including minEnclosedArea) violations: 0.
Number of manufacturing grid violations: 0.
The routing is clean!
==============================================================================
```

extend_pins

```
extend_pins
     [ -all
     | -fix_all
     | -set d_setObj
     | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -instance_pins_only [ true | false ] ]
     [ -fix_extension_routes [ true | false ] ]
     [ -report [ true | false ] [ -file s_filename ] ]
```

Extends any pin shapes that are not `minArea` and `minEndOfLineSpacing` compliant. You can specify whether the command operates on a specified set of pins, a region, or the entire design. This command should be run before routing. By default, processing is restricted to instance pins, but you can include top-level pin shapes. Shapes that are added by this command have the `TrimExtend` `routeType` property. You can delete these shapes by using `delete_routing` `-type trim_extend`. You can optionally set the `routeFix` property to `Fixed` for the added extension routes.

Important

You must set the `trimShape` and `trimMinSpacing` constraints before running this command.

Arguments

`-all` Processes all the pin shapes in the current design. This is the default.

`-fix_all` Processes hierarchically all kinds of geometries to fix the `minArea` and `minEndofLine` violations. In this mode, geometries are not limited to level-1 instTerms. They can include other objects such as unused pins, blockages, wires, and rectShapes.

`-file` *`s_filename`*

Outputs the summary of results to the specified file. This argument is valid only when `-report` is `true`.

`-fix_extension_routes [ true | false ]`

Sets the `routeFix` property for the new extension routes to `Fixed` when this is set to `true`. This prevents the autorouter from moving or deleting them but it can connect to them. By default (`false`), the `routeFix` property for the extension routes is set to `Unfixed`.

`-instance_pins_only [ true | false ]`

Processes only instance pins by default (`true`). When `false`, all the instance pins and top-level pin shapes are processed.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Processes pin shapes in the area specified by the lower-left (*`f_xlo f_ylo`*) and upper-right (*`f_xhi f_yhi`*) coordinates.

`-report [ true | false ]`

Outputs the summary of results to the Transcript area. The default is `false`.

`-set` *`d_setObj`* Processes only the pin shapes in the given set.

Value Returned

`0` Pins were extended.

`-1` No pins were extended or the command was not run due to a command error.

Examples

Extends instance pins inside the window area and outputs the results to the `report.log` file.

```
extend_pins -region [get_window_area] -report -file ./report.log
```

Extends instance pins and top-level pin shapes in the entire design.

```
extend_pins -instance_pins_only false
```

extend_wire_to_pin_edge

```
extend_wire_to_pin_edge
     [ -set d_setObj ]
```

Adds custom extents to all segments attached to pin shapes in the top cellview, extending the segments to the pin edges. Processing can optionally be restricted to end segments of nets in the given set.

Arguments

`-set` *`d_setObj`*	Restricts processing to end segments of nets in the given set.

fix_errors

```
fix_errors
     -error_types {[all] [minarea] [minedge ] [minenclosed] [mfggrid] [minwidth]
       [numcut] [extension] [rgrid] [robustpinconnection] [portshort] [crossing]
       [viastacklimit][minspacing]}
     [ -set d_setObj ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -check_mode {hard | soft} ]
     [ -disable_checks {all | min_edge_length | same_net_crossings} ]
     [ -eco_mode [ true | false ] ]
     [ -fix_min_area_at_pins [ true | false ] ]
     [ -num_tracks i_count ]
     [ -ripup [ true | false ]
     [ -threads i_threads ]
     [ -layers { all | {s_layerName …} } ]
```

Fixes violations in a routed design for a given type or all types currently supported by the command, in the entire cellview or in a given area. You can specify how much of the existing wire (`num_tracks`) can be removed in the violating areas, then re-routed to fix the errors. To work properly, constraints must be set for the error types being fixed.

Arguments

`-check_mode {hard|soft}`

Checks for and fixes violations for `hard` or `soft` constraints.

Default: `hard`

`-disable_checks {all | min_edge_length | same_net_crossings}`

Suppresses the specified type of design rule checking. By default, all geometry that is modified by this command will be DRC clean. Use of this option may result in design rule violations.

`all`	No design rule checking is performed.
`min_edge_length`	Suppresses edge length checks.
`same_net_crossings`	Suppresses same net crossing checks.

`-eco_mode [ true | false ]`

When set `true`, removes fill shapes that cause DRC violations for the processed nets and removes `gapFill` shapes that are no longer physically connected to their respective net.

Default: `false`

`-error_types {`*`s_type…`*`}`

Specifies the types of violations to fix.

Default: All error types are fixed except `crossing`, `portshort`, `extension` and `robustpinconnection`.

`all`	All error types supported by the command.
`crossing`	Crossing violations
`extension`	Extension violations
`mfggrid`	Manufacturing grid violations
`minarea`	Minimum area violations

`minedge`	Minimum edge length violations
`minenclosed`	Minimum enclosed area violations
`minspacing`	Minimum spacing violations
`minwidth`	Minimum width violations
`numcut`	Minimum number of cuts violations
`portshort`	Port short violations
`rgrid`	Routing grid violations
`robustpinconnection`	Pin connections where the length of the diagonal of the intersection rectangle between a wire or via and a pin is less than the width of the wire/via and the longest edge of the pin
`viastacklimit`	Via stack limit violations

`-exclude_net` **{***`s_netName`***…}**

Excludes the given nets from processing.

`-exclude_set` *`d_setObj`*

Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Excludes the given types of nets from processing.

`-fix_min_area_at_pins [ true | false ]`

Attempts to fix minimum area violations at pins that are not minimum area-compliant. Default is `false`.

`-layers { all |` **{***`s_layerName`* **…}** `}`

Restricts processing to the specified layers. The default is `all`; all routing layers are processed.

`-num_tracks` *`i_count`*

Specifies the maximum length, in tracks, of existing routing that can be ripped up and re-routed to fix an error. Default is 50.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

	Fixes violations in the area given by the lower-left (*f_xlo f_ylo*) and upper-right (*f_xhi f_yhi*) coordinates. By default, the entire top cellview is processed.
`-ripup [ true | false ]`	
	When `false`, existing routing cannot be ripped up and re-routed to fix errors. Default is `true` (rip up and re-routing are allowed).
`-set` *d_setObj*	Processes nets in the given set.
`-threads` *i_threads*	Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

Example

The following example fixes minimum area violations currently visible in the workspace.

```
fix_errors -error_types {minarea} -region [get_window_area]
```

Related Information

Tcl Command	check_grid check_min_edge_length check_minarea check_route_quality check_vias check_width

global_route

```
global_route
     [ -all | -net s_netName | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj
     ]
     [ -exclude_net {s_netName...} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -passes i_limit ]
     [ -start_pass i_number ]
     [ -pin_access_check ]
     [ -mode {full | incremental | ECO | steiner | floorplan | auto |
     searchAndRepair | steinerECO | optimize | init | bus} ]
     [ -threads i_threads ]
```

Global routes the active design, replacing all opens with global routes and rerouting input global routes to reduce congestion.

Arguments

`-all`

Global routes the nets in the active window. This is the default.

`-exclude_net` **{***`s_netName`* …**}**

Prevents listed nets from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-exclude_set` *`d_setObj`*

Prevents nets in the given set from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-exclude_type` **{**`[power][ground][clock]`**}**

Prevents nets of the given type from being processed. This argument is ignored if `-net` is given. By default, no nets are ignored.

`-mode {Full | Incremental | ECO | Steiner | Floorplan | Auto | searchAndRepair | steinerECO | optimize | init | bus}`

Specifies the global routing mode.

`auto`

Automatically determines the global routing mode. This is the default mode. If global route has not been run in the session, `Full` mode is invoked. If the global router has been run in the session, `Incremental` mode is used.

`bus`

Routes only buses and causes congestion analysis to choose the size of the gcell grid from the input buses. This mode is automatically run by bus_route.

`ECO`

Routes the modified nets and congested neighbors. Modified nets are given by the `-set` or `-net` option.

`floorplan` Assumes that the input design is from a floorplanner, and the core cells can overlap. Runs full mode, and modifies other arguments to produce reasonable results.

`full` Global routes all nets, starting with the `start_pass`, and proceeding through all of the global routing passes. Input global routes, if given, are used to seed the locations of the routes, but not the congestion. The default `start_pass` in full mode is 1.

`incremental` Global routes all nets, starting with the pass following the last completed global routing pass. The input design is expected to be routed. The input global routes seed the initial congestion values and the locations of the routes.

`init` Re initializes the global router when Space-based Router and Chip Optimizer is restarted. No routing is performed. Rebuilds the gcell grid and initializes all of the data structures. Particularly useful in an ECO flow.

`optimize` Removes hooks by replacing them with guides, then routing the guides.

`searchAndRepair` Looks for congested areas, lowers the congestion targets in those areas, then re-routes the affected nets. Can be called successively. Each pass of searchAndRepair will use a lower congestion threshold to determine what a congested area is, and will set a lower congestion target.

`steiner` Creates the shortest possible interconnection, also known as steiner routes. Can be used to seed the full global router.

`steinerECO` Creates the shortest possible interconnection for modified nets given by the `-set` or `-net` option.

`-net` *`s_netName`* Global routes the given net.

`-passes` *`i_limit`* Limits the number of passes that the global router can attempt.

`-pin_access_check` Causes the global router to examine all pins to ensure that they are accessible and to determine how they can be reasonably accessed. If this argument is not given, all standard cell pins are given “up” access only.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Global routes only in the area given by the lower-left (*`f_xlo f_ylo`*) and upper-right (*`f_xhi f_yhi`*) coordinates.

`-set` *`d_setObj`* Global routes the items in the set. If an item is a net, the entire net will be routed. If an item is not a net, the route corresponding to that item will be global routed.

`-start_pass` *`i_number`* Specifies the starting pass number for the global router. Defaults to 1 if mode is `full`.

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

Example

The following command global routes the entire design.

```
global_route
```

Here is an example of information that is output to the Transcript area when the `global_route` command is invoked. Congestion analysis is run at the start and end of each pass. The initial congestion analysis divides the design into square gcells, determines the resources available for each gcell, and outputs a congestion summary.

```
Begin congestion analysis...
   Tracks per gcell = 20   Average track pitch = 1.000   Master Unit = 1000
   Initializing gcell pattern...
      Initializing layer stack...
      End initializing layer stack  cpu:0.0sec user:0.0 16.6meg
   Gcell pattern loaded:  cpu:0.0sec user:0.0sec 16.6meg
   Adding gcells...
```

```
    Gcell grid constructed:   cpu:0.0sec user:0.0sec 16.7meg
    Chip assembly design (based on internal analysis)
    Begin gcell analysis...
    End gcell analysis:  cpu:0.0sec user:0.0sec 16.8meg
    Begin update routing congestion...
    End update routing congestion:  cpu:0.0sec user:0.0sec 16.8meg
    Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  --------------------------------------------------------------------------------------
  metal1:cell     65         0  ( 0.00%) 98   0   1   0   0   0   0   0   0   0    0    0
  metal1:edge     87         0  ( 0.00%) 97   0   0   1   0   0   0   1   0   0    0    0
  metal2:cell     65         0  ( 0.00%) 98   0   1   0   0   0   0   0   0   0    0    0
  metal2:edge     87         0  ( 0.00%) 97   0   0   1   0   0   0   1   0   0    0    0
  metal2:down-via 65         0  ( 0.00%)
  --------------------------------------------------------------------------------------
End congestion analysis
full routing...
Begin global router initialization...
End global router initialization
Begin global route...
# = LAYER DETAILS
========================================================================================
|        |   |    Total |    Horizontal     |     Vertical      |  Guide  |   Down Vias  |
|  Layer |Dir|   Length | Detail |  Global | Detail |  Global |  Length | Total | Guide|
|--------------------------------------------------------------------------------------|
| metal1 | V |  2466.85 |   0.00 |    0.00 |   0.00 |    0.00 | 2466.85 |     0 |    0 |
| metal2 | H |     0.00 |   0.00 |    0.00 |   0.00 |    0.00 |    0.00 |     0 |    0 |
|--------------------------------------------------------------------------------------|
| Totals |   |  2466.85 |   0.00 |    0.00 |   0.00 |    0.00 | 2466.85 |     0 |    0 |
| Percent|   |   100.00 |   0.00 |    0.00 |   0.00 |    0.00 |  100.00 |       |      |
========================================================================================
```

The first routing summary gives the guide lengths and the total lengths from the preroutes. Here, there is no global routing length.

```
    Begin 1st global routing pass....
    End 1st global routing pass:  cpu:0.0sec user:0.0sec 17.6meg
    Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  --------------------------------------------------------------------------------------
  metal1:cell     65         5  ( 7.69%)  9   6   1   9   3   3   3  23  13  20    0    0
  metal1:edge     87         7  ( 8.05%) 34   8   1   8   0   0   0  16  12  11    0    0
  metal2:cell     65         1  ( 1.54%)  9  10  12  15   4   6   1  15  15   7    0    0
  metal2:edge     87         4  ( 4.60%) 32  10  14  10   0   1   0  14  11   0    0    0
  metal2:down-via 65         0  ( 0.00%)
  --------------------------------------------------------------------------------------

# = LAYER DETAILS
========================================================================================
|        |   |    Total |    Horizontal     |     Vertical      |  Guide  |   Down Vias  |
|  Layer |Dir|   Length | Detail |  Global | Detail |  Global |  Length | Total | Guide|
|--------------------------------------------------------------------------------------|
| metal1 | V |  6513.09 |   0.00 |  956.22 |   0.00 | 5556.87 |    0.00 |     0 |    0 |
| metal2 | H |  5184.12 |   0.00 | 3787.76 |   0.00 | 1396.36 |    0.00 |   100 |    0 |
|--------------------------------------------------------------------------------------|
| Totals |   | 11697.21 |   0.00 | 4743.98 |   0.00 | 6953.23 |    0.00 |   100 |    0 |
| Percent|   |   100.00 |   0.00 |   40.56 |   0.00 |   59.44 |    0.00 |       |      |
========================================================================================
```

The congestion summary shows some overcongestion that must be resolved in succeeding routing passes. Here, all guide length is gone, replaced by global routes.

```
   Begin 2nd global routing pass....
   End 2nd global routing pass:  cpu:0.0sec user:0.0sec 17.6meg
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
 -----------------------------------------------------------------------------------------
  metal1:cell     65         4 ( 6.15%)   6   7   3   9   3   3   6  21  26   7    0    0
  metal1:edge     87         2 ( 2.30%)  33   6   3   6   0   1   3  17  12  12    0    0
  metal2:cell     65         1 ( 1.54%)   6   4  18  24   6   6   1   9  20   1    0    0
  metal2:edge     87         0 ( 0.00%)  31   6  13  18   3   0   1   9  11   4    0    0
  metal2:down-via 65         0 ( 0.00%)
 ------------------------------------------------------------------------------------------

# = LAYER DETAILS
=======================================================================================
|        |   |    Total |    Horizontal     |      Vertical     |   Guide |  Down Vias  |
|  Layer |Dir|   Length | Detail |  Global  | Detail |  Global  |  Length | Total | Guide|
|-------------------------------------------------------------------------------------|
| metal1 | V |  6410.71 |   0.00 | 1076.22 |   0.00 | 5334.49 |    0.00 |     0 |     0 |
| metal2 | H |  5224.18 |   0.00 | 3605.44 |   0.00 | 1618.74 |    0.00 |   112 |     0 |
|-------------------------------------------------------------------------------------|
| Totals |   | 11634.89 |   0.00 | 4681.66 |   0.00 | 6953.23 |    0.00 |   112 |     0 |
| Percent|   |   100.00 |   0.00 |   40.24 |   0.00 |   59.76 |    0.00 |       |       |
=======================================================================================

   Begin 3rd global routing pass....
   End 3rd global routing pass:  cpu:0.0sec user:0.0sec 17.6meg
   Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
 -----------------------------------------------------------------------------------------
  metal1:cell     65         2 ( 3.08%)   6   7   3   7   3   3  10  26  21   7    0    0
  metal1:edge     87         0 ( 0.00%)  33   5   4   6   0   1   6  16  12  12    0    0
  metal2:cell     65         1 ( 1.54%)   9   6  15  20   7   6   1  10  20   1    0    0
  metal2:edge     87         0 ( 0.00%)  33   4  12  18   0   4   0  10  11   4    0    0
  metal2:down-via 65         0 ( 0.00%)
 -----------------------------------------------------------------------------------------

# = LAYER DETAILS
=======================================================================================
|        |   |    Total |    Horizontal     |      Vertical     |   Guide |  Down Vias  |
|  Layer |Dir|   Length | Detail |  Global  | Detail |  Global  |  Length | Total | Guide|
|-------------------------------------------------------------------------------------|
| metal1 | V |  6265.75 |   0.00 | 1068.38 |   0.00 | 5197.37 |    0.00 |     0 |     0 |
| metal2 | H |  5298.82 |   0.00 | 3542.96 |   0.00 | 1755.86 |    0.00 |   104 |     0 |
|-------------------------------------------------------------------------------------|
| Totals |   | 11564.57 |   0.00 | 4611.34 |   0.00 | 6953.23 |    0.00 |   104 |     0 |
| Percent|   |   100.00 |   0.00 |   39.87 |   0.00 |   60.13 |    0.00 |       |       |
=======================================================================================

   Begin 4th global routing pass....
   End 4th global routing pass:  cpu:0.0sec user:0.0sec 17.6meg
   Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
 -----------------------------------------------------------------------------------------
  metal1:cell     65         1 ( 1.54%)   6   7   3   7   3   3   7  26  27   6    0    0
  metal1:edge     87         0 ( 0.00%)  32   6   4   6   0   1   3  19  14  10    0    0
  metal2:cell     65         1 ( 1.54%)   9   4  15  24   7   3   1  10  21   0    0    0
  metal2:edge     87         0 ( 0.00%)  32   4  13  18   3   1   0  10  12   3    0    0
  metal2:down-via 65         0 ( 0.00%)
 -----------------------------------------------------------------------------------------

# = LAYER DETAILS
=======================================================================================
|        |   |    Total |    Horizontal     |      Vertical     |   Guide |  Down Vias  |
|  Layer |Dir|   Length | Detail |  Global  | Detail |  Global  |  Length | Total | Guide|
|-------------------------------------------------------------------------------------|
| metal1 | V |  6296.25 |   0.00 | 1088.38 |   0.00 | 5207.87 |    0.00 |     0 |     0 |
| metal2 | H |  5268.32 |   0.00 | 3522.96 |   0.00 | 1745.36 |    0.00 |   104 |     0 |
|-------------------------------------------------------------------------------------|
| Totals |    | 11564.57 |   0.00 | 4611.34 |   0.00 | 6953.23 |    0.00 |   104 |    0 |
| Percent|    |   100.00 |   0.00 |   39.87 |   0.00 |   60.13 |    0.00 |       |      |
=======================================================================================

End global route:  cpu:0.2sec user:0.3sec 17.6meg
Begin writing global routes...

# = LAYER DETAILS
=======================================================================================
|        |   |    Total |    Horizontal    |      Vertical     |   Guide |  Down Vias  |
|  Layer |Dir|   Length | Detail |  Global | Detail |  Global |  Length | Total | Guide|
|-------------------------------------------------------------------------------------|
| metal1 | V |  6296.25 |   0.00 | 1088.38 |   0.00 | 5207.87 |    0.00 |     0 |    0 |
| metal2 | H |  5268.32 |   0.00 | 3522.96 |   0.00 | 1745.36 |    0.00 |   104 |    0 |
|-------------------------------------------------------------------------------------|
| Totals |    | 11564.57 |   0.00 | 4611.34 |   0.00 | 6953.23 |    0.00 |   104 |    0 |
| Percent|    |   100.00 |   0.00 |   39.87 |   0.00 |   60.13 |    0.00 |       |      |
=======================================================================================

   15 nets written.
   15 routes written.
End write global routes:  cpu:0.0sec user:0.0sec 17.6meg
Begin reclaiming memory...
End reclaiming memory:  cpu:0.0sec user:0.0sec 17.6meg
0
```

The final routing summary states how may nets and routes were created by the global router.

local_route

```
local_route
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -exclude_net {s_netName...} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -threads i_threads ]
```

Escapes pins in the entire top cellview or, optionally, in a given region or for nets in a set. Metal1 and poly pins are escaped to metal2. Any disconnects between the ends of the new connections and the existing global routes are connected with guides to keep the connectivity legal.

Arguments

`-exclude_net {s_netName …}`

Prevents listed nets from being processed.

Note: Unless they are fixed or locked, nets in this list may be shifted while routing other nets.

`-exclude_set d_setObj`

Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Prevents nets of the given type from being processed. By default, no nets are ignored.

`-region {f_xlo f_ylo f_xhi f_yhi}`

Processes routes in the area given by the lower-left (*f_xlo f_ylo*) and upper-right (*f_xhi f_yhi*) coordinates. By default, the entire top cellview is processed.

`-set d_setObj` Processes nets in the given set. By default, the entire top cell is routed.

`-threads i_threads` Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

optimize_routes

```
optimize_routes
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -maximize_cuts [none | useMinRule | useMaxRule | useViaDef] ]
```

Attempts to improve existing routing by reducing wrongway routing, removing unnecessary level changes, straightening wires, and, optionally, maximizing cuts on wide wires. This command does not operate on guides.

Important

> optimize_routes will be removed in a future release. You are encouraged to migrate immediately to route_optimize, which provides better performance and more options, including pin connection, taper, and via extent optimization.

Arguments

-exclude_net **{***s_netName* …**}**

Excludes the named nets from processing.

Note: Unless excluded nets are locked or fixed, they can be shifted when routing other nets.

-exclude_set *d_setObj*

Excludes nets in the given set from processing.

Note: Unless excluded nets are locked or fixed, they can be shifted when routing other nets.

-exclude_type **{**[power][ground][clock]**}**

Excludes nets of the specified types from processing.

Note: Unless excluded nets are locked or fixed, they can be shifted when routing other nets.

-maximize_cuts [none | useMinRule | useMaxRule | useViaDef]

When connecting to wide wires, will maximize the number of cuts per via after routing, where possible, according to the specified setting:

none (Default) No post-routing effort to maximize cuts.

useMaxRule Uses the maximum of via parameters from the applicable oaStdViaDef and existing via rules to maximize cuts.

useMinRule Uses constraint values for the cut layer and if not found, allows these constraints to be derived from the default oaStdViaDef.

useViaDef Uses via parameters from the applicable oaStdViaDef instead of using constraint values for the cut layer. No vias will be created that violate existing via rules.

-region **{***f_xlo f_ylo f_xhi f_yhi***}**

	Operates in the specified region, given the lower-left and upper-right coordinates. If not specified, operates on the entire top cellview.
`-set` *`d_setObj`*	Operates only on nets in the given set. If not specified, operates on the entire top cellview.

Example

The following example optimizes top-level routes currently in the workspace.

```
optimize_routes -region [get_window_area]
```

Value Returned

`0`	Command was run.
`-1`	The command failed due to a syntax or command error.

Related Information

Tcl Commands	optimize_jogs

p2p_route

```
p2p_route
     -set d_setObj
     [ -allow_violations [ true | false ] ]
     [ -undoable [ true | false ] ]
```

Replaces guides in the given set with wires.

Arguments

`-allow_violations [ true | false ]`

When `true`, routes the guides even when the routing causes DRC errors. This is typically used only after a DRC clean solution fails.

Default: `false`

`-set` *`d_setObj`*

Specifies the set. Space-based Router and Chip Optimizer will attempt to replace all guides in the set with routing.

`-undoable [ true | false ]`

Indicates whether the routing created by this command can be undone. By default, this is false.

Example

The following command replaces guides in the selected set.

```
p2p_route -set [get_selection_set]
```

route_optimize

```
route_optimize
     [ -all | -net s_netName | -set d_setObj | -nets {s_netName…} ]
     [ -exclude_set d_setObj | -exclude_net {s_netName…} ]
     [ -area { f_xlo f_ylo f_xhi f_yhi } ]
     [ -z_pattern_factor <positive integer> ]
     [ -vu_pattern_factor <positive integer> ]
     [ -max_iterations i_count ]
     [ -max_passes i_passes ]
     [ -prune [ true | false ] ]
     [ -process_rules_only [ true | false ] ]
     [ -allow_jump [ true | false ] ]
     [ -pin_optimize [ true | false ] ]
     [ -insert_offset_vias [ true | false ] ]
     [ -uniform_taper_to_first_via {none | useMinRule | useMaxRule | useViaDef |
     useSingle} ]
     [ -maximize_cuts {none | useMinRule | useMaxRule | useViaDef | useSingle} ]
     [ -make_only_net_spec_vias [ true | false ] ]
     [ -allow_width_expand_for_maxcuts [ true | false ] ]
     [ -via_orient [ true | false ] ]
     [ -pin_connect_optimization {default | truncate | cover | cover_io_only} ]
     [ -via_extent_optimization {default | truncate | cover} ]
     [ -reduce_wrongway [ true | false ] ]
     [ -allow_wrongway [ true | false ] ]
     [ -wrongway_tolerance i_tracks]
     [ -poly_contacts [ true | false ] ]
     [ -stdout_report [ true | false ] ]
     [ -exclude_pattern {[all][U][Z][L][V]} ]
     [ -layer { s_layerName… } ]
     [ -minus {[diffnet] [samenet] [width] [area] [enclarea] [portshorts]
     [crossing] [edge ] [grid] [mfggrid] [partial] [numcuts] [centerline ] [inter]
     [ext] [all]} ]
     [ -ignore_net_types
     {[pair][match][symmetry][shieldParallel][shieldTandem][shieldCoaxial][all]}
     ]
     [ -paired_nets [ true | false ] ]
     [ -symmetry_nets [ true | false ] ]
     [ -parallel_shielded_nets [ true | false ] ]
     [ -tandem_shielded_nets [ true | false ] ]
     [ -coaxial_shielded_nets [ true | false ] ]
     [ -z_shift [ true | false ] ]
     [ -use_taper_for_inline_maxcuts [ true | false ] ]
```

Attempts to improve existing routing using various techniques. By default, all nets in the design are processed, except matched length nets. You can optionally limit processing to named nets, nets and/or routes in an area or set, nets of specific types, and certain layers.

The routing is scanned for various types of patterns: U, Z, L, V, H. When a pattern is found, routing is optimized to reduce the number of jogs and layer changes, as shown in the following figures:

- U Pattern Optimization
- Z Pattern Optimization
- V Pattern Optimization
- L Pattern Optimization

You can optionally exclude any of these patterns and specify pattern threshold arguments.

Figure 11-13 U Pattern Optimization

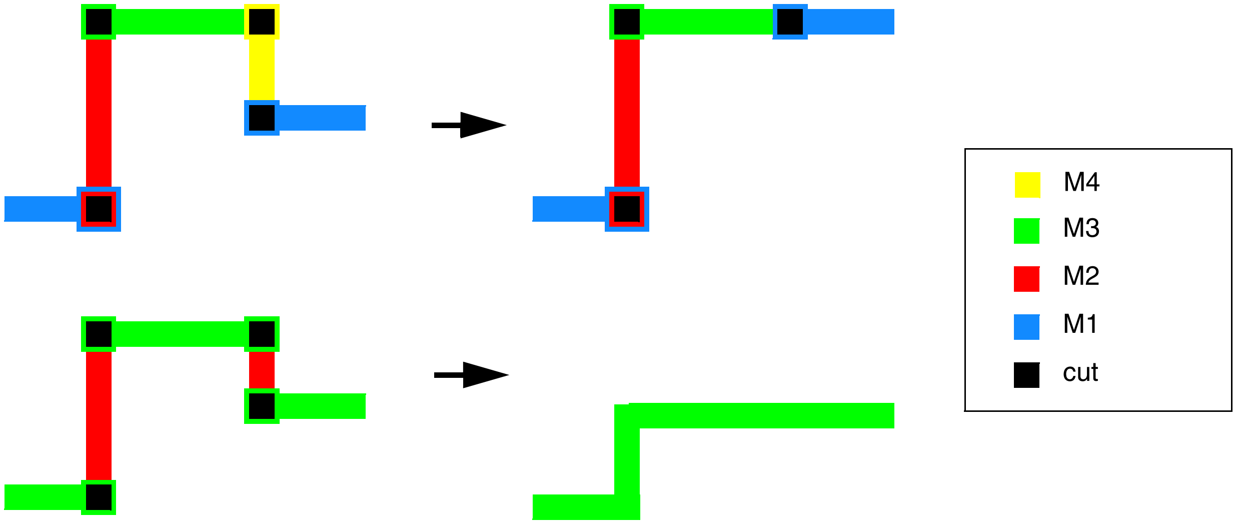


Figure 11-14 Z Pattern Optimization

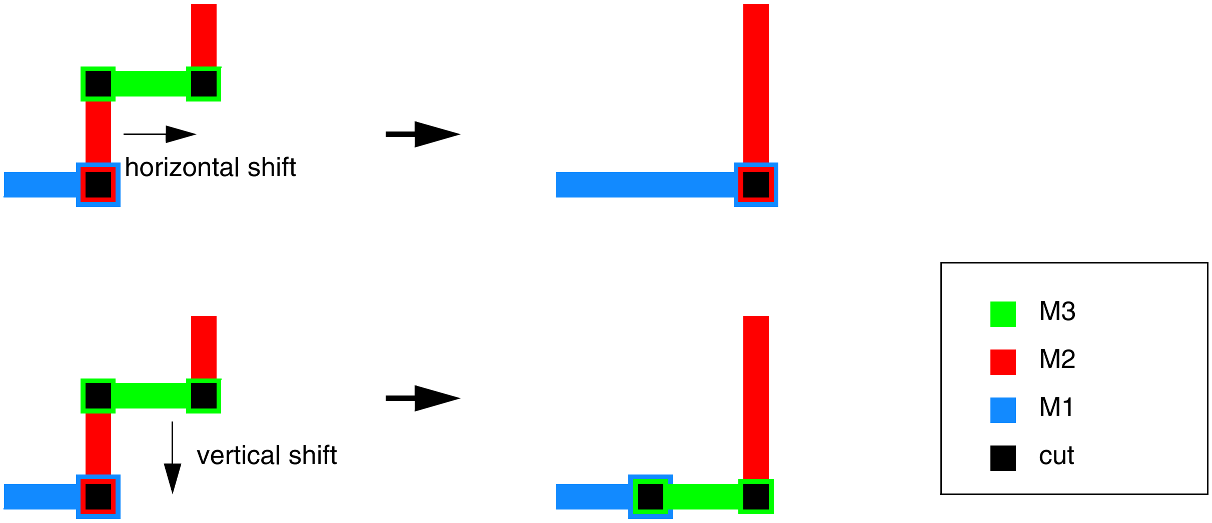


Figure 11-15 V Pattern Optimization

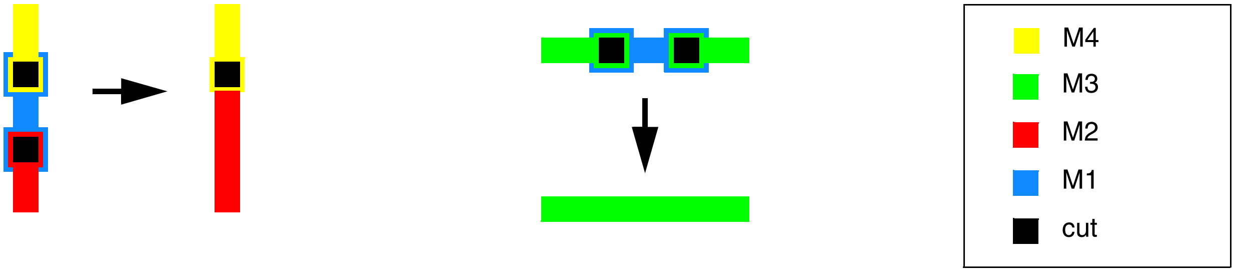
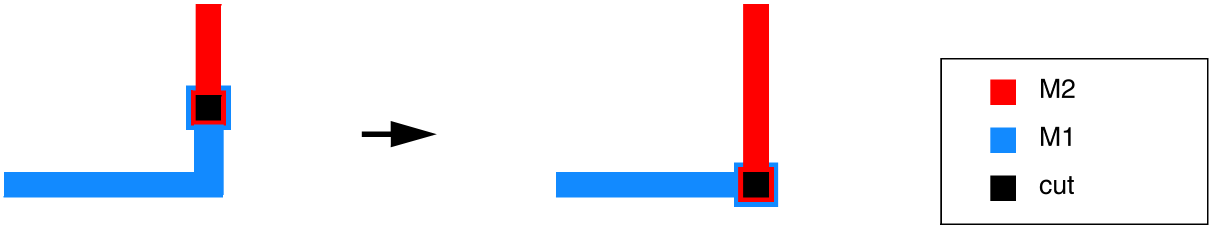


Figure 11-16 L Pattern Optimization



Other options let you specify optimization for the following:

- Pin Connection Optimization
- Taper Optimization
- Via Extent Optimization
- Via Optimization
- Pattern Threshold
- Miscellaneous

Arguments

`-all` Operates on the entire top cellview. This is the default.

`-allow_jump [ true | false ]`

When `true`, allows optimization of scenic routes using routing on a layer above or below to avoid obstructions, such as vias and pathSegments on other nets.

Default: Re-routing on other layers is not used.

`-allow_width_expand_for_maxcuts [ true | false ]`

When `true`, permits the wires to be widened in the overlap area between connecting wires to enable the use of maximized cuts (see Figure 11-27).

Default: The overlap area is limited to the width of the wires.

`-allow_wrongway [ true | false ]`

When `true`, allows the use of wrong-way routing to optimize more costly routing.

Default: Wrong-way routing is not used for optimizing.

`-area` **{** *`f_xlo f_ylo f_xhi f_yhi`***}**

Operates on the specified region, given the lower-left and upper-right coordinates. By default, operates on the entire top cellview.

`-coaxial_shielded_nets [ true  | false ]`

When set to `false`, coaxial shielded nets are not processed.

Default: Permits optimization of coaxial shielded nets.

`-exclude_net` **{***`s_netName…`***}**

Excludes nets in the list from processing.

`-exclude_pattern` **{**`[all][U][Z][L][V]`**}**

Specifies the patterns to be excluded during optimization. By default, all patterns are optimized.

`-exclude_set` *`d_setObj`* Excludes nets in the set from processing.

`-ignore_net_types {[pair] [match] [symmetry] [shieldParallel] [shieldTandem] [shieldCoaxial] [all]}`

Excludes processing nets of the given types. By default, all nets **except** matched nets are processed.

`-insert_offset_vias [ true | false ]`

When `true`, offset vias are used (see Figure 11-24).

Default: Centered vias are used during optimization.

`-layer` **{** *`s_layerName…`* **}**

Operates only on the specified layers. By default, all layers are processed.

`-make_only_net_spec_vias [ true | false ]`

When `true`, only oaStdViaDefs and cdsViaDefs in the `validRoutingVias` for the net can be used when creating vias for via maximization. By default, all defined oaStdViaDefs and cdsViaDefs can be used.

`-matched_nets [ true | false ]`

When `true`, matched length nets are optimized.

Default: Matched length nets are not processed.

Caution

Matched length nets use specific techniques for length matching which can be removed by route optimization.

`-max_iterations` *`i_count`*

By default, the command runs up to three (3) iterations of up to twenty (20) passes each (see Figure 11-32).

`-max_passes` *`i_passes`*

By default, the command runs up to twenty (20) passes for each iteration. The first pass for an iteration processes all required nets, then up to nineteen (19) additional passes are run on the impacted nets of the previous pass. Impacted nets are those surrounding optimized nets.

`-maximize_cuts {none | useMinRule | useMaxRule | useViaDef | useSingle}`

When connecting to wide wires, will attempt to maximize the number of cuts (see Figure 11-25) or the size of a single cut after routing, according to the specified setting:

`none`	(Default) There is no effort to maximize cuts.
`useMaxRule`	Uses the maximum number of via parameters from the applicable oaStdViaDef and existing via rules to maximize the number of cuts.
`useMinRule`	Uses constraint values for the cut layer and if not found, allows these constraints to be derived from the default oaStdViaDef. This is the default when `-maximize_cuts` is given with no setting.
`useSingle`	Uses constraint values for the upper metal to maximize the via using a large cut via. For an example, refer to Figure 11-26).
`useViaDef`	Uses via parameters from the applicable oaStdViaDef instead of using constraint values for the cut layer. No vias will be created that violate existing via rules.

`-minus` **{**`[diffnet] [samenet] [width] [area] [enclarea] [portshorts] [crossing] [edge ] [grid] [mfggrid] [partial] [numcuts] [centerline ] [inter] [ext] [all]`**}**

Excludes the specified checks during processing. By default, all checks are performed.

`-net` *`s_netName`* Operates only on the specified net.

`-nets` **{***`s_netName…`***}** Operates only on the specified nets.

`-paired_nets [ true | false ]`

When set to `false`, paired nets are not processed.

Default: Permits optimization of paired nets.

`-parallel_shielded_nets [ true | false ]`

When set to `false`, parallel shielded nets are not processed.

Default: Permits optimization of parallel shielded nets.

`-pin_connect_optimization {default | truncate | cover | cover_io_only}`

Specifies the wire extent for path segments over pins.

`default` (Default) The path segment extent will cover one-half (1/2) the pin width.

`cover` The path segment extent will cover the pin (see Figure 11-17).

`truncate` There will be no extent for the path segment over the pin (see Figure 11-18).

`cover_io_only` The path segment completely overlaps only the IO pins along the direction of the wire.

`-pin_optimize [ true | false ]`

Optimizes routes to pins; removes unnecessary jogs (see Figure 11-19).

`-poly_contacts [ true | false ]`

When `true`, aligns and maximizes cuts with the long axis of poly and contactless metal (see Figure 11-21).

Default: `false`

`-process_rules_only [ true | false ]`

When `true`, only process rules are applied. By default and when set to `false`, process rules and nondefault rules will be considered.

`-prune [ true | false ]`

When set to `true`, removes path segments with null endpoints. These are types of dangles.

Default: `false`

`-reduce_wrongway [ true | false ]`

When set to `true`, will attempt to re-route wires to reduce wrong-way routes.

Default: `false`

`-set` *`d_setObj`*

Processes only nets and/or routes in the set.

`-symmetry_nets [ true | false ]`

When set to `false`, symmetry nets are not processed.

Default: Permits optimization of symmetry nets.

`-tandem_shielded_nets [ true | false ]`

When set to `false`, tandem shielded nets are not processed.

Default: Permits optimization of tandem shielded nets.

`-uniform_taper_to_first_via {none | useMinRule | useMaxRule | useViaDef | useSingle}`

If this argument is given with no setting or a setting other than `none`, tapering will apply from the pin to the first via, and cuts will be maximized after routing, where possible, according to the specified setting (see Figure 11-20):

`none`	(Default) There is no change to tapers.
`useMaxRule`	Uses the maximum number of via parameters from the applicable oaStdViaDef and existing via rules to maximize the number of cuts.
`useMinRule`	Uses constraint values for the cut layer and if not found, allows these constraints to be derived from the default oaStdViaDef. This is the default when the argument is given with no value setting.
`useSingle`	Uses constraint values for the upper metal to maximize the via using a large cut via.

`useViaDef`	Uses via parameters from the applicable oaStdViaDef instead of using constraint values for the cut layer. No vias will be created that violate existing via rules.

`-use_taper_for_inline_maxcuts [ true | false ]`

When set to `true`, maximizes cuts based on the width of the taper route spec when a taper wire connects to a via or via stack and the next wire is oriented in the same direction as the taper wire. When set to `false`, cuts are maximized based on the width of the route or net route spec (see Figure 11-29).

Default: `false`

`-via_extent_optimization` **{**`default | truncate | cover`**}**

Specifies the coverage of path segment extents over vias.

`default`	(Default) If `-max_iterations` is greater than 0, then the path segment extent will be made large enough to cover the via; otherwise, no change is made to path segment extents over vias.
`cover`	The extent for the path segment is made large enough to cover the via (see Figure 11-23).
`truncate`	If the wire extent is greater than or equal to one-half the wire width, then the extent is trimmed to one-half the wire width (see Figure 11-22). If the wire extent is greater than 0 but less than one-half the wire width, then it is changed to a wire extent of 0.

`-via_orient [ true | false ]`

When `true`, allows via orientation to change so that vias are in line with the wires, overlapping the wire as much as possible, and reducing corners, making the routing as efficient as possible (see Figure 11-28).

Default: Orientation of vias is not changed.

-vu_pattern_factor *i_vuFactor*

Specifies the factor that determines when vertical U pattern optimization should be attempted according to the formula (W * `vu_pattern_factor < D)`, where W is the width of the wire to be removed and D is the length of the wire (see Figure 11-31).

Default: 1000

-wrongway_tolerance *i_tracks*

When wrong-way routing is allowed for optimization, wrong-way routing is limited to this number of tracks.

Default: 2

-z_pattern_factor *i_zFactor*

Specifies the factor that determines when Z pattern optimization should be attempted according to the formula (W * `z_pattern_factor >= D)` where W is the width of the wire to be removed and D is the length of the wire (see Figure 11-30).

Default: 1000

-z_shift [true | false]

When `true`, allows Z patterns to be moved to free space to allow movement of other wires when the pattern cannot be removed by optimization.

Default: If the Z pattern optimization is not successful for a Z pattern, the pattern is not moved.

Examples

The following example optimizes routing using larger single cuts and no wrong-way routing, without changing the topology.

```
route_optimize -maximize_cuts useSingle -exclude_pattern all
```

Pin Connection Optimization

Figure 11-17 -pin_connection_optimization cover

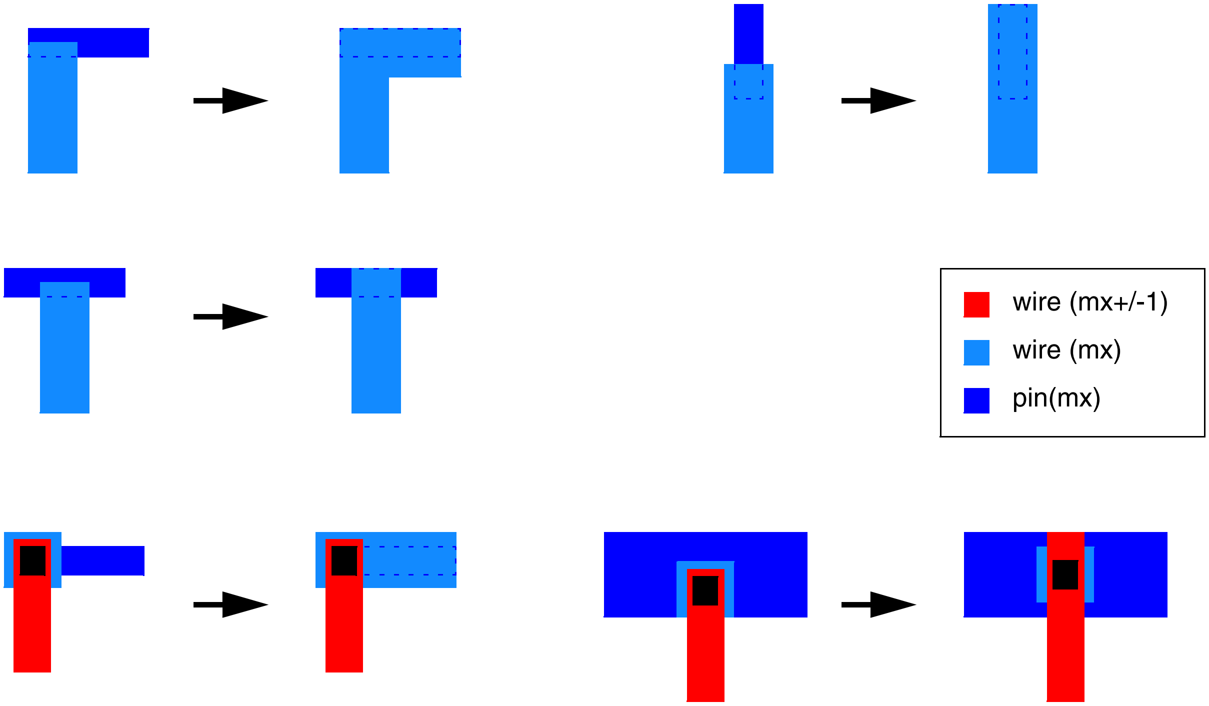


Figure 11-18 -pin_connection_optimization truncate

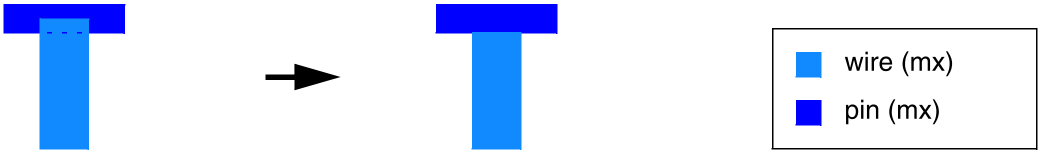
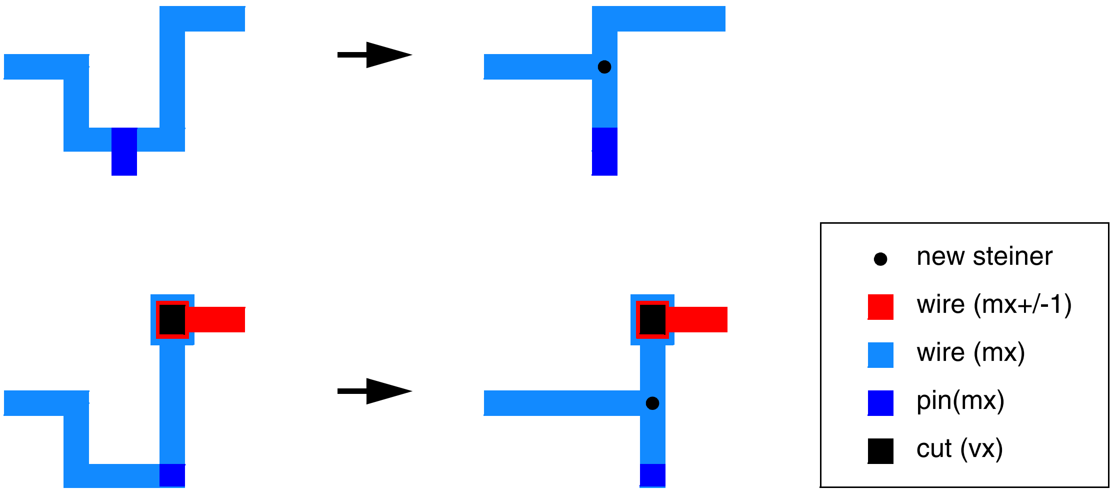


Figure 11-19 -pin_optimize true



Taper Optimization

Figure 11-20 -uniform_taper_to_first_via

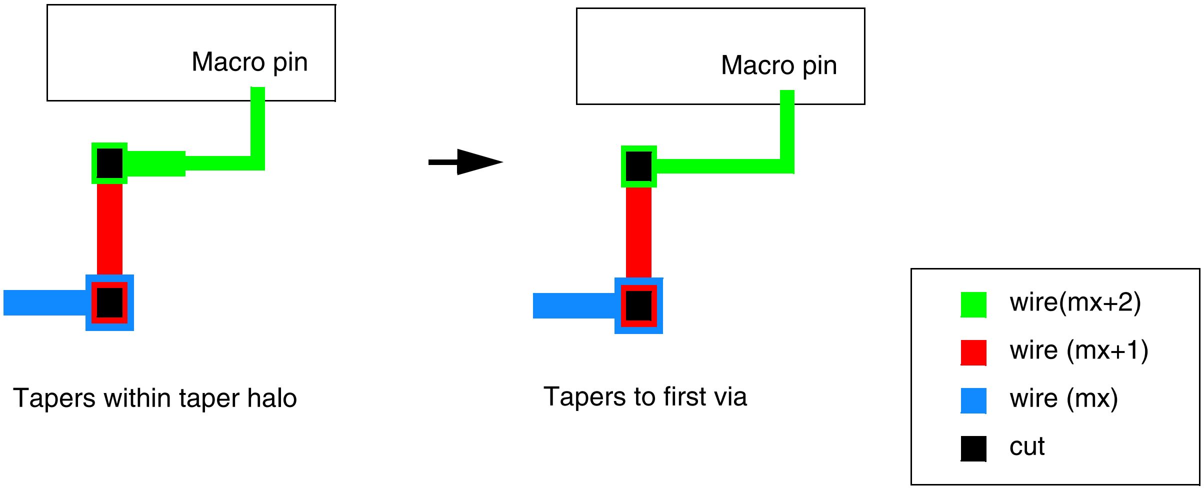
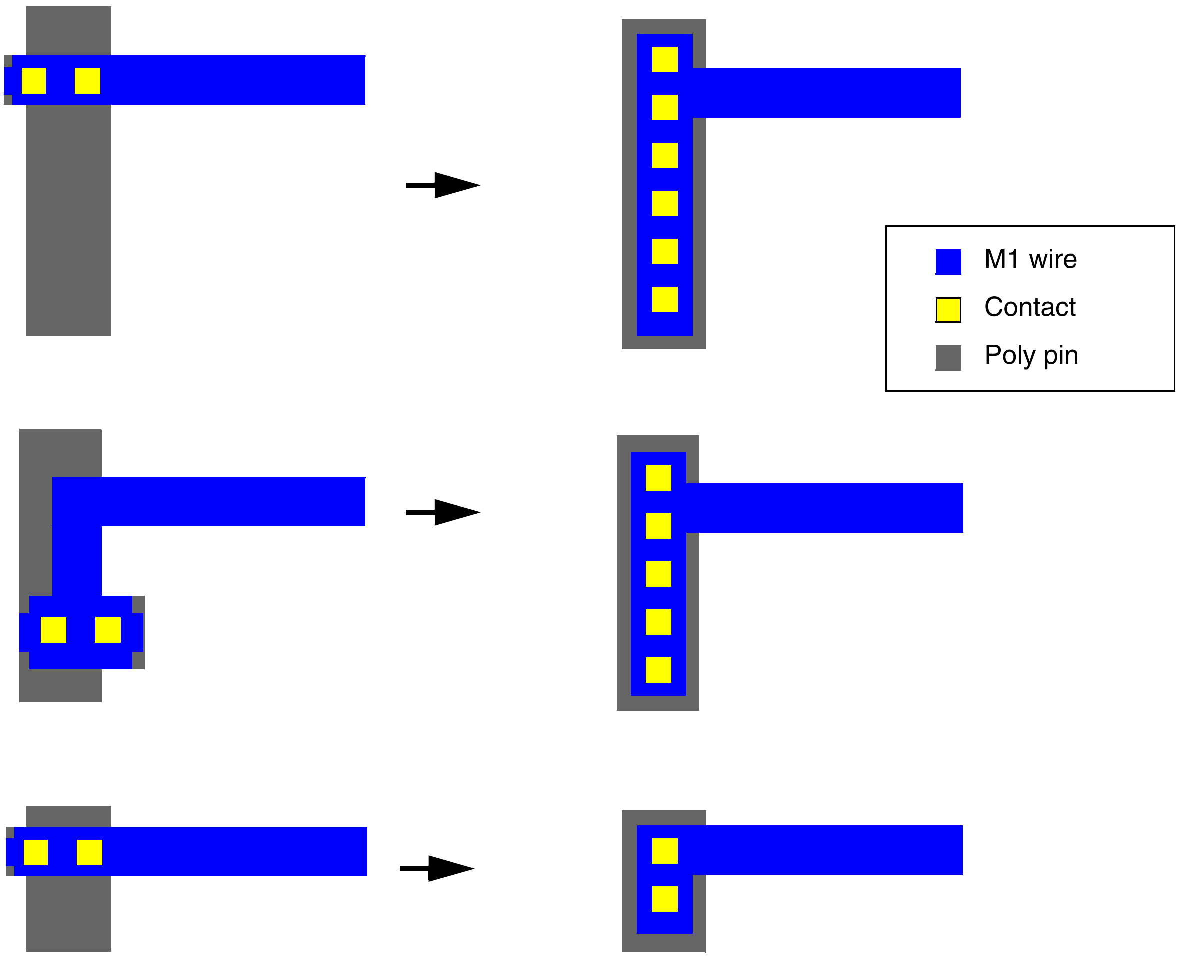
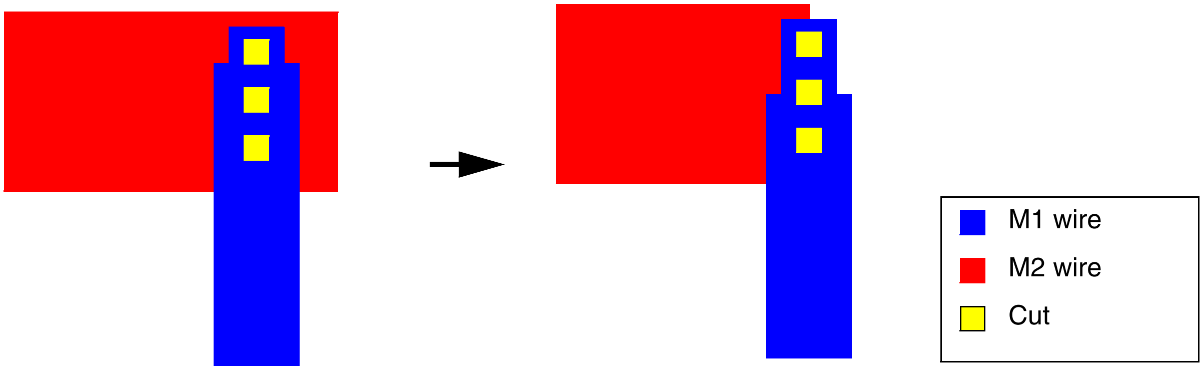


Figure 11-21 -poly_contacts true



Via Extent Optimization

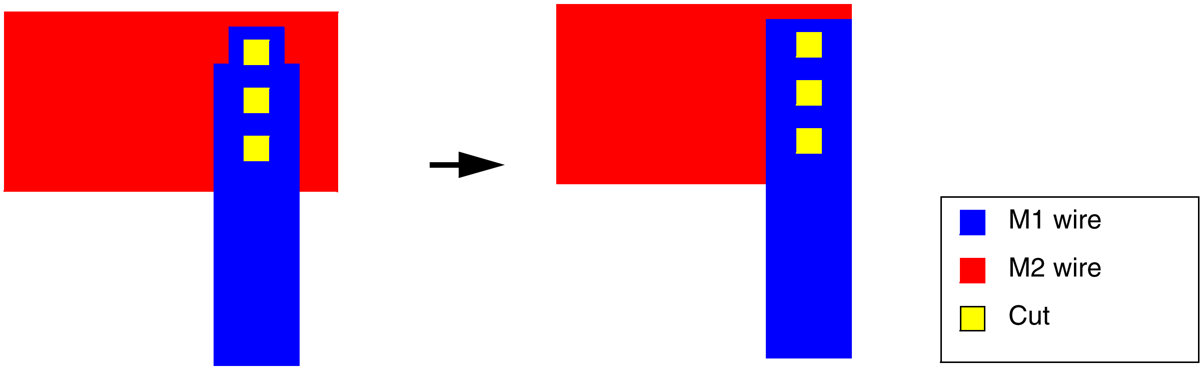
Figure 11-22 -via_extent_optimization truncate



Both M1 and M2 wire extents are >= 1/2 width of the overlapping wire.

During optimization, wire extents for both M1 and M2 are trimmed back to 1/2 the width of the overlapping wire.

Figure 11-23 -via_extent_optimization cover

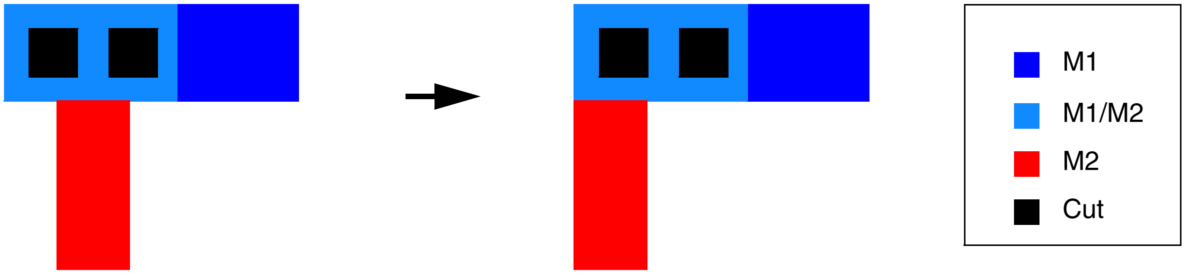


Both M1 and M2 wire extents are >= 1/2 width of the overlapping wire.

During optimization, the wire extents for both M1 and M2 are adjusted to cover the vias.

Via Optimization

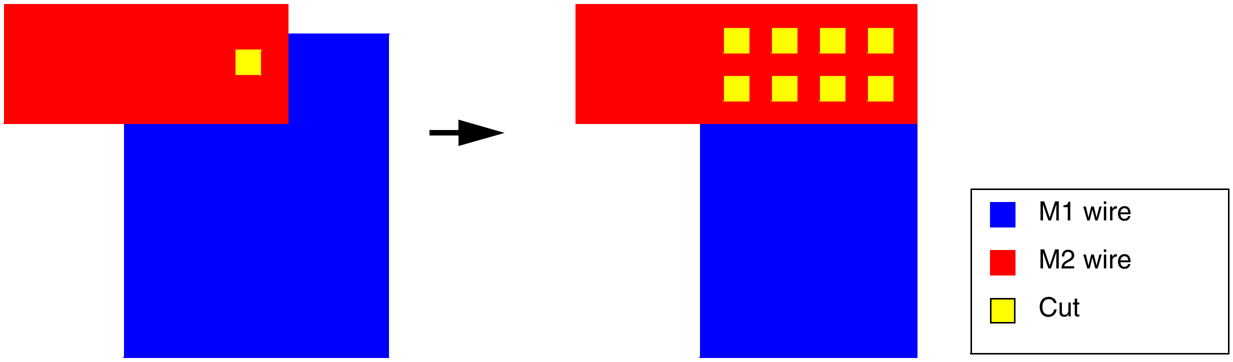
Figure 11-24 -insert_offset_vias true



By default, vias are centered on routing.

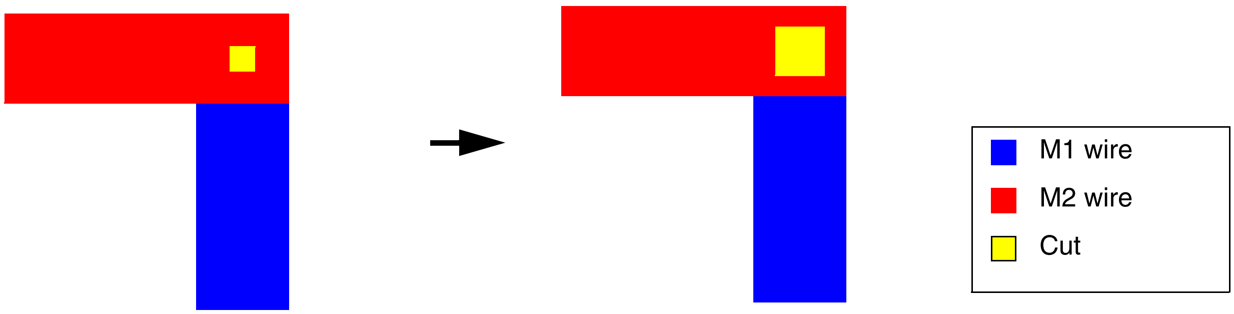
For `-insert_offset_vias true`, vias are offset on routing.

Figure 11-25 -maximize_cuts {useMinRule | useMaxRule | useViaDef}



These arguments attempt to maximize the number of cuts at connections.

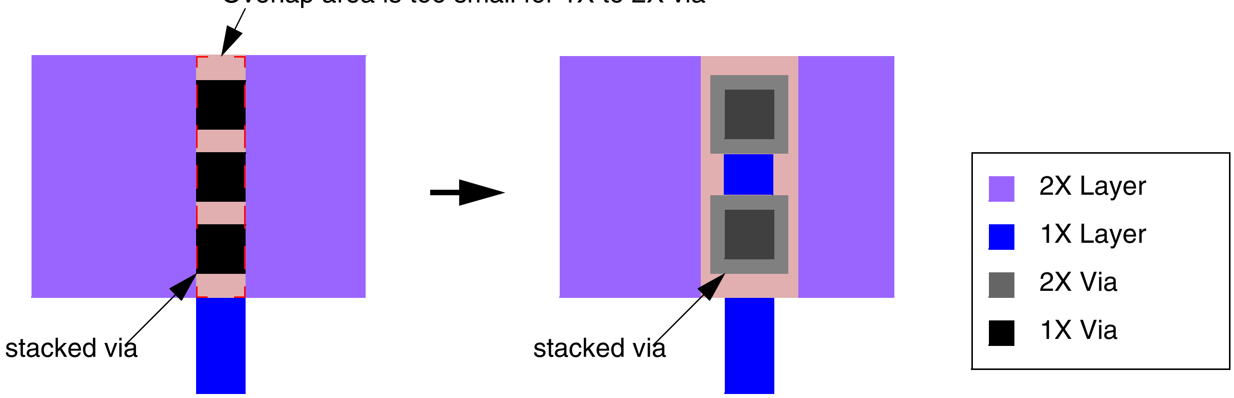
Figure 11-26 -maximize_cuts useSingle



`useSingle` attempts to replace single cuts with larger single cuts.

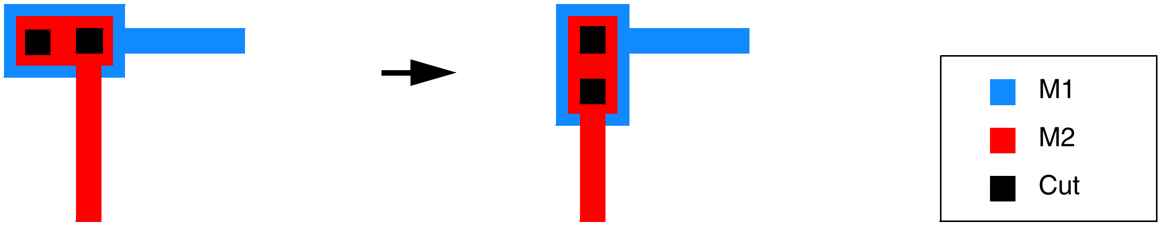
Figure 11-27 -allow_width_expand_for_max_cuts true

Overlap area is too small for 1X to 2X via



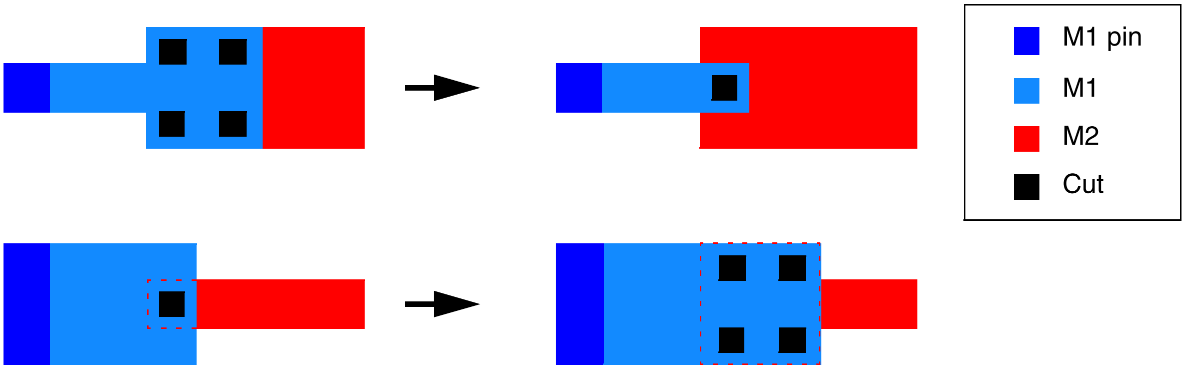
This argument allows wires to be widened to maximize cuts in the overlap area.

Figure 11-28 -via_orient true



The via orientation is changed from horizontal to vertical which results in improved routing.

Figure 11-29 -use_taper_for_inline_maxcuts

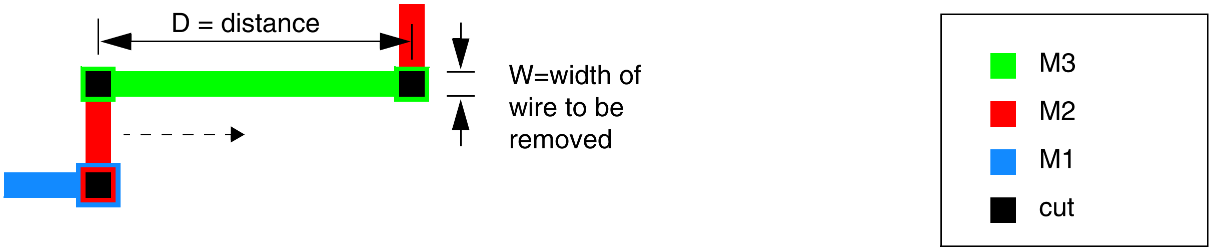


(Default/false) Maximizes inline cuts based on the width from the route spec of the net or route.

(true) Maximizes inline cuts based on the width of the taper spec.

Pattern Threshold

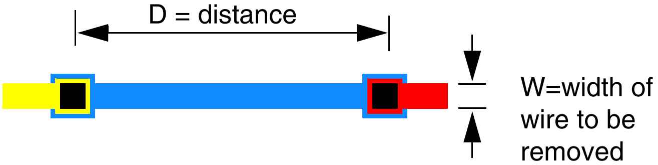
Figure 11-30 -z_pattern_factor

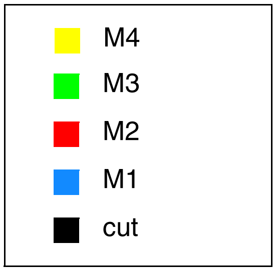


Z pattern optimization will be attempted if (W * z_pattern_factor >= D)

Default value for z_pattern_factor is 1000.

Figure 11-31 -vu_pattern_factor

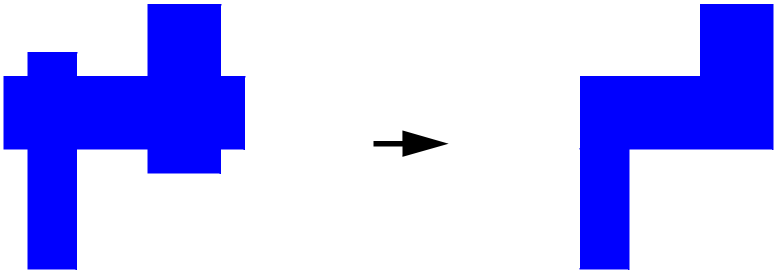




Vertical U-pattern optimization will be attempted if (W * vu_pattern_factor < D)

Default value for vu_pattern_factor is 1000.

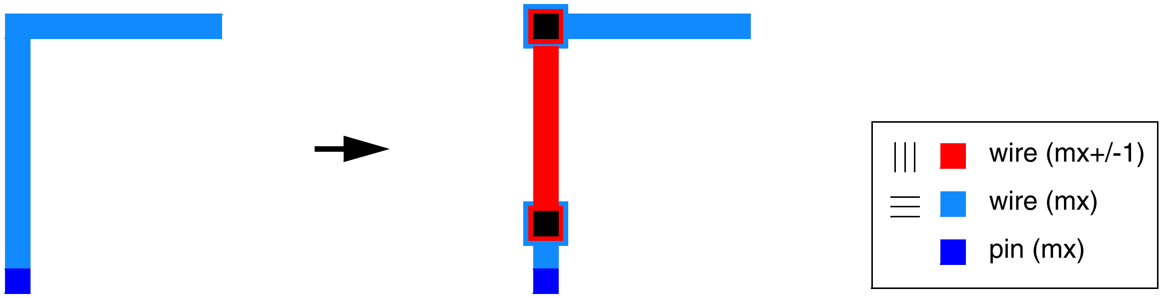
Figure 11-32 -max_iterations



If `-max_iterations` = 0, squaring off of extents, as shown above, will not occur.

Miscellaneous

Figure 11-33 `-reduce_wrongway`



search_and_repair

```
search_and_repair
     [-region {f_xlo f_ylo f_xhi f_yhi}]
     [-set d_setObj]
     [-exclude_net {s_netName...}]
     [-exclude_set d_setObj]
     [-exclude_type {[power][ground][clock]}
     [-use_check_annotations [true|false]]
     [-verbose [true|false]]
     [-calibre_rule_names {s_ruleName…}]
     [-close_opens [true|false]]
     [-check_level {0 | 1 | 2| 3}]
     [-check_mode {hard | soft}]
     [-eco_mode [true|false]]
     [-threads i_threads | -initial_check_threads i_threads]
```

Searches for and fixes same net and different net spacing violations. Optionally uses existing annotations that were created by check_space or by read_calibre_errors to identify the errors to be fixed. You can optionally use multiple processors for the search step and a single processor for the repair (`-initial_check_threads`).

Arguments

`-calibre_rule_names {s_ruleName…}`

Specifies the error marker names of Mentor Graphics® Calibre® rules to process. To use this option, you must have loaded the Calibre error file using read_calibre_errors.

`-check_level {0|1|2|3}` Specifies the level of checking to use. A lower number results in faster, but less accurate checking, a value of 3 performs complete checks.

`-check_mode {hard|soft}`

Determines whether to use preferred, then hard rules (`soft`), or only hard rules (`hard`) for checking and repairing. Default is hard rules only (`hard`).

`-close_opens [true|false]`

If `true`, the detail router will be used to close the remaining opens. If `false`, no attempt will be made to close the remaining opens. Default is `true`.

`-eco_mode [true|false]` When set `true`, removes fill shapes that cause DRC violations for processed nets and removes `gapFill` shapes that are no longer physically connected to their respective net.

Default: `false`

`-exclude_net {s_netName …}`

Prevents listed nets from being processed.

Note: Unless they are fixed or locked, nets in this list may be shifted while routing other nets.

`-exclude_set d_setObj` Excludes nets in the given set from processing.

`-exclude_type {[power][ground][clock]}`

Prevents nets of the given type from being processed. By default, no nets are ignored.

`-initial_check_threads i_threads`

Specifies the number of threads or processors to use in parallel to run the search step, then use a single processor to fix the violations. This option is useful when you have fewer than 1000 violations to process. Cannot be used with `-threads`.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Restricts processing to violations in the area given by the lower left (*`f_xlo f_ylo`*) and upper right (*`f_xhi f_yhi`*) coordinates. By default, the entire top cellview is processed.

`-set` *`d_setObj`*

Processes nets in the given set. By default, the entire top cell is routed.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel to run the search and repair steps of this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used. This argument cannot be used with `-initial_check_threads`.

`-use_check_annotations [true|false]`

If `true`, uses existing same net and different net spacing violation annotations to identify the violations to fix. You must run check_space to create the violations before using this option. If `false`, the search step is performed to identify the violations to fix. Default is `false`.

`-verbose [true|false]`

If true, outputs additional information to the transcript area. Default is `false`.

Related Information

Tcl Command

check_space
read_calibre_errors

show_congestion

```
show_congestion
     [ -cells [ true | false ] ]
     [ -edges [ true | false ] ]
     [ -off ]
```

Shows a pictorial representation of the amount of resources used by the router as determined by congestion_analysis. Each gcell is associated with a congestion score that represents the estimation of how congested that gcell is. By default, both cell and edge gcell representations are included.

If the gcell preferred layer direction is horizontal, the congestion score is represented by a vertical bar in the middle of the gcell. A thick bar corresponds to the area-type *cell* gcell, while a thin bar corresponds to an *edge* gcell. If the gcell preferred layer direction is vertical, the congestion score is represented by horizontal bars.

Each layer can be displayed independently. If several layers are displayed simultaneously, gcells having the same preferred layer direction are combined into a single score. If gcells are combined, the worst-case congestion percentage will be shown.

The congestion score is converted to colors using the following table:

% Congestion	Color
<70%	No color
70-80%	Blue
80-85%	Green
85-90%	Yellow
90-95%	Red
95-100%	Magenta
>100%	White (overcongested)

Tip

The color map is displayed on top of the layout design and only applies to the visible layers. To see which layer is congested, turn off all layers and then turn them on one-at-a-time.

Arguments

`-cells [ true | false ]`

Determines whether cell gcell representations are included. By default, cell gcell representations are included.

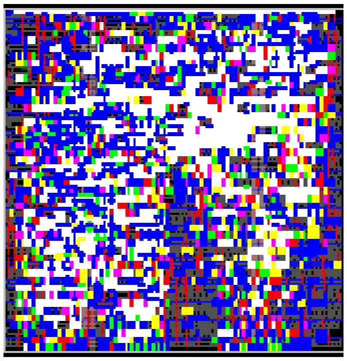
`-edges [ true | false ]`

Determines whether edge gcell representations are included. By default, edge gcell representations are included.

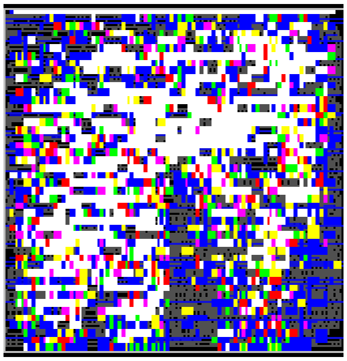
`-off` Turns off the congestion map.

Example

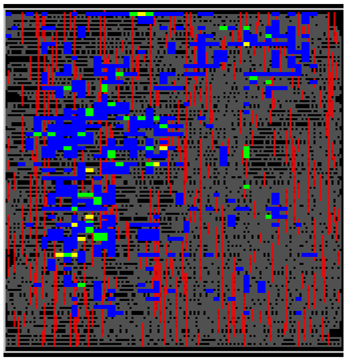
The following figures show congestion maps for a design. All layers are represented in the composite map. The other two maps isolate the congestion for Metal1 and Metal2, which are the only layers used for routing signals. The maps show that Metal1 contributes to the majority of the congestion, and areas colored white are the most severely congested.



Composite Map



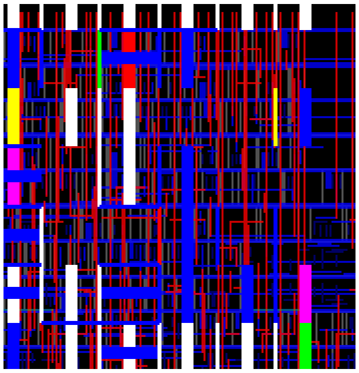
Metal1 Map



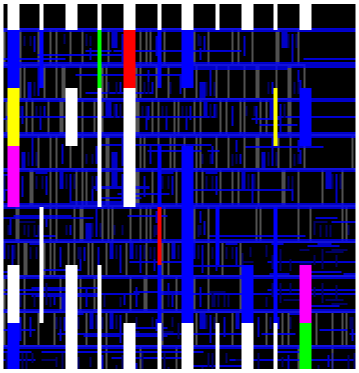
Metal2 Map

Zooming in to the upper-right corner of the design results in the following congestion maps. The closer view helps you to identify problem areas. In this example, Metal1 wires are blue, and Metal2 wires are red. Since the preferred direction for Metal1 is horizontal, the congestion bars for Metal1 are vertical. The thick bars represent congestion within gcells, while thin bars represent congestion between gcells. The preferred direction for Metal2 is vertical, so horizontal bars represent congestion for Metal2. In the Metal2 Zoom Map, the right half of the area has congestion under 70% and no congestion bars are drawn. The left side of the Metal2

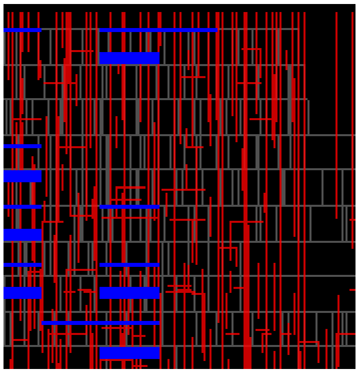
Zoom Map has some blue horizontal bars, both thick and thin, that identify gcells and edge gcells, respectively, with congestion between 70 and 80%.



Composite Zoom Map



Metal1 Zoom Map



Metal2 Zoom Map

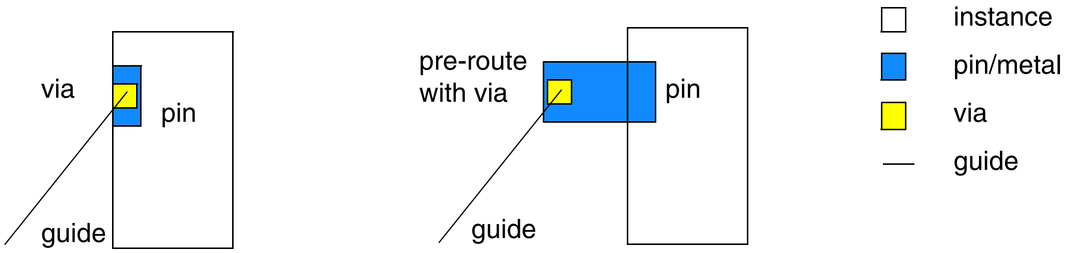
Related Information

Tcl Command	congestion_analysis display_color_map

wrongway_pin_escape

```
wrongway_pin_escape
     -set d_setObj
     [ -exclude_set d_setObj ]
     [ -exclude_term_set d_setObj ]
     [ -wrongway_preroute_status [none | split_and_lock | unsplit_and_unlock] ]
     [ -honor_existing_guides [ true | false ] ]
```

Routes pin escapes for wrong-way pins in the given nets. New guides are created for the nets unless `-honor_existing_guides` is specified. Usually a via is added on the pin, but if that is not possible (for example, due to a blockage), then a pre-route with a via is connected to the pin for pin access.



a) Via added for pin access

b) Pre-route and via added for pin access

You can optionally exclude nets in a set and/or terminals in a set. If the pin access is through a pre-route, the `routeFix` status of the pre-route can be set to `locked`.

Arguments

`-exclude_set` *`d_setObj`*

Excludes nets in the set.

`-exclude_term_set` *`d_setObj`*

Excludes terminals in the set.

`-honor_existing_guides [ true | false ]`

If `true`, prevents the removal of the existing guides to the wrong-way pins and the creation of new guides. Pin escapes will not be created for wrong-way pins without guides. Use this argument to use existing guides for specialty routing, such as spine routing. When this argument is not specified or is `false`, existing guides are removed and new guides are created using `update_net_connectivity`.

`-set` *`d_setObj`*

Adds pin escapes for the wrong-way pins on nets in the set.

`-wrongway_preroute_status [ none | split_and_lock | unsplit_and_unlock ]`

Sets the `routeFix` status of the pre-routes.

`none`

The `routeFix` status is not set on the pre-route.

`split_and_lock`

Splits the route from the wrong-way pin into two routes: one containing the pre-route and via, and the other containing the guide. The `routeFix` status of the pre-route is set to `locked`. This action can be reversed using the `unsplit_and_unlock` setting.

`unsplit_and_unlock`

Reverses the `split_and_lock` on wrong-way pin escapes, reverting two routes to one route. The `routeFix` status of the pre-route is set to `unfixed`.

Value Returned

`0`	Indicates that the command completed. Results are reported in the Transcript area.
`-1`	The command did not run.

Examples

The following example creates pin escapes for the wrong-way pins of the nets in set `s1`.

```
wrongway_pin_escape -set s1
```

The following example creates pin escapes for wrong-way pins of the nets in set `s1`. For pin escapes with pre-routes, the pre-route and via are assigned to a route and the guide is assigned to another route. The `routeFix` status for the pre-route is set to `locked`.

```
wrongway_pin_escape -set s1 -wrongway_preroute_status split_and_lock
```

Preparing the Routing Environment

Important

> Space-based Router and Chip Optimizer only operates correctly on centerline-connected routing data. If you import third party data that contains non-centerline-connected data and run update_net_connectivity, guides will be created and will appear as opens. To remedy this problem, use clean_nets. This command will establish centerline connectivity and remove loops, dangles, and redundant route elements without changing the footprint of the metal.

Before you begin routing, you can customize routing settings by:

- Choosing the Routing Mode
- Specifying the Grids
- Enabling/Disabling Routing on a Layer
- Routing Poly Layers
- Using Tapers
- Specifying Valid Routing Vias
- Using Via Abstraction
- Controlling Via Stacking
- Handling Blockages
- Setting the prBoundary Spacing
- Setting Spacing for Subcells
- Connecting IO Pins
- Creating Fences
- Setting Net Priorities
- Applying Hard and Soft Spacing Rules
- Protecting Existing Routes
- Setting the Routing Layer Direction in a Local Area
- Modifying Internal Routing Costs
- Minimizing Parallelism Effects

- Classifying Neighbor Nets
    - Avoiding Crosstalk
    - Half-Shielding Nets
- Setting Constraints
- Minimizing Potential Violations

Choosing the Routing Mode

Routing can be performed in gridless or gridded mode.

- In gridless mode, edges of all shapes must be on the *manufacturing* grid.
- In *strictly gridded* mode, the centerlines of route segments must be on a routing grid, and the origins of vias must be placed on the via grid.
- In *hybrid gridded* mode, the router will try to stay on a routing grid. If there are off-grid pins in your design, then the router will attempt to connect those pins on the manufacturing grid. This is the default.

To specify the routing mode, use

```
set_route_on_grid -style {hybrid_gridded | manufacturing | strictly_gridded}
```

Specifying the Grids

To set the routing grid for a metal layer, use `set_routing_grid`. For example:

```
set_routing_grid -layer M1 -x 0.28 -y 0.28 -x_offset 0.0 -y_offset 0.0
```

To set the routing grid for a via layer, use `set_via_grid`. For example:

```
set_via_grid -layer V12 -x 6 -y 8 -x_offset 3 -y_offset 3
```

To change the manufacturing grid for a layer, use `change_layer`. For example:

```
change_layer -name M1 -manufacturing_grid 0.05 -tech_lib $lib_name
```

Enabling/Disabling Routing on a Layer

To enable or disable routing on a layer, set the `limitRoutingLayers` constraint using the `set_constraint` command. For example:

```
set_constraint -constraint limitRoutingLayers -group s_routeSpec
-LayerArrayValue {s_layerName…}
```

The layers given by this constraint are ANDed with the layers given by the `validRoutingLayers` constraint to determine which layers to route on.

In the following example, the design's valid routing layers are read in as `M1`, `M2`, `M3`, `M4`, and `M5` for the `rs1` route spec, and the equivalent Tcl command is given as:

```
set_constraint -constraint validRoutingLayers -group rs1 -LayerArrayValue {M1 M2
M3 M4 M5}
```

For this example, the following command causes the router to use only `M2` and `M3` layers when routing nets on the `rs1` route spec.

```
set_constraint -constraint limitRoutingLayers -group rs1 -LayerArrayValue { M2 M3}
```

To enable routing on all valid routing layers, unset the `limitRoutingLayers` constraint. For example:

```
unset_constraint -constraint limingRoutingLayers -group rs1
```

Routing Poly Layers

Space-based Router and Chip Optimizer can only route on one poly layer at a time. Poly layers must be defined in the technology database. Routing on poly will be determined according to this precedence:

- `db.poly_layer_name` environment variable

  If this environment variable is set when the cellview is loaded, the named layer will automatically be recognized as the poly layer and will be a valid routing layer.

- `validRoutingLayers` constraint

  A poly layer named in this constraint is a valid routing layer. Only one poly layer may be included.

Using Tapers

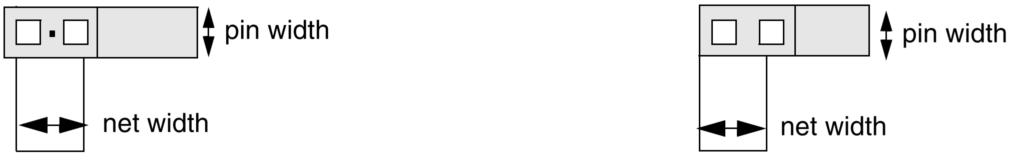
Tapering is needed when pins cannot be accessed due to one of the following conditions:

- The wire is wider than the width of the pin.
- The `validRoutingVias` for a net are so large that they will cause a DRC violation when accessing a pin.
- Spacing requirements cannot be met in the pin area because the pins are too close together.

Space-based Router and Chip Optimizer will automatically attempt to taper when one of these conditions exists. You can customize tapers, as described in "Custom Tapers" on page 791, otherwise the Default Taper Settings are used.

Transition vias are created dynamically, as needed, for tapering between metal layers. When double cut vias are required (`minNumCut` constraint value is 2), the default via origin is at the center of the cuts. When the `droute.offset_transition_vias` environment variable is set to `true`, double cut transition vias will be offset and aligned, as shown in Figure 11-34.

Figure 11-34 Illustration for droute.offset_transition_vias



a) When `droute.offset_transition_vias` is `false` (default), the transition via origin is centered between the cuts.

b) When `droute.offset_transition_vias` is `true`, transition vias are offset and aligned.

Default Taper Settings

- All non-default route specs taper to the global net default route spec (for example, `LEFDefaultRouteSpec`). This route spec will typically specify foundry constraints, such as `minWidth` and `minSpacing`, and smaller `validRoutingVias` that make it easier to complete the routing to/from the pin. The search order for taper constraints is described in *Scoping Taper Constraints*.
- When needed, tapering starts within a distance of 10 tracks from the pin.

Custom Tapers

For custom tapers, you can set constraints or issue commands to do the following:

- Taper to the width of pins on a net.
- Assign a route spec as the taper route spec and customize taper parameters (width, spacing, vias, and the maximum distance from the pin where tapering must start) for nets or terms.
- Force tapering by specifying `minTaperWindow` on the taper route spec. Even if tapering is not needed, setting this constraint forces tapering within this distance from the pin.

The following sections describe how to use Tcl commands for tapering:

- Setting Pin Width-based Tapering
- Setting Custom Tapering
- Removing Custom Tapers

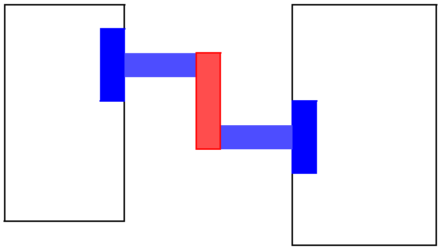
Setting Pin Width-based Tapering

For pin width-based tapering, use

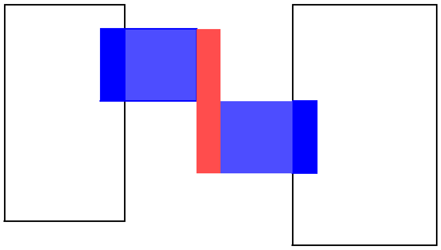
```
set_taper_width_nets {-all | -set d_setObj| -net {s_netName…}}
```

This command creates a taper spec for the terminals of the specified nets which forces the router to connect to those pins using the pin width in the access direction. The taper specs are assigned as `default` constraint groups to the routes connected to the terminals.

The following figure shows a comparison example of a net routed using the global net default route spec and the same net routed after `set_taper_width_nets` is issued for the net.



Instance terminals connected using the global net default route spec



Instance terminals connected after set_taper_width_nets is issued for the net

To remove a taper spec that was created by `set_taper_width_nets`, use `unset_taper_width_nets`.

For more information and additional options, refer to `set_taper_width_nets` in *Cadence Space-based Router and Chip Optimizer Command Reference*.

Setting Custom Tapering

To set a custom taper, you create a constraint group and assign it to a net or term. Then you set taper constraints and taper window constraints in the constraint group. If any constraint is not set in the constraint group, the value for the constraint is taken from the fallback sequence in the hierarchy.

To ensure that settings will be properly saved, you must specify the appropriate constraint group type for the object type: nets or terms.

- For nets, use **inputtaper** and **outputtaper** constraint groups for input and output tapers, respectively, or use the **taper** constraint group when the same constraints apply to all pins.
- For instance terminals, use the **taper** constraint group.

For more information on how constraint group types can be assigned to terms and nets, refer to the *Search Order for Taper Constraints*.

If a `taper` constraint group is not assigned to a term, the term will inherit taper constraints from the net's `inputtaper` or `outputtaper` constraint group, as appropriate for the pin type, and depending on whether these are assigned to the net. Otherwise, the term inherits constraints from the default taper constraint group, which is the global net default route spec (typically `LEFDefaultRouteSpec`).

For custom tapering,

1. Create a constraint group using **`create_constraint_group`**.

   ```
   create_constraint_group -name s_taperGroupName
   ```

2. Assign the new constraint group as either `inputtaper` or `outputtaper` (for nets), or `taper` (for instance terminals or nets) using **`set_constraint_group`**.

   For example, the following command sets the input taper group for a bit net:

   ```
   set_constraint_group -net s_netName -inputtaper s_taperGroupName
   ```

   The following command sets the output taper group for a bit net:

   ```
   set_constraint_group -net s_netName -outputtaper s_taperGroupName
   ```

   The following command sets the taper group for pins in a set:

   ```
   set_constraint_group -set $setOfInstTermsAndNets -taper s_taperGroupName
   ```

3. (Optional) Using **`set_constraint`**, specify the taper window using one of the following methods:

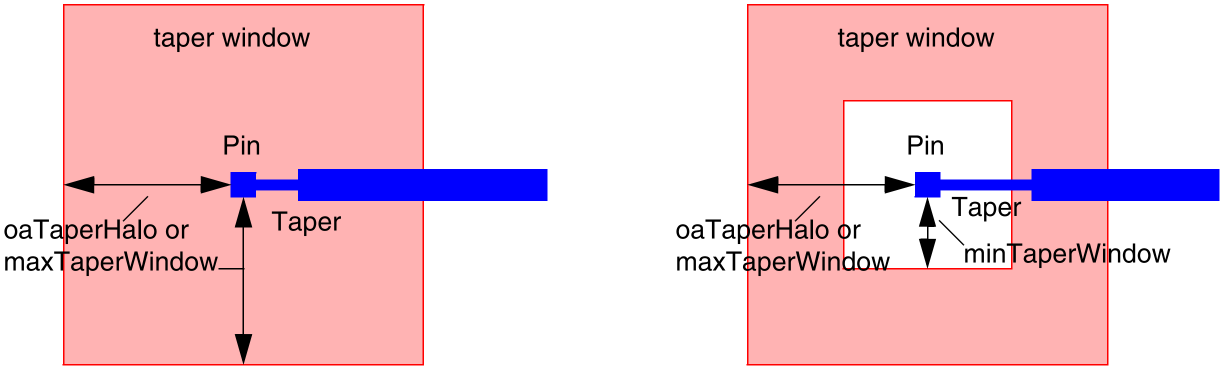
   - ❑ To taper to the first via or bend from instance pins or top-level pins, set the `taperToFirstVia` constraint to `true`.
   - ❑ To specify a taper window, as in Figure 11-35 on page 794, set these constraints:
     - ❍ `maxTaperWindow` or its OpenAccess equivalent, `oaTaperHalo`, to specify the maximum distance from the pin where tapering must start, if tapering is

needed. If both `oaTaperHalo` and `maxTaperWindow` are given, `oaTaperHalo` is used.

- ❍ `minTaperWindow` to force tapering and specify the minimum distance from the pin where tapering must occur.

These constraints are only recognized in a taper route spec. The default is 10 tracks from the pin, with no minimum distance required.

Figure 11-35 Taper Window Constraints



When needed, tapering must start within the shaded taper window area.

When `minTaperWindow` is set, tapering is required and must start within the shaded taper window area.

4. Specify the taper constraints, as needed:

- ❑ `minWidth`, `minSpacing`, and `minNumCut` (use **`set_layer_constraint`**)
- ❑ `validRoutingLayers` and `validRoutingVias` (use **`set_constraint`**)

Removing Custom Tapers

To remove a custom taper, use `set_constraint_group` with the appropriate taper group argument (-`taper`, -`inputtaper`, or -`outputtaper`), and do not include a constraint group name. For example, the following command removes the custom input taper for `net1`:

```
set_constraint_group -net net1 -inputtaper
```

Specifying Valid Routing Vias

Vias that can be used by the router must be specified in the `validRoutingVias` or `extendedValidRoutingVias` constraint. The `validRoutingVias` are typically defined in the design database. The extendedValidRoutingVias constraint can **only** be set by running the create_derived_vias command and can include the validRoutingVias. If extendedValidRoutingVias exists, it is used for the list of valid routing vias, otherwise, validRoutingVias is used.

This section describes the following:

- Types of Valid Routing Vias
- Getting Via Information
- Determining Which Via the Router Will Use
- Advanced Via Extension Control and Selection

Types of Valid Routing Vias

Table 11-1 describes the types of valid routing vias that can be used in a design.

Table 11-1 Types of Valid Routing Vias

Via Type	Description
standard vias	(Defined in the viaDefs section of the technology file) Typically, this is a set of standard vias between the poly layer and the highest metal layer. Standard vias are defined by a fixed set of parameters.
custom vias	(Defined in the viaDefs section of the technology file) Custom vias are based on cellviews that must exist in the design library.
via variants	(Can be defined in the viaDefs section of the technology file, or created using the create_via_variant command) These include variants of both standard and custom vias.
derived vias	(Created using the create_derived_vias command) Derived vias are design rule compliant for the standard vias specified in the validRoutingVias list for each route spec that is used in the cellview. You can specify the number and type of derived vias that are created based on available constraints and command settings.

Getting Via Information

To get specific information on a routing via, including cut layer and metal layer names, the number of cuts, columns, and rows, and preferred extension directions, use get_route_via_info. The information can be used to sort vias based on user-defined cost functions.

Determining Which Via the Router Will Use

The router computes the cost of the available vias based on each via's bounds on metal and cut layers, extension values, preferred orientation, and origin offset. A via with the smallest footprint and with longer extensions in the preferred routing direction will have the lowest cost. Vias with a lower cost are preferred for routing.

The selection of vias can be further controlled as described in the following sections:

- Advanced Via Extension Control and Selection
- Routing with Aligned Single-Cut Vias
- Routing with On-Wire Multi-Cut Vias

Advanced Via Extension Control and Selection

To further control the router's selection of vias, you can specify preferred extension directions and origin offsets, and also save a preferred via selection order for routing.

To use these features,

1. Enable the use of these features by setting the `db.use_separate_pref_ext_dir` environment variable to `true`.

   ```
   setvar db.use_separate_pref_ext_dir true
   ```

2. Set constraints, as described in "Specifying Via Preferences" on page 796.
3. (Optional) Customize the sorted list of valid vias, as described in "Reviewing and Changing the Sorted List of Valid Vias" on page 797.

Specifying Via Preferences

By setting the following constraints, you instruct the router to assign a lower cost to vias with your specified preferred extension direction, extension alignment, and/or origin offset.

- preferredExtensionDirection

When this constraint is set, a via with extensions aligned to the preferredExtensionDirection of the lower and upper metal layers will have the lowest cost and will be preferred for routing over another via that has the same number of cuts and the same extension dimensions but violates this constraint.

- inlineViaPreferred

  By setting this constraint to `true`, the preference is for inline vias, with longer extensions aligned with the longer dimension of the cut bound, and will override the preferred extension direction of the upper layer.

- preferredViaOrigin

  By default, the via origin does not affect the cost of a via. By setting this constraint, you can assign a preference (lower cost) to centered-at-origin vias or offset vias.

For more information on setting constraints, refer to *Cadence Space-based Router and Chip Optimizer Constraint Reference*.

Reviewing and Changing the Sorted List of Valid Vias

To get the sorted list of valid routing vias, use get_sorted_route_via_list. The returned list will include all valid routing vias for the design rule spec (default) or a given rule spec, sorted in order of increasing cost, as seen by the router. When `db.use_separate_pref_ext_dir` is `true`, then settings for `preferredExtensionDirection`, `inlineViaPreferred`, and `preferredViaOrigin` constraints will be considered when determining costs, otherwise those settings are ignored.

In addition, you can explicitly set the via preference order, by setting the list of valid routing vias (`validRoutingVias`) to your own ordered list of valid routing vias and setting `db.preserve_routing_via_order` to `true`. When `db.preserve_routing_via_order` is `true`, the router will select vias in the order specified by the list of valid routing vias and will assume that the cost of vias appearing earlier in the list will be lower than the cost of vias appearing later, irrespective of the via dimensions or any other parameters.

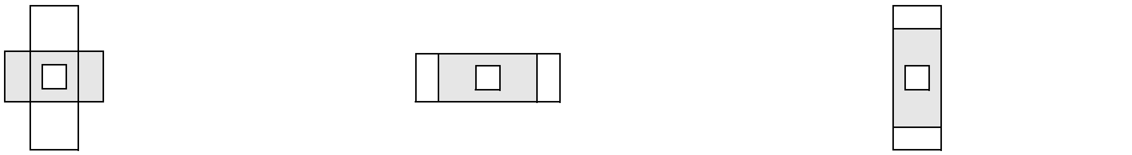
Routing with Aligned Single-Cut Vias

By default, the router will use vias with extensions in the preferred routing direction for the respective metal layers. To route using single cut vias with aligned extensions in the direction of routing, set the `droute.align_vias` environment variable to `true` before detail routing.

```
setvar droute.align_vias true
```

The aligned vias must be included in the `validRoutingVias` constraint or created using create_derived_vias.

Figure 11-36 Illustrations for droute.align_vias



a) Default `droute.align_vias false`

b) `droute.align_vias true` with horizontal routing direction

c) `droute.align_vias true` with vertical routing direction

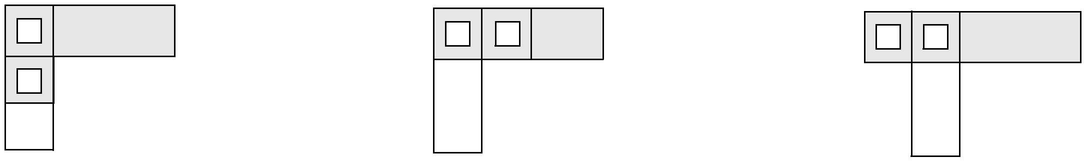
Routing with On-Wire Multi-Cut Vias

To prefer routing with on-wire vias, set the `droute.offset_vias` environment variable to `true` before detail routing.

```
setvar droute.offset_vias true
```

The vias must be included in the `validRoutingVias` constraint or created using create_derived_vias.

Figure 11-37 Illustrations for droute.offset_vias true



a) On-wire vias

b) On-wire vias

c) An off-wire via is used when the on-wire options result in design rule violations.

Using Via Abstraction

If your design uses multi-cut via masters in which the cut spacing is smaller than the minimum spacing rule, you can instruct Space-based Router and Chip Optimizer to treat vias as abstracts. When set true, via shapes inside the via master are not checked but bounding box-to-bounding box spacing is checked. By default, all via master shapes are checked against spacing rules. To treat vias as abstracts, use

```
set_treat_via_as_abstract -abstract true
```

Controlling Via Stacking

Via stacking is allowed by default.

To control the stackability of vias up and down, use

```
set_via_stackability -via s_viaName -up [ true | false ] -down
[ true | false ]
```

To check the stackability properties on all vias, use

```
report_via_stackability
```

Handling Blockages

All blockages and their constraints are loaded when you read in a design. To change this default behavior, you can set the following environment variables **prior to loading the design**:

- `db.load_blockages` determines whether to load standard cell blockages that originated in LEF. Default is `true`.
- `db.load_core_blockages` determines whether to load core cell blockages (usually top-level) that originated in the blockage section in DEF. Default is `true`.
- `db.enable_blockage_constraints` determines whether constraints for blockage shapes are read in. Default is `true`.

By default, the minimum spacing required by blockages is used for block-to-neighboring shapes spacing, regardless of the spacing rules for the neighboring shapes. You can change the default behavior by using the set_treat_blockage_as_metal Tcl command. This command uses four arguments to determine the way that blockages are treated: `min_width`, `min_space`, `override`, and `force_min_space`.

- The `min_space` and `min_width` arguments are mutually exclusive. These arguments determine how blockages without any pre-assigned constraints are to be treated.
    - `min_width` causes the checker to treat all unassigned blockages as if they have an effective width equal to the `minWidth` rule for the current layer. Thus, the minimum spacing is determined based on the `minWidth` value.

  - ❑ min_space causes the checker to use the minSpacing rule for the current layer as the minimum spacing for all unassigned blockages.

- The override argument causes all pre-assigned constraints to be ignored and the min_width and min_space arguments determine the spacing to be used for all blockages.
- The force_min_space argument, when set to true, causes the checker to use the spacing required by blockages for blockage-to-neighboring shape spacing, rather than using the larger of the minimum spacing for the blockage and the neighboring shape.

Using Dynamic Methods

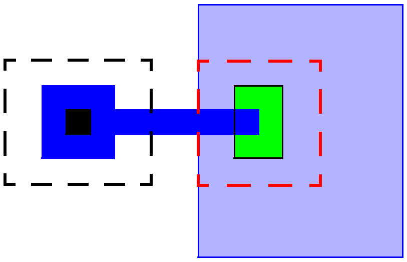
Dynamic checking lets Space-based Router and Chip Optimizer operate on designs without cutouts in the blockage shapes. It is not restricted to a per macro basis, does not change the original blockages and/or applied constraints, and does not create blockage:raw or blockage:pin purposes. With dynamic checking, top level blockages can intersect pin shapes at lower levels and be accounted as if they were cut.

Dynamic checking *masks* some violations based on the local interaction of the blockages, the pin shapes, and the route shapes. The router uses a similar method to determine routability.

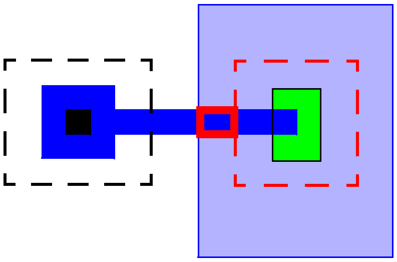
The guidelines for dynamic checking and routing are as follows:

- If a pin is wholly or partially embedded within blockage and is no farther than minimum spacing from the closest edge, the router can route to and connect to that pin from the edge of the blockage to which it is closest. This will not be considered a violation between the blockage and signal route that intersect.

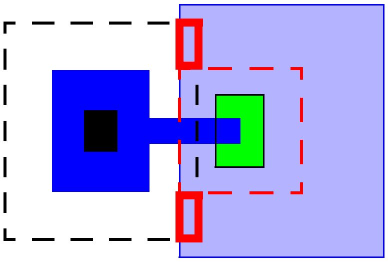
In this case, the shorted route segment and blockage violation is **masked** because the connecting pin is no farther than `minSpacing` from the edge of the blockage.



In this example, the shorted route segment and blockage is a **violation** because the pin is farther than `minSpacing` from the edge of the blockage.



This example shows a **violation** because the minimum spacing for the via shape overlaps the blockage area.



⬚ `minSpacing` bloat around pin

⬚ `minSpacing` around route shape

□ Violation area

- If a pin is wholly embedded in blockage and is farther than minimum spacing from the nearest blockage edge, that pin can be accessed only by dropping a via on the pin and the via metal must be totally enclosed within the pin shape. Any routing that does meet these requirements is considered a violation.

These guidelines apply equally to designs with or without cutouts.

Setting the prBoundary Spacing

If your design does not have existing routing outside of the prBoundary, you can use the environment variable, `db.make_prboundary_blockage`, to prevent the router from routing outside the prBoundary. You must set the variable prior to loading your design.

```
setvar db.make_prboundary_blockage true
```

When `db.make_prboundary_blockage` is set true when you load a design, blockage shapes for each routing layer are derived to represent the prBoundary and each of the blockage shapes is assigned a constraint group with the `minBoundaryInteriorHalo` constraint. This constraint specifies the minimum distance a shape on a specified layer must be from the enclosing prBoundary. The blockage shapes are used by the router and other functions, such as checking. If the `minBoundaryInteriorHalo` constraint is not already set, it defaults to one-half the minimum spacing (`minSpacing`) value. To change the minimum distance required between shapes and the enclosing prBoundary, use the `set_layer_constraint` command and specify the constraint group that is assigned to the prBoundary blockage shape. For example:

```
set_layer_constraint -layer Metal2 -constraint minBoundaryInteriorHalo -Value .3
-group myMetal2_minBoundaryInteriorHalo
```

Setting Spacing for Subcells

When data is loaded for a lower level of the hierarchy, blockage representing the subcell at that level is created at the higher levels. By default, spacing is assigned to the blockage according to the set_treat_blockage_as_metal setting described in "Handling Blockages" on page 799. If there are routes in the subcell that require special spacing, such as two times the minimum spacing, that spacing can be assigned to the blockage at the higher levels by setting the `db.load_as_detailed_abstract` environment variable.

```
setvar db.load_as_detailed_abstract i_level
```

where *`i_level`* is the hierarchy level for the lower level shapes. `minSpacing` for the route spec attached to lower level routes or nets will be used for the corresponding blockage.

Connecting IO Pins

By default the router will connect only one pin shape for each top-level IO terminal.To make the router connect to all pin shapes for each top-level IO terminal, **before** loading the design, use

```
    setvar db.connect_IO_pin_shapes true
```

Creating Fences

Use soft fences to control routing in a rectangular or polygonal area. When routing with fences the following guidelines usually apply:

- If all terminals of a net are outside of fence areas, all wiring for the net must be outside of fence areas.
- If all terminals of a net are inside of a fence area, all wiring for the net must be inside of the fence area.
- If terminals of a net are inside and outside of a fence area, then the wiring for the net can cross the fence to make the connection.

The soft fence boundary is viewable in the workspace and its visibility is controlled by the `area_boundary:soft fence` entry in the Object section of the Layer Object Display Panel.

To create a rectangular fence that covers the current workspace area,

```
create_soft_fence -name fence1 -region [get_window_area]
```

Setting Net Priorities

Use net priorities to more directly route critical nets or any given net. There are eleven levels of priority with 0 as the lowest priority and 10 as the highest priority. Net priorities can be set in DEF and by some data translation utilities. Unless you know that there are previously set net priorities that you want to keep, it is advisable to reset all net priorities prior to setting a few. The priority is relative, therefore, setting a net priority to 10 does nothing more than setting it to 1 over a default of 0. You can set net priorities for all nets, a given net, or nets in a set.

To reset the net priority for all nets, use

```
    set_net_priority -all -priority 0
```

To set the net priority to 1 for all nets whose names begin with `data_input_bus`, use

```
    replace_set -set1 [find_net -name {data_input_bus*} -ignore_case true
    -no_wildcard false ] -set2 [get_selection_set]
    set_net_priority -set [get_selection_set] -priority 1
```

The global router, conduit router, and detail router will honor net priorities. Nets with priorities greater than zero will be sorted and the nets with the highest priority will be routed first, thereby placing them close to their *ideal* position. During conduit routing, priority on nets with long segments will be more effective than priority on short nets because the primary goal of the conduit router is to track assign long route segments. Short segments are better handled by the detail router. At each of these three stages of routing—global, conduit, and detail—

priority nets are routed first, followed by all other nets. Priority nets, if set judiciously, should maintain their *ideal* topology through the routing flow. However, priority nets can be rerouted in order to cleanly complete other connections.

If a net is *long* due to some other constraint, you can apply your own formula to determine when the routing of a net is unacceptable and then re-route it using `global_route -mode eco`. Use `report_net_stats` and the Net Manager to determine the quality of the length of the net.

Applying Hard and Soft Spacing Rules

Hard rules must be followed, soft rules are preferred. You will typically prefer to route with soft rules but if routing fails to converge or detours excessively, hard rules can be applied to relax the requirements. Use set_soft_rule_adherence to choose what level of effort should be applied by the router to satisfy preferred spacing rules. The effort levels are described in Table 11-2 on page 804.

Table 11-2 Soft Spacing Rule Adherence Effort Levels

Effort	Description	Global Router	Conduit Router	Detail Router
Maximum	Router attempts to apply the soft rule, only abandoning the attempt if it fails to make the connection.	Use soft rule.	Use soft rule if space is available.	Use soft rule, but use hard rule if the point-to-point router fails with the soft rule.
High	Router attempts to make room for soft rules until several passes in detail route.	Use hard/soft heuristic algorithm to select where each rule can be applied.	Use soft rule if space is available.	Use the soft rule check in earlier passes and the hard rule in later passes.
Medium	Router attempts to make room for soft rules at earlier stages but abandons the attempt in detail route.	Use hard/soft heuristic algorithm to select where each rule can be applied.	Use soft rule if space is available.	Collect violations and route to the hard rule.

Effort	Description	Global Router	Conduit Router	Detail Router
Low	Router makes little effort to make space for soft rules, but it may use them if there is space available.	Use hard rule.	Use soft rule if space is available.	Collect violations and route to the hard rule.

To determine the current router effort level, use get_soft_rule_adherence. To check soft spacing rule adherence by layer and net, use check_route_quality `-checks softSpace`.

Protecting Existing Routes

If your design includes critical nets that you do not want the router to change, you can protect those nets by setting their status to `locked`. Locking the net will not permit any changes to it, including fixing any connectivity problems.

To set the net status to *locked*, use

```
    set_net_fix_status -net s_netName -status locked
```

Setting the Routing Layer Direction in a Local Area

Typically used for power grid orientation, you can set the routing layer direction in a local area by setting the preferred routing directions for the layers in a constraint group and specifying the rectangular or polygonal region to apply the routing directions to using `create_preferred_direction_region`. The preferred routing directions for the given layers will override the global settings in the region.

The following example sets the preferred routing direction for the M2 layer to be horizontal, and the preferred routing direction for the M3 layer to be vertical, within the given rectangular region.

```
create_constraint_group -name wrongWay -type route
set_layer_constraint -constraint oaPreferredRoutingDir -group wrongWay \
  -StringAsIntValue horzPrefRoutingDir -layer M2
set_layer_constraint -constraint oaPreferredRoutingDir -group wrongWay \
  -StringAsIntValue vertPrefRoutingDir -layer M3
create_preferred_direction_region -name region_wrongWayM2 -group wrongWay -region
{ -3230 -18 -2684 386 }
```

Modifying Internal Routing Costs

The `set_router_tax` command lets you adjust the internal costs used by the router. Currently, only the wrongway cost can be taxed and it is only recognized by the detail router. A wrongway tax specifies the relative cost of using a wrongway path to complete a connection versus using a preferred direction path and possibly additional routing. A real value greater than 0.0 and less than 100.0 can be specified and represents a multiplier to the internal wrongway cost set by the router. A tax greater than 1.0 tends to reduce the amount of wrongway routing and increase the number of vias used. A tax less than 1.0 and greater than 0.0 tends to increase the amount of wrongway routing used. The default value is 1.0.

For the current value of the wrongway router tax, use `get_router_tax`.

Minimizing Parallelism Effects

Long, parallel wires can affect signal integrity by introducing crosstalk. To minimize the effects of parallelism, you can specify minimum spacing requirements for different lengths of wire using `set_layer_constraint -constraint minSpacing`. For example:

```
set_layer_constraint -constraint minSpacing -layer M2 -row_name length
-OneDTblValue { 0 0.1 50 0.2 100 0.3 150 0.4 }
```

This command results in the following minimum spacing requirements for `M2`.

Length	Minimum Spacing
length<50	0.1
50<=length<100	0.2
100<=length<150	0.3
150<=length	0.4

Classifying Neighbor Nets

Prior to routing, you can set the `crossTalkNeighborIndex` constraint to classify the relationship between two net groups. The conduit router (`croute`) uses this constraint to determine the placement of wires and spacing between the nets. Nets can be considered to be good neighbors (route close to each other) or bad neighbors (keep apart).

To set the `crossTalkNeighborIndex` constraint, do the following:

1. Create two net groups of type `cross_talk`.

```
create_group -name s_group1Name -set d_setObj1 -type cross_talk
create_group -name s_group2Name -set d_setObj2 -type cross_talk
```

2. Create a constraint group.

```
create_constraint_group -name s_cg
```

3. Assign the constraint group to the two net groups at the group level.

```
assign_group_group -group1 s_group1Name -group2 s_group2Name
-group_group_spec s_cg
```

4. Set the `crossTalkNeighborIndex` constraint.

```
set_constraint -constraint crossTalkNeighborIndex -IntValue i_Val -group s_cg
```

The valid `crossTalkNeighborIndex` values and descriptions are given in Table 11-3.

Table 11-3 crossTalkNeighborIndex Constraint Values

Value	Description
0	Good neighbors. Route groups close together. This can be used to half-shield nets with power rails as described in Half-Shielding Nets.
1	Neutral. No special relationship between the net groups.
2	Bad neighbors. Route groups apart as described in Avoiding Crosstalk

Avoiding Crosstalk

To avoid crosstalk between nets, specify nets in one group that must be routed at a distance from the nets in a second group, and set the `crossTalkNeighborIndex` group-to-group constraint to 2. Alternatively, the same group of noisy nets can be given for both net groups, as is done in the Crosstalk Control Example, so that the conduit router will try to avoid assigning any pair of the given nets as immediate neighbors on adjacent tracks.

For example:

```
# Create the net groups as type 'cross_talk'
create_group -name group1 -set $selNetsSetA -type cross_talk
create_group -name group2 -set $selNetsSetB -type cross_talk

# Create a constraint group for bad neighbors
create_constraint_group -name bad_neighbor
set_constraint -constraint crossTalkNeighborIndex -IntValue 2 -group bad_neighbor

# Assign the bad neighbor constraint group to the two net groups
assign_group_group -net_group1 group1 -net_group2 group2 -group_group_spec
bad_neighbor
```

Half-Shielding Nets

To half-shield nets with power rails, create a group of power nets to use as shields, and a group of nets to be shielded. Then, set the `crossTalkNeighborIndex` group-to-group constraint to `0`. For example:

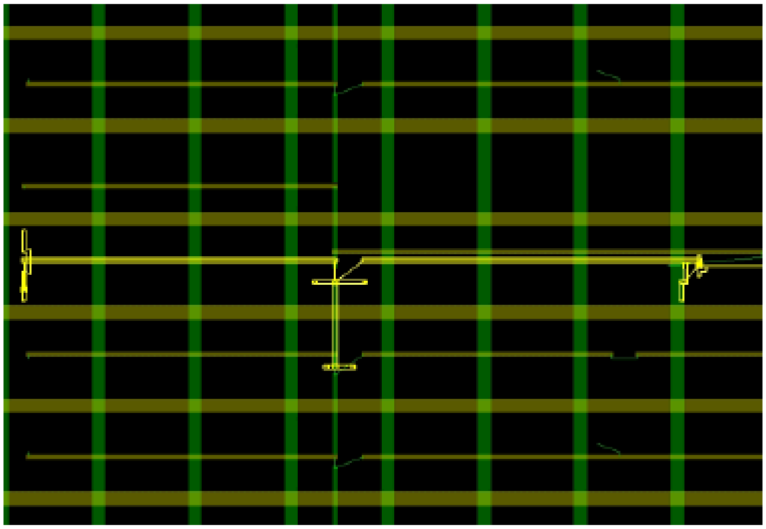
```
# Create the net group of nets to be shielded as type 'cross_talk'
create_group -name group1 -set $selNetsSet -type cross_talk

# Create a net group for the VDD net as type 'cross_talk'
set PWR [find_net -name {VDD}]
create_group -name group2 -set $PWR -type cross_talk

# Create a constraint group for preferred neighbors
create_constraint_group -name prefer_neighbor
set_constraint -constraint crossTalkNeighborIndex -IntValue 0 -group
prefer_neighbor

# Assign the preferred neighbor constraint group to the two net groups
assign_group_group -net_group1 group1 -net_group2 group2 -group_group_spec
prefer_neighbor
```

Crosstalk Control Example

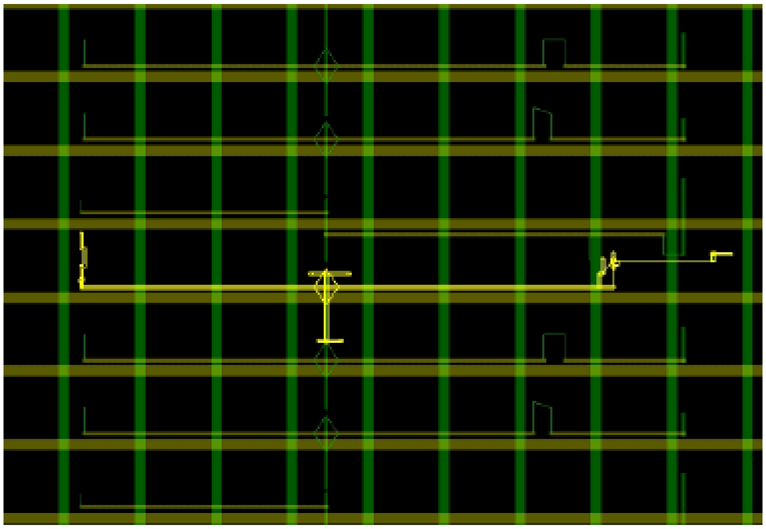
In the following example, two net groups are created: a group of noisy nets and a group of power nets. The `crossTalkNeighborIndex` is set to make the conduit router prefer to route the each noisy net next to a power stripe, while also avoiding routing noisy nets immediately next to each other.

```
# create a group of nets
set selNetsSet [find_name -name {someNets}]
create_group -name noisyNets -set $selNetsSet -type cross_talk

# create a group for power
set pwr_set2 [find_net -name {VDD}]
create_group -name pwrNets -set $pwr_set2 -type cross_talk

# Create a constraint group for encouraged coupling/parallelism
create_constraint_group -name prefer_neighbor
set_constraint -constraint crossTalkNeighborIndex -IntValue 0 -group
prefer_neighbor


# Create a constraint group for discouraged coupling/parallelism
create_constraint_group -name bad_neighbor
set_constraint -constraint crossTalkNeighborIndex -IntValue 2 -group bad_neighbor


# Create the preferred relationship between the noisy nets and VDD
assign_group_group -net_group1 noisyNets -net_group2 pwrNets -group_group_spec
prefer_neighbor

# Create the unpreferred relationship between all nets in the noisyNets group
```

```
assign_group_group -net_group1 noisyNets -net_group2 noisyNets -group_group_spec
bad_neighbor
```



In this example, crossTalkNeighborIndex is not set, so the conduit router is free to put the track anywhere in the free space.



Here, crossTalkNeighborIndex is set to 0 to make the net and VDD preferred neighbors, causing the conduit router to route the net next to the power net.

Setting Constraints

While some of the Tcl commands given in this section are used to set specific Space-based Router and Chip Optimizer constraints, there are many constraints that affect the routing of the design. For more information, refer to *Cadence Space-based Router and Chip Optimizer Constraint Reference*.

Minimizing Potential Violations

When constraints are set for your design, design rule checks will flag violations. Some violations can be minimized by setting conditions for routing. For example, a partially overlapping via over a pin shape can result in a minimum edge violation. To prevent this condition during routing, use

```
setvar droute.vias_must_be_fully_enclosed true
```

When `droute.vias_must_be_fully_enclosed` is `true`, Space-based Router and Chip Optimizer will only add vias at pins if the via can be fully enclosed in the pin shape. In addition, to restrict the vias that are affected by the fully enclosed requirement by specifying layers, use

```
setvar droute.vias_must_be_fully_enclosed_on_layer {layerName…}
```

Examples:

```
setvar droute.vias_must_be_fully_enclosed_on_layer {M01 M02}
setvar droute.vias_must_be_fully_enclosed_on_layer {Metal9}
setvar droute.vias_must_be_fully_enclosed_on_layer {} // same as all layers
```

The `droute.vias_must_be_fully_enclosed_on_layer` variable is only recognized when `droute.vias_must_be_fully_enclosed` is `true`.

A Routing Example

The following is an example of a simple Tcl routing script for Space-based Router and Chip Optimizer. The script assumes that you have already opened your design.

```
set_treat_via_as_abstract -abstract true
set metal_layers [get_layers -material metal]
set_treat_blockage_as_metal -layers $metal_layers -min_width false -min_space true
-override true -force_min_space false
global_route
local_route
croute
detail_route
```

Output Example

The following is an example of Transcript area output when the routing script is invoked.

```
set_treat_via_as_abstract -abstract true
0
set metal_layers [get_layers -material metal]
"metal1" "metal2" "metal3"
set_treat_blockage_as_metal -layers $metal_layers -min_width false -min_space true
-override true -force_min_space false
0
```

During global route, congestion analysis is run at the start and end of each pass. The initial congestion analysis divides the design into square gcells, determines the resources available for each gcell, and outputs a congestion (`Gcell`) summary.

```
global_route
Begin design analysis...
End design analysis 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 131.7MB vm, 131.9MB peak vm
Begin congestion analysis...
    Tracks per gcell = 20   Track pitch = 0.6500   Master Unit = 2000
    Layer Metal1 Wire-to-wire pitch = 0.6000 Wire-to-via pitch = 0.6500
    Layer Metal2 Wire-to-wire pitch = 0.6000 Wire-to-via pitch = 0.6500
    Layer Metal3 Wire-to-wire pitch = 0.6000 Wire-to-via pitch = 0.6500
Begin initializing gcell grid...
        Begin layer stack...
      End layer stack 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 131.7MB vm, 131.9MB peak vm
    End initializing gcell grid 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 131.7MB vm,
131.9MB peak vm
    Begin building gcells...
        Adding gcells...
   End building gcells 0.0s (kernel), 1.0s (user), 1.0s (elapsed), 133.6MB vm, 133.6MB peak
vm
    Standard cell design (based on internal analysis)
    Begin analyzing gcells...
    End analyzing gcells 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 133.6MB vm, 133.6MB
peak vm
    Begin update routing congestion...
    End update routing congestion 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 133.6MB vm,
133.6MB peak vm
    Gcell summary

  Layer/type   #Gcell   #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  ------------------------------------------------------------------------------------------
  Metal1:cell     2209      0 ( 0.00%)     0    4    3    3  10   18   24   23   11    0     0     0
```

```
    Metal1:edge     2162     0 ( 0.00%)    0   8   7   4   9  12   9  10  16  20    0    0
    Metal2:cell     2209     0 ( 0.00%)  100   0   0   0   0   0   0   0   0   0    0    0
    Metal2:edge     2162     0 ( 0.00%)   98   0   0   0   0   0   0   0   0   0    0    0
    Metal2:down-via 2209     0 ( 0.00%)
    Metal3:cell     2209     0 ( 0.00%)   97   2   0   0   0   0   0   0   0   0    0    0
    Metal3:edge     2162     0 ( 0.00%)   93   6   0   0   0   0   0   0   0   0    0    0
    Metal3:down-via 2209     0 ( 0.00%)
    ----------------------------------------------------------------------------------------
End congestion analysis 0.0s (kernel), 1.0s (user), 1.0s (elapsed), 133.6MB vm, 133.6MB peak
vm
```

The congestion summary indicates the following:

- The number of gcells for each metal layer by type: cell, edge, and down-via
- The number and percentage of overcongested gcells for each grouping
- The gcell resource usage (The `<10%` column is the number of gcells using less than 10% of their resources, the `10%` column is the number of gcells using at least 10% but less than 20% of their resources, and so on)
- The number of gcells with 100% blockage (`Blk%`)

```
Full routing...
Begin global router initialization...
    Automatic layer assignment is enabled
    Full pin access check is enabled
    Detailed pin escape is disabled
    Local detail route is disabled
    Begin update routing congestion...
    End update routing congestion 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 132.5MB vm,
133.6MB peak vm
    Begin initializing guides...
   End initializing guides 0.0s (kernel), 1.5s (user), 1.5s (elapsed), 134.1MB vm, 134.9MB
peak vm
    Begin pin access check...
    End pin access check 0.0s (kernel), 2.5s (user), 2.5s (elapsed), 134.1MB vm, 134.9MB
peak vm
    Begin prepare nets...
   End prepare nets 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 134.1MB vm, 134.9MB peak vm
    Metal1 layer density = 70%
    Metal2 layer density = 70%
    Metal3 layer density = 100%
Begin net prioritization...
   End net prioritization 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 134.7MB vm, 134.9MB
peak vm
End global router initialization 0.0s (kernel), 4.1s (user), 4.1s (elapsed), 134.7MB vm,
134.9MB peak vm
Begin global route...
        #Priority net count:
        Class   0
             5697
        ####################
# = LAYER DETAILS
============================================================================================
|        |   |    Total |     Horizontal     |      Vertical      |   Guide |  Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|------------------------------------------------------------------------------------------|
| Metal1 | H | 360495.55|   0.00 |    0.00 |    0.00 |    0.00 |360495.55|     0 |    0 |
| Metal2 | V |    459.47|   0.00 |    0.00 |    0.00 |    0.00 |  459.47 |    43 |   43 |
| Metal3 | H |      0.00|   0.00 |    0.00 |    0.00 |    0.00 |    0.00 |    23 |   23 |
|------------------------------------------------------------------------------------------|
| Totals |    | 360955.02|    0.00 |     0.00 |     0.00 |     0.00 |360955.02|    66 |   66 |
| Percent|    |   100.00 |    0.00 |     0.00 |     0.00 |     0.00 |  100.00 |       |      |
==========================================================================================
```

The first routing summary gives the guide lengths and the total lengths from the preroutes. Here, there is no global routing length.

```
Begin 1st global routing pass...
    End 1st global routing pass 0.0s (kernel), 2.8s (user), 2.9s (elapsed), 140.0MB vm,
140.6MB peak vm
    Gcell summary
  Layer/type   #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  ---------------------------------------------------------------------------------------
  Metal1:cell    2209        0 ( 0.00%)   0   4   3   3  10  18  24  23  12   0    0    0
  Metal1:edge    2162        0 ( 0.00%)   0   8   7   4  10  12   9  10  16  20    0    0
  Metal2:cell    2209        0 ( 0.00%)   9   7  14  18  20  20   8   0   0   0    0    0
  Metal2:edge    2162        0 ( 0.00%)   7  10  14  18  21  17   8   0   0   0    0    0
  Metal2:down-via2209        0 ( 0.00%)
  Metal3:cell    2209        0 ( 0.00%)   8  11  15  17  16  12  12   5   1   0    0    0
  Metal3:edge    2162        0 ( 0.00%)   6   9  15  17  15  14  12   6   1   0    0    0
  Metal3:down-via2209        0 ( 0.00%)
  ---------------------------------------------------------------------------------------
# = LAYER DETAILS
==========================================================================================
|        |   |    Total |     Horizontal     |      Vertical      |   Guide |  Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|----------------------------------------------------------------------------------------|
| Metal1 | H |    436.98|    0.00 |   434.02|    0.00 |     2.96|    0.00 |     0 |    0 |
| Metal2 | V | 182674.93|    0.00 |   228.71|    0.00 |182446.22|    0.00 | 20611 |    0 |
| Metal3 | H | 198775.35|    0.00 |198446.62|    0.00 |   328.73|    0.00 | 24585 |    0 |
|----------------------------------------------------------------------------------------|
| Totals |   |381887.26 |    0.00 |199109.35|    0.00 |182777.91|    0.00 | 45196 |    0 |
| Percent|   |   100.00 |    0.00 |   52.14 |    0.00 |   47.86 |    0.00 |       |      |
==========================================================================================
```

After the first global routing pass, all guide length has been replaced by global routes.

```
Begin 2nd global routing pass...
    End 2nd global routing pass 0.0s (kernel), 0.6s (user), 0.6s (elapsed), 140.3MB vm,
140.6MB peak vm
    Gcell summary
  Layer/type   #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  ---------------------------------------------------------------------------------------
  Metal1:cell    2209        0 ( 0.00%)   0   4   3   3  10  18  24  23  12   0    0    0
  Metal1:edge    2162        0 ( 0.00%)   0   8   7   4  10  12   9  10  16  20    0    0
  Metal2:cell    2209        0 ( 0.00%)   8   6  12  18  26  21   5   0   0   0    0    0
  Metal2:edge    2162        0 ( 0.00%)   7   8  12  20  26  19   4   0   0   0    0    0
  Metal2:down-via2209        0 ( 0.00%)
  Metal3:cell    2209        0 ( 0.00%)   7  10  14  16  17  14  13   4   0   0    0    0
  Metal3:edge    2162        0 ( 0.00%)   6   8  13  17  16  15  14   6   1   0    0    0
  Metal3:down-via2209        0 ( 0.00%)
  ---------------------------------------------------------------------------------------
# = LAYER DETAILS
==========================================================================================
|        |   |    Total |     Horizontal     |      Vertical      |   Guide |  Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|----------------------------------------------------------------------------------------|
| Metal1 | H |    450.02|    0.00 |   447.06|    0.00 |    2.96 |    0.00 |     0 |    0 |
| Metal2 | V | 183274.32|    0.00 |   226.70|    0.00 |183047.62|    0.00 | 20617 |    0 |
| Metal3 | H | 205556.42|    0.00 |205229.03|    0.00 |  327.39 |    0.00 | 24769 |    0 |
|----------------------------------------------------------------------------------------|
| Totals |   | 389280.76|    0.00 |205902.79|    0.00 |183377.97|    0.00 | 45386 |     0 |
| Percent|   |    100.00|    0.00 |    52.89|    0.00 |    47.11|    0.00 |       |       |
===========================================================================================
Begin 3rd global routing pass...
    End 3rd global routing pass 0.0s (kernel), 1.5s (user), 1.5s (elapsed), 140.4MB vm,
140.6MB peak vm
    Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  -----------------------------------------------------------------------------------------
  Metal1:cell    2209        0 ( 0.00%)   0    3   2   4  10  18  24  22  12   0    0    0
  Metal1:edge    2162        0 ( 0.00%)   0    6   7   5  10  12   9  10  16  20    0    0
  Metal2:cell    2209        0 ( 0.00%)   8    5  10  17  33  24   0   0   0   0    0    0
  Metal2:edge    2162        0 ( 0.00%)   6    7  11  19  34  18   1   0   0   0    0    0
  Metal2:down-via2209        0 ( 0.00%)
  Metal3:cell    2209        0 ( 0.00%)   6    9  12  15  17  19  17   1   0   0    0    0
  Metal3:edge    2162        0 ( 0.00%)   4    7  11  16  16  18  19   5   0   0    0    0
  Metal3:down-via2209        0 ( 0.00%)
  -----------------------------------------------------------------------------------------
# = LAYER DETAILS
===========================================================================================
|        |   |    Total |    Horizontal     |      Vertical      |   Guide |   Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|-----------------------------------------------------------------------------------------|
| Metal1 | H |    838.26|    0.00 |   834.99|    0.00 |     3.26 |    0.00 |     0 |     0 |
| Metal2 | V | 184276.54|    0.00 |   229.93|    0.00 |184046.61|    0.00 | 20636 |     0 |
| Metal3 | H | 213740.70|    0.00 |213419.88|    0.00 |  320.82 |    0.00 | 25010 |     0 |
|-----------------------------------------------------------------------------------------|
| Totals |   | 398855.50|    0.00 |214484.80|    0.00 |184370.70|    0.00 | 45646 |     0 |
| Percent|   |    100.00|    0.00 |   53.78 |    0.00 |   46.22 |    0.00 |       |       |
===========================================================================================
Begin 4th global routing pass...
    End 4th global routing pass 0.0s (kernel), 3.3s (user), 3.3s (elapsed), 140.6MB vm,
140.6MB peak vm
    Gcell summary
  Layer/type  #Gcell  #Overcon %Overcon <10% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% Blk%
  -----------------------------------------------------------------------------------------
  Metal1:cell     2209       0 ( 0.00%)   0    3   3   4  10  18  24  22  12   0    0    0
  Metal1:edge     2162       0 ( 0.00%)   0    6   7   5  10  12   9  10  16  20    0    0
  Metal2:cell     2209       0 ( 0.00%)   7    5   8  17  45  16   0   0   0   0    0    0
  Metal2:edge     2162       0 ( 0.00%)   5    5  10  21  43  12   0   0   0   0    0    0
  Metal2:down-via2209        0 ( 0.00%)
  Metal3:cell     2209       0 ( 0.00%)   5    6  10  15  20  23  17   1   0   0    0    0
  Metal3:edge     2162       0 ( 0.00%)   4    5   9  16  19  19  21   4   0   0    0    0
   Metal3:down-via2209       0 ( 0.00%)
  -----------------------------------------------------------------------------------------
# = LAYER DETAILS
===========================================================================================
|        |   |    Total |    Horizontal     |      Vertical      |   Guide |   Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|-----------------------------------------------------------------------------------------|
| Metal1 | H |    804.41|    0.00 |   798.69|    0.00 |     5.71|    0.00 |     0 |     0 |
| Metal2 | V | 183542.31|    0.00 |   253.77|    0.00 |183288.54|    0.00 | 20644 |     0 |
| Metal3 | H | 221058.46|    0.00 |220734.97|    0.00 |   323.49|    0.00 | 25214 |     0 |
|-----------------------------------------------------------------------------------------|
| Totals |   | 405405.17|    0.00 |221787.43|    0.00 |183617.74|    0.00 | 45858 |     0 |
| Percent|   |    100.00|    0.00 |    54.71|    0.00 |    45.29|    0.00 |       |       |
===========================================================================================
End global route 0.2s (kernel), 8.5s (user), 8.7s (elapsed), 140.6MB vm, 140.8MB peak vm
Begin writing global routes...
    Begin update routing congestion...
    End update routing congestion 0.0s (kernel), 0.3s (user), 0.3s (elapsed), 141.5MB vm,
142.2MB peak vm
# = LAYER DETAILS
===========================================================================================
|        |   |    Total |    Horizontal     |      Vertical      |   Guide |   Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|-----------------------------------------------------------------------------------------|
| Metal1 | H |    804.41|    0.00 |   798.69|    0.00 |     5.71|    0.00 |     0 |     0 |
| Metal2 | V | 182749.74|    0.00 |   252.46|    0.00 |182497.28|    0.00 | 20617 |     0 |
| Metal3 | H |  218981.9|    0.00 |218660.32|    0.00 |   321.60|    0.00 | 23876 |     0 |
|----------------------------------------------------------------------------------------|
| Totals |   | 402536.07|    0.00 |219711.48|    0.00 |182824.59|    0.00 | 44493 |     0 |
| Percent|   |    100.00|    0.00 |    54.58|    0.00 |   45.42 |    0.00 |       |       |
==========================================================================================
5578 nets written.
    22411 routes written.
End writing global routes 0.0s (kernel), 2.0s (user), 2.1s (elapsed), 141.5MB vm, 142.2MB
peak vm
Begin reclaim memory...
End reclaim memory 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 139.3MB vm, 142.2MB peak vm
0
```

Local route escapes pins and replaces any disconnects with guides between the ends of the new connections and the existing global routes to keep the connectivity legal.

```
local_route
Begin local route...
    Escaping 19872 instTerms and 0 terms
    Starting with 0 unroutes and 0 shapes in violation
    Escaped 19872, failed 0
    final results : unroutes     0  errors     0  violatees     0  weak     0  offgrid     0
End local route 0.0s (kernel), 14.6s (user), 14.7s (elapsed), 141.9MB vm, 142.2MB peak vm
Cpu time (local_route): 14.61sec
Elapsed time (local_route): 14.68sec
0
```

The `croute` command lays down as many wires as possible along routing conduits. Its goal is to keep the remaining missing connections short in length and to ensure that spacing violations that are created in this step can be corrected in the detail route step.

```
croute
max memory before CRoute: 142.2meg
total_shape_seen 121355; total_shape_used 30461; total_intra_used 5371; total_intra_pinned
0
cpu time (constructConduits): 0.44sec
max memory after data prep: 145.0meg
## Assigned in passes:
p0 = 8722
p1 = 16548
p2 = 0
p3 = 0
## Failure type 1: 0 2: 0, 3: 0
## Bend counts: total 3203 ; type2 (partial blockage) 555 ; type4 (cross blockage) 0
## Off Track: direct  3859 ; indirect 2896 ; diff Z direct  0 ; indirect 634 ;other pins: 6135
## Miss Lineup same Z: direct   1886 ; indirect  1707 ; diff Z direct 0 ; indirect 539
## Violation: Space 12 ; Net 0 ; LoBound 0 ; HiBound 0 ;Shape 0
## Tangle: Tangle 8 ; OutOfVCG 0
## spacing saved 38273 ; spacing called 71234
## Assigned in passes:
p0 = 0
p1 = 0
p2 = 4626
p3 = 1644
## Failure type 1: 0 2: 0, 3: 0
## Bend counts: total 2956 ; type2 (partial blockage) 670 ; type4 (cross blockage) 0
## Off Track: direct  3689 ; indirect 4160 ; diff Z direct  0 ; indirect 882 ;other pins: 6576
## Miss Lineup same Z: direct  2141 ; indirect  2836 ; diff Z direct 0 ; indirect 1222
## Violation: Space 38 ; Net 0 ; LoBound 0 ; HiBound 0 ;Shape 0
## Tangle: Tangle 86 ; OutOfVCG 0
## spacing saved 186604 ; spacing called 340808
length on layer  1:     12926360; via from layer: 5652
length on layer  2:    425017720; via from layer: 17788
length on layer  3:    458425547; via from layer: 0
length on layer  4:            0; via from layer: 0
length on layer  5:            0; via from layer: 0
length on layer  6:            0; via from layer: 0
Grand total length:    896369627; via: 23440; Embeded ratio:  0.68
cpu time (track assignment): 5.11sec
total cpu time (croute [trackAssignment+embed]): 7.23sec
max memory after embed: 148.8meg
memory added by CRoute:  6.6meg
### area push time 5.09sec ktime 0.00sec; attempted 3233, succeeded 2789
Max memory used after pushing: 148.8meg
### area pack time   4.73sec ktime   0.00sec; attempted 23940 Pack failed 10872
Memory usage after packing: 149.1meg
max memory after packing: 149.1meg
### critic time   2.90sec
max memory after critic: 149.1meg
# = LAYER DETAILS
====================================================================================
|        |   |    Total |     Horizontal     |      Vertical      |   Guide |  Down Vias  |
|  Layer |Dir|   Length |  Detail |  Global |  Detail |  Global |  Length | Total | Guide|
|----------------------------------------------------------------------------------|
| Metal1 | H |   2910.57|  421.90 |    0.00 |    0.00 |    0.00 |  2488.67|     0 |     0|
| Metal2 | V | 223552.07|  3267.85|    0.00 | 96234.38|    0.00 |124049.85| 20479 |   598|
| Metal3 | H | 189880.43|137930.25|    0.00 |  491.59 |    0.00 | 51458.59| 13153 | 10301|
|----------------------------------------------------------------------------------|
| Totals |   | 416343.07|141620.00|    0.00 | 96725.97|    0.00 |177997.11| 33632 | 10899|
| Percent|   |    100.00|  34.02  |    0.00 |    23.23|    0.00 |    42.75|       |      |
====================================================================================
Current memory after Croute =   145.4961meg (max memory =   149.1484meg)
total cpu time (croute): 20.09sec
```

Here, global routes have been replaced by detail routes and guides.

In the final step of the routing flow, Space-based Router and Chip Optimizer attempts to complete all connections according to the design rules. In this example, one cycle is run with multiple passes.

```
detail_route
Begin droute...
    Starting with 19487 guides
    Begin cycle 1...
[pass  1 index  1 begin] rips 22005  (unroutes 19487  errors  2518  violatees    0
weak     0  offgrid     0 giveUp     0)
[pass  1 end ] 68.30 secs, 68.30 secs total
[pass  2 index  2 begin] rips  5613  (unroutes  3106  errors  1655  violatees   852
weak     0  offgrid     0 giveUp     0)
[pass  2 end ] 71.69 secs, 139.99 secs total
[pass  3 index  3 begin] rips  9547  (unroutes    36  errors  3893  violatees  5618
weak     0  offgrid     0 giveUp     0)
[pass  3 end ] 125.80 secs, 265.79 secs total
[pass  4 index  4 begin] rips  7267  (unroutes     0  errors  4192  violatees  3075
weak     0  offgrid     0 giveUp     0)
[pass  4 end ] 117.64 secs, 383.43 secs total
[pass  5 index  5 begin] rips  2357  (unroutes     0  errors  1802  violatees   555
weak     0  offgrid     0 giveUp     0)
[pass  5 end ] 68.80 secs, 452.23 secs total
[pass  6 index  6 begin] rips  1712  (unroutes     0  errors   948  violatees   764
weak     0  offgrid     0 giveUp     0)
[pass  6 end ] 20.33 secs, 472.56 secs total
[pass  7 index  7 begin] rips   817  (unroutes     0  errors   578  violatees   239
weak     0  offgrid     0 giveUp     0)
```

```
[pass  7 end ] 37.49 secs, 510.05 secs total
[pass  8 index  8 begin] rips   591  (unroutes     0  errors   341  violatees   250
weak      0  offgrid     0 giveUp     0)
[pass  8 end ] 11.73 secs, 521.78 secs total
[pass  9 index  9 begin] rips   360  (unroutes     0  errors   263  violatees    97
weak      0  offgrid     0 giveUp     0)
[pass  9 end ] 7.23 secs, 529.01 secs total
[pass 10 index 10 begin] rips   307  (unroutes     0  errors   222  violatees    85
weak      0  offgrid     0 giveUp     0)
[pass 10 end ] 6.37 secs, 535.38 secs total
[pass 11 index  6 begin] rips   194  (unroutes     0  errors   190  violatees     4
weak      0  offgrid     0 giveUp     0)
[pass 11 end ] 6.28 secs, 541.66 secs total
[pass 12 index  7 begin] rips   236  (unroutes     0  errors   153  violatees    83
weak      0  offgrid     0 giveUp     0)
[pass 12 end ] 15.09 secs, 556.75 secs total
[pass 13 index  8 begin] rips   233  (unroutes     0  errors   122  violatees   111
weak      0  offgrid     0 giveUp     0)
[pass 13 end ] 23.61 secs, 580.36 secs total
[pass 14 index  9 begin] rips   136  (unroutes     0  errors    93  violatees    43
weak      0  offgrid     0 giveUp     0)
[pass 14 end ] 2.63 secs, 583.00 secs total
[pass 15 index 10 begin] rips    71  (unroutes     0  errors    49  violatees    22
weak      0  offgrid     0 giveUp     0)
[pass 15 end ] 2.46 secs, 585.46 secs total
[pass 16 index  6 begin] rips    31  (unroutes     0  errors    29  violatees     2
weak      0  offgrid     0 giveUp     0)
[pass 16 end ] 1.38 secs, 586.84 secs total
[pass 17 index  7 begin] rips    37  (unroutes     0  errors    28  violatees     9
weak      0  offgrid     0 giveUp     0)
[pass 17 end ] 4.23 secs, 591.07 secs total
[pass 18 index  8 begin] rips    52  (unroutes     0  errors    26  violatees    26
weak      0  offgrid     0 giveUp     0)
[pass 18 end ] 6.78 secs, 597.85 secs total
[pass 19 index  9 begin] rips    34  (unroutes     0  errors    22  violatees    12
weak      0  offgrid     0 giveUp     0)
[pass 19 end ] 1.52 secs, 599.37 secs total
[pass 20 index 10 begin] rips    32  (unroutes     0  errors    19  violatees    13
weak      0  offgrid     0 giveUp     0)
[pass 20 end ] 1.43 secs, 600.80 secs total
[pass 21 index  6 begin] rips    17  (unroutes     0  errors    17  violatees     0
weak      0  offgrid     0 giveUp     0)
[pass 21 end ] 1.19 secs, 602.00 secs total
[pass 22 index  7 begin] rips    28  (unroutes     0  errors    28  violatees     0
weak      0  offgrid     0 giveUp     0)
[pass 22 end ] 2.67 secs, 604.67 secs total
[pass 23 index  8 begin] rips    24  (unroutes     0  errors    14  violatees    10
weak      0  offgrid     0 giveUp     0)
[pass 23 end ] 4.95 secs, 609.62 secs total
[pass 24 index  9 begin] rips    17  (unroutes     0  errors    13  violatees     4
weak      0  offgrid     0 giveUp     0)
[pass 24 end ] 0.39 secs, 610.01 secs total
[pass 25 index 10 begin] rips     4  (unroutes     0  errors     4  violatees     0
weak      0  offgrid     0 giveUp     0)
[pass 25 end ] 0.02 secs, 610.03 secs total


Pass  1=    68.30s(atmp  20376/v  792/f 3437)Avg   0.00s Size  14.02 Rip  1.0 Push
0.00s Err 2518 Sn    0 Gd 19487 Gp    0 Wk    0 fb   0 fx   0
Pass  2=    71.69s(atmp   3726/v  228/f   66)Avg   0.02s Size  22.64 Rip  1.2 Push
0.00s Err 2507 Sn 2123 Gd 3106 Gp    0 Wk    0 fb   0 fx   0
```

```
Pass  3=  125.80s(atmp   3612/v  805/f   0)Avg  0.03s Size  27.19 Rip  4.1 Push
0.00s Err 9511 Sn  433 Gd   36 Gp    0 Wk    0 fb   0 fx   0
Pass  4=  117.64s(atmp   1994/v  613/f   0)Avg  0.06s Size  36.29 Rip  5.9 Push
6.09s Err 7267 Sn  283 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass  5=   68.80s(atmp    529/v  187/f   0)Avg  0.13s Size  44.52 Rip  8.4 Push
0.00s Err 2357 Sn  138 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass  6=   20.33s(atmp    573/v  218/f   0)Avg  0.04s Size  24.45 Rip  4.8 Push
2.26s Err 1712 Sn   37 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass  7=   37.49s(atmp    197/v   75/f   0)Avg  0.19s Size  53.59 Rip  9.3 Push
0.00s Err  817 Sn   36 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass  8=   11.73s(atmp    212/v   85/f   0)Avg  0.06s Size  33.63 Rip  5.4 Push
1.33s Err  591 Sn   12 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass  9=    7.23s(atmp    114/v   44/f   0)Avg  0.06s Size  24.35 Rip  6.3 Push
1.27s Err  360 Sn   13 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 10=    6.37s(atmp    101/v   39/f  81)Avg  0.06s Size  63.31 Rip  9.6 Push
0.00s Err  307 Sn   13 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 11=    6.28s(atmp     77/v   31/f   3)Avg  0.08s Size  33.17 Rip  5.4 Push
0.66s Err  194 Sn   10 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 12=   15.09s(atmp     69/v   14/f   0)Avg  0.22s Size  64.96 Rip  9.4 Push
0.00s Err  236 Sn   10 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 13=   23.61s(atmp    637/v  275/f  253)Avg  0.04s Size  54.58 Rip  9.5 Push
0.33s Err  233 Sn    7 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 14=    2.63s(atmp     32/v    9/f   0)Avg  0.08s Size  35.67 Rip  7.0 Push
0.18s Err  136 Sn   30 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 15=    2.46s(atmp     33/v   10/f  26)Avg  0.07s Size  77.91 Rip  6.2 Push
0.00s Err   71 Sn    7 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 16=    1.38s(atmp     13/v    3/f   0)Avg  0.11s Size  42.89 Rip  6.5 Push
0.09s Err   31 Sn    5 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 17=    4.23s(atmp     11/v    0/f   0)Avg  0.38s Size  74.46 Rip  8.8 Push
0.00s Err   37 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 18=    6.78s(atmp    158/v   55/f  89)Avg  0.04s Size  55.50 Rip 11.5 Push
0.11s Err   52 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 19=    1.52s(atmp      9/v    1/f   0)Avg  0.17s Size  44.39 Rip  8.7 Push
0.10s Err   34 Sn    5 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 20=    1.43s(atmp     16/v    5/f  15)Avg  0.09s Size  88.58 Rip  7.5 Push
0.00s Err   32 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 21=    1.19s(atmp      8/v    1/f   0)Avg  0.15s Size  54.53 Rip  6.0 Push
0.05s Err   17 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 22=    2.67s(atmp     10/v    3/f   0)Avg  0.27s Size  82.53 Rip 11.1 Push
0.00s Err   28 Sn    4 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 23=    4.95s(atmp     96/v   39/f  47)Avg  0.05s Size  55.86 Rip 11.6 Push
0.04s Err   24 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 24=    0.39s(atmp      4/v    0/f   0)Avg  0.10s Size  53.84 Rip  6.3 Push
0.01s Err   17 Sn    4 Gd    0 Gp    0 Wk    0 fb   0 fx   0
Pass 25=    0.02s(atmp      1/v    0/f   0)Avg  0.02s Size  91.47 Rip  4.0 Push
0.00s Err    4 Sn    2 Gd    0 Gp    0 Wk    0 fb   0 fx   0
       final results (cycle  1):  errors    2  unroutes    0  weak    0  offgrid
0 giveUp     0
        Begin Search and Repair...
        End Search and Repair 0.0s (kernel), 0.0s (user), 0.0s (elapsed), 148.9MB
vm, 167.2MB peak vm
    End cycle 1 7.8s (kernel), 10m6.3s (user), 10m17.7s (elapsed), 148.9MB vm,
167.2MB peak vm
End droute 7.8s (kernel), 10m6.3s (user), 10m17.8s (elapsed), 148.9MB vm, 167.2MB
peak vm
Ending guides 0
Cpu time (detail_route): 614.06sec
Elapsed time (detail_route): 617.78sec
0
```

When the routing script is finished, the results for the detail route step indicate whether the routing completed successfully. In this example, there are some errors and no remaining guides. Refer to the “Evaluating Router Results” on page 820 for tips on how to complete routing.

Evaluating Router Results

If the Space-based Router and Chip Optimizer router reports failures when it is finished, you must evaluate the results to determine what to do next.

Fixing Unroutes

If there are unroutes remaining, do the following:

1. On the Command line, type

   ```
   update_net_connectivity -all
   ```

   Guides are created for all unroutes.

2. In the Layer Object Display Panel, make all layers and objects *inactive* **except** for the *guides* in the Object section.

3. Choose *Edit—Select All* to select the guides only.

4. On the Command line, type

   ```
   p2p_route -set [get_selection_set]
   ```

   Space-based Router and Chip Optimizer completes routes where possible. For remaining unroutes, try using the Wire Editor.

Long guides can sometimes be routed by running global_route on the nets with the guides, followed by detail_route on those nets.

Fixing Violations

To fix any remaining DRC or connectivity violations,

1. Use fix_errors to repair minimum area, minimum edge length, minimum enclosed area, minimum width, minimum number of cuts, manufacturing grid, routing grid, port shorts, crossing, and minimum spacing violations.

2. Use adjust_vias to fix via wire extension, edge length, area and number of cuts violations.

3. Use fill_notch to repair same net violations that are notches.

   In some cases, filling a notch can trigger a new width-based spacing violation, so the results should be re-checked.

4. Use extend_wire_to_pin_edge to fully enclose pins with connecting wire segments.

Checking for Setup or Data Translation Issues

If a substantial portion of your design was not routed, use the following procedure to check for possible setup or data translation issues.

1. Load the placed design with no special options.
2. Run

   ```
   update_net_connectivity -all
   ```
3. Select some guides, then run

   ```
   p2p_route -set [get_selection_set]
   ```
4. Use the interactive wire editor to replace guides that could not be routed using `p2p_route`.

12

Optimize Commands

The Optimize commands help you to increase manufacturing reliability and yields.

The commands are presented in alphabetic order:

adjust_vias

```
adjust_vias
     [ -set d_setObj | -net s_netName | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | unfixed | moved | replaced] ]
     [ -annotation_limit i_limit ]
     [ -center_origin [ true | false ] ]
     [ -check_minedge [ true | false ] ]
     [ -check_mode {soft | hard}]
     [ -clear_annotations [ true | false ] ]
     [ -create_custom_via [ true | false ] ]
     [ -disable_checks {all | min_edge_length | same_net_crossings} ]
     [ -do_via_layers {s_layerName…} ]
     [ -edgelength [ true | false ] ]
     [ -exclude_nets {s_netName…} ]
     [ -exclude_set d_setObj ]
     [ -extension [ true | false ] ]
     [ -filter_minarea [ true | false ] ]
     [ -ignore_active_route_status [ true | false ] ]
     [ -lock_layers {s_layerName…} ]
     [ -maintain_stack [ true | false ]
     [ -maximize_numcuts [ true | false ] ]
     [ -minarea [ true | false ] ]
     [ -minspacing [ true | false ] ]
     [ -no_remaster [ true | false ] ]
     [ -no_repaint [ true | false ] ]
     [ -numcuts [ true | false ] ]
     [ -offset_via [ true | false ]
       [ -adjust_origin [ true | false ] ]
       [ -use_best_rule_spec [ true | false ] ] ]
     [ -protrusion_numcuts [ true | false ] ]
     [ -push [ true | false ] ]
     [ -push_vias [ true | false ] ]
     [ -swap_vias [ true | false ] ]
     [ -top_level_only [ true | false ] ]
     [ -use_best_cost [ true | false ] ]
     [ -use_grid {mfg | route} ]
```

Moves, rotates, or replaces vias to fix via wire extension, edge length, area, spacing, and numcut violations. You can also rotate vias to align edges. The scope can be limited to vias in a given set, a given net, or a given region and on specific cut layers. By default, all vias in the entire design are processed.

Arguments

`-adjust_origin [ true | false ]`

(Only applies with `-offset_via true`) Adds a new stdVia with an adjusted origin so that connecting wires do not need to be added to maintain centerline connectivity. By default and when this argument is set to `false`, connecting wires are added to the offset via if needed to maintain centerline connectivity.

`-annotate [ all | none | unfixed | moved | replaced ]`

Specifies which types of adjustments to annotate. The annotations are listed under *AdjustVia* in the Optimizations page of the Annotation Browser and are added to the `annotation:viaOpt` purpose of the via layer. By default, annotations are not added.

`all`	Adds annotations for all adjustment types.
`none`	Omits annotations.
`moved`	Adds annotations for vias that are moved.
`replaced`	Adds annotations for vias that are replaced.
`unfixed`	Adds annotations for vias that are not fixed.

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that can be added for a net on a cut layer. The default value is 1000.

`-center_origin`

Replaces vias with off-center origins with matching vias with centered origins. Vias are considered to have off-centered origins if their origin is not in the center of their bounding box nor the center of the cuts. Default is `false`.

`-check_minedge [ true | false ]`

When `true`, via adjustments that create minimum edge violations are not permitted. When `false`, via adjustments that create minimum edge violations are permitted.

Default: `true`

`-check_mode { soft | hard }`

Enables `hard` or `soft` constraint lookups.

Default: `hard`

`-clear_annotations`

Removes existing adjust vias annotations before adding new ones.

`-create_custom_via`

Specifies that if no available vias correct the violation, a stdVia should be created to fix the problem. In this case, symbolic routes to which stdVias were added will be converted to geometric routes.

Default: stdVias are not created.

`-disable_checks {all | min_edge_length | same_net_crossings}`

Suppresses the specified type of design rule checking. By default, all vias that are modified by this command will be DRC clean. Use of this option may result in design rule violations.

`all` No design rule checking is performed.

`min_edge_length`

Suppresses edge length checks.

`same_net_crossings`

Suppresses same net crossing checks.

`-do_via_layers {`*`s_layerName …`*`}`

Restricts processing to the specified cut layers. If not specified, all via layers are processed.

`-edgelength`

Indicates whether vias with edge length violations should be fixed.

Default: Adjustments are only attempted for vias with extension and/or numcut violations.

`-exclude_nets {`*`s_netNames…`*`}`

Specifies the names of nets to exclude from processing.

`-exclude_set` *`d_setObj`*

Excludes nets, routes and vias in the set from processing.

`-extension`

Indicates whether vias with extension violations should be fixed. By default, vias with extension violations are fixed.

`-filter_minarea [ true | false ]`

If `true`, potential `minArea` violations are ignored for vias that have a connecting guide on the same layer.

Default: `false`

`-ignore_active_route_status [ true | false ]`

When `true`, vias belonging to fixed routes are also processed.

Default: `(false`) Vias belonging to fixed routes are skipped.

`-lock_layers {`*`s_layerName…`*`}`

Prevents metal layers in the list from being changed.

`-maintain_stack` Specifies whether via stacks should be maintained and moved as a unit if any via in the stack is moved.

Default: (`false`) Vias are moved independently even when they are part of a stack.

`-maximize_numcuts` Specifies that if the best cost via is used, the vias should be costed considering the number of cuts. By default, the via with the best cost will be picked, but not necessarily the largest number of cuts.

`-minarea` Indicates whether vias with minimum area violations should be fixed.

Default: Adjustments are only attempted for vias with extension and/or numcut violations.

`-minspacing` Indicates whether minimum spacing violations should be fixed.

Default: Minimum spacing violations are ignored.

`-net` *`s_netName`* Limits processing to vias on the given net. By default, vias in the entire design are processed.

`-no_remaster` Specifies whether via remastering is permitted. If `true`, via remastering is permitted.

Default: (`false`) Vias can be changed to a different master occurrence.

`-no_repaint` Disables screen repaint after vias are adjusted. This argument is intended for use in scripts when the command is invoked multiple times in succession. Repainting the screen only after the last command is done can save time. By default the screen is repainted on completion of the command whenever vias have been adjusted.

`-numcuts` Indicates whether vias with numcut violations should be fixed. By default, vias with numcut violations are fixed.

`-offset_via [ true | false ]`

When `true`, attempts to move vias to align one edge of the via with each connecting segment without creating a DRC violation. If multiple movements are possible, the smallest movement is attempted first. When equal moves are possible, the direction is arbitrarily chosen.

Default: `false`

`-protrusion_numcuts [ true | false ]`

Specifies whether vias with protrusion numcut violations should be fixed. By default, vias with protrusion numcut violations are not fixed.

`-push` Permits pushing of route segments. By default, pushing of route segments is not permitted.

`-push_vias` Permits pushing of route vias. By default, pushing of other route vias is not permitted.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Limits processing to the given area (defined by the lower left and upper right coordinates). By default, vias in the entire design are processed.

`-set` *`d_setObj`* Limits processing to vias in the given set. By default, vias in the entire design are processed.

`-swap_vias` Overrides the `validRoutingVias` constraint and looks for the *from* via to swap with the *to* via.

`-top_level_only` Restricts processing to top-level shapes.

`-use_best_cost` Indicates that all vias should be evaluated and the via with the best cost should be used. By default, if the existing via can be moved to correct the violation, the via will be moved.

`-use_best_rule_spec [ true | false ]`

(Only applies with `-offset_via true`) Specify this argument when the vias being processed are at the intersection of two routes with different minWidth values in the respective rule specs so that the smaller of the two wire widths is used.

`[ -use_grid {mfg | route} ]`

Specifies the grid choices for via placement.

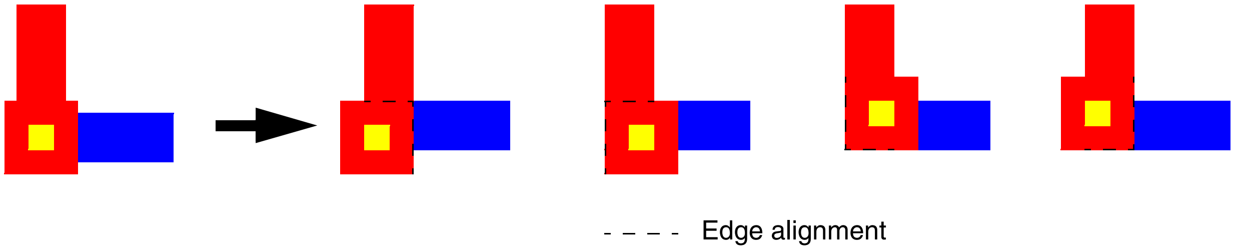
`mfg` (Default) Manufacturing grid is used.

`route` Routing grid is used.

Example

The following example aligns one via edge with each connecting metal layer segment.

```
adjust_vias -offset_via true -extension false -numcuts false
```



Before alignment, the via is centerline-connected with the connecting segments.

During alignment, the via is shifted to align edges with connecting metal layer segments. This example shows four possible solutions with alignment of two via edges when all displacements in all directions are equal. Any movements that cause a DRC violation are not permitted, and can result in one or no aligned edges for a given via.

Related Information

Tcl Command check_vias

check_litho_errors

```
check_litho_errors
     -layers {s_layerName…}
     { -region {f_xlo f_ylo f_xhi f_yhi} | -check_area [ true | false ] }
     [ -clear_annotations [ true | false ] ]
     [ -guidelines i_numHints ]
```

Runs Litho Physical Analyzer to check for lithography errors in the specific region or the areas specified by `Checked area` annotations that were created by get_changed_area and/or read_check_area. To use check_litho_errors, you must first enable Litho Physical Analyzer (initialize_lpa).

Litho Physical Analyzer generates markers for the lithography errors that it finds. These markers are added as annotations to the `annotation:violation` purpose for the layer and appear on the Violations page of the Annotation Browser under *Lithography Errors*, grouped by error type, layer, and severity.

Arguments

`-check_area [ true | false ]`

Limits checking to the *Checked area* annotations listed in the Optimizations page of the Annotation Browser that were created by `get_changed_area` and/or read_check_area.

`-clear_annotations [ true | false ]`

By default and when set to `true`, clears existing *Lithography Errors* annotations before checking.

`-guidelines` *`i_numHints`* Specifies the maximum number of hints to generate.

Default: 0

`-layers` **{***`s_layerName…`***}** Specifies the layers to check.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Limits checking to the specified region.

Value Returned

`i_count` Is the total number of errors found.

`-1`	The checking was not performed because Litho Physical Analyzer is not enabled or due to an error in syntax.

Example

The following command checks for lithography errors in current workspace on layer M2.

```
check_litho_errors -region [get_window_area] -layers M2
```

Related Information

Tcl Command	get_changed_area initialize_lpa read_check_area

clear_redundant_via_mapping

```
clear_redundant_via_mapping
```

Removes all existing redundant via mappings and enables the preferred via remastering method which uses `validRoutingVias` to specify the vias that can be used for remastering.

Arguments

None

Related Information

Tcl Command	map_redundant_via remaster_via show_redundant_via_mapping

connect_fill

```
connect_fill
     [ -nets {s_netName…} | -set d_setObj ]
     [ -layer {s_layerName …} ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -mode {mesh | tree} ]
     [ -keep_simple_connections [ true | false ] ]
     [ -lock_via_layers {s_layerName …} ]
     [ -threads i_threads ]
```

Connects fill shapes to the specified power and/or ground nets.

Arguments

`-all` Processes the entire design. This is the default.

`-keep_simple_connections [ true | false ]`

If `false`, which is the default, connections to fill shapes that do not participate in reducing the IR drop are excluded. This argument only applies to the mesh connection mode.

`-layer {s_layerName…}` Specifies the layers to process. By default, all routing layers are processed.

`-lock_via_layers {s_layerName…}`

Prevents this command from adding vias on the specified cut layers. By default, all via layers can be used.

`-mode {mesh | tree}` Chooses the connection method.

`mesh` (Default) Connects fill shapes to the specified power/ground nets using the maximum number of vias possible. Fill shapes can carry current as part of the power and ground structure. Increasing the number of cuts helps to reduce IR drop.

`tree` Connects fill shapes to the specified power/ground nets using the minimum number of vias possible given by the `minNumCut` rule.

`-nets` **{***`s_netName…`***}**

Connect added fill shapes to the given nets. The power/ground nets must be listed in the `SPECIALNETS` section of the DEF file. If several nets are given, connections to the nets are made in the order they are given in the list. If neither `-nets` nor `-set` is given, Space-based Router and Chip Optimizer attempts to connect to all power and ground nets.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary of the area to process, given the lower left and upper right coordinates.

`-set` *`d_setObj`*

Connects fill shapes to nets in the set. Specified power/ground nets must be listed in the `SPECIALNETS` section of the DEF file. There is no control on the net priorities to connect the fill shapes. If neither `-nets` nor `-set` is given, Space-based Router and Chip Optimizer attempts to connect to all power and ground nets.

`-threads` *`i_threads`*

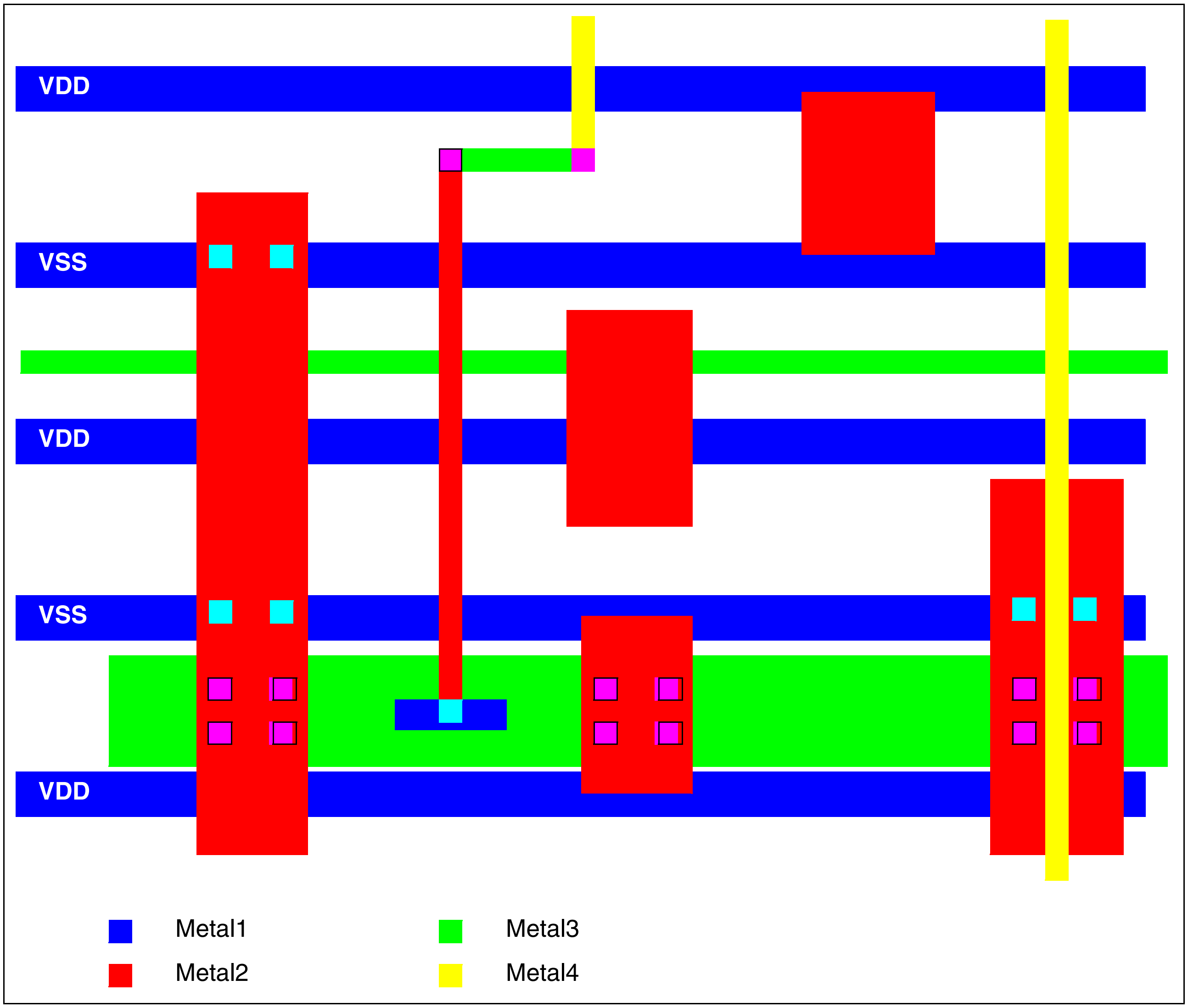
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

Examples

Mesh Connection Mode Without Keeping Simple Connections (Default)

The following figure shows vias added to connect fill shapes to power (VDD) and ground (VSS) nets in mesh mode using the following command:

```
connect_fill -mode mesh -nets {VSS VDD}
```

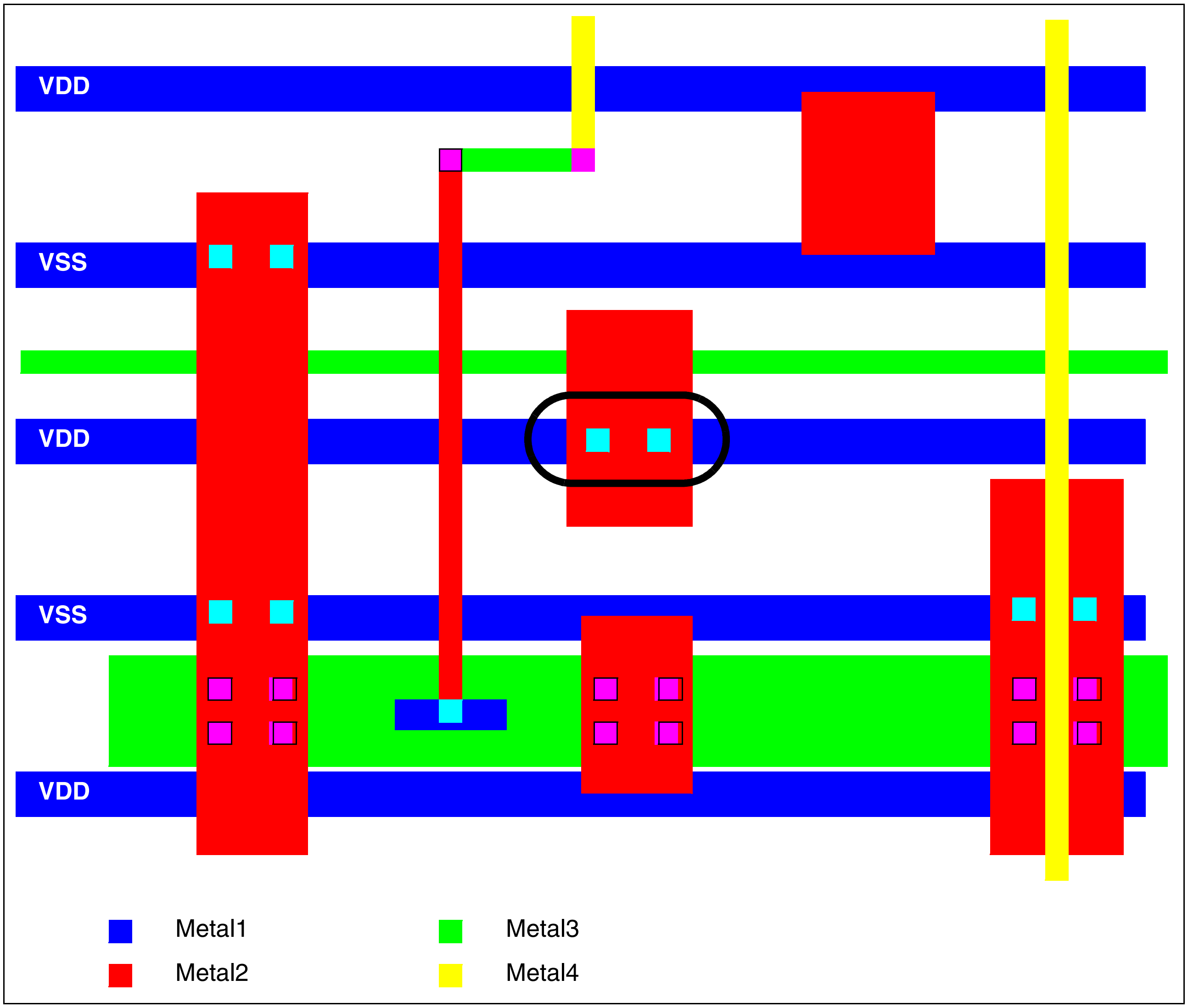


In the example, only connections which improve the IR drop are included.

Mesh Connection Mode Keeping Simple Connections

The following figure shows vias added to connect fill shapes to power (VDD) and ground (VSS) nets in mesh mode using the following command:

```
connect_fill -mode mesh -nets {VSS VDD} -keep_simple_connections
```

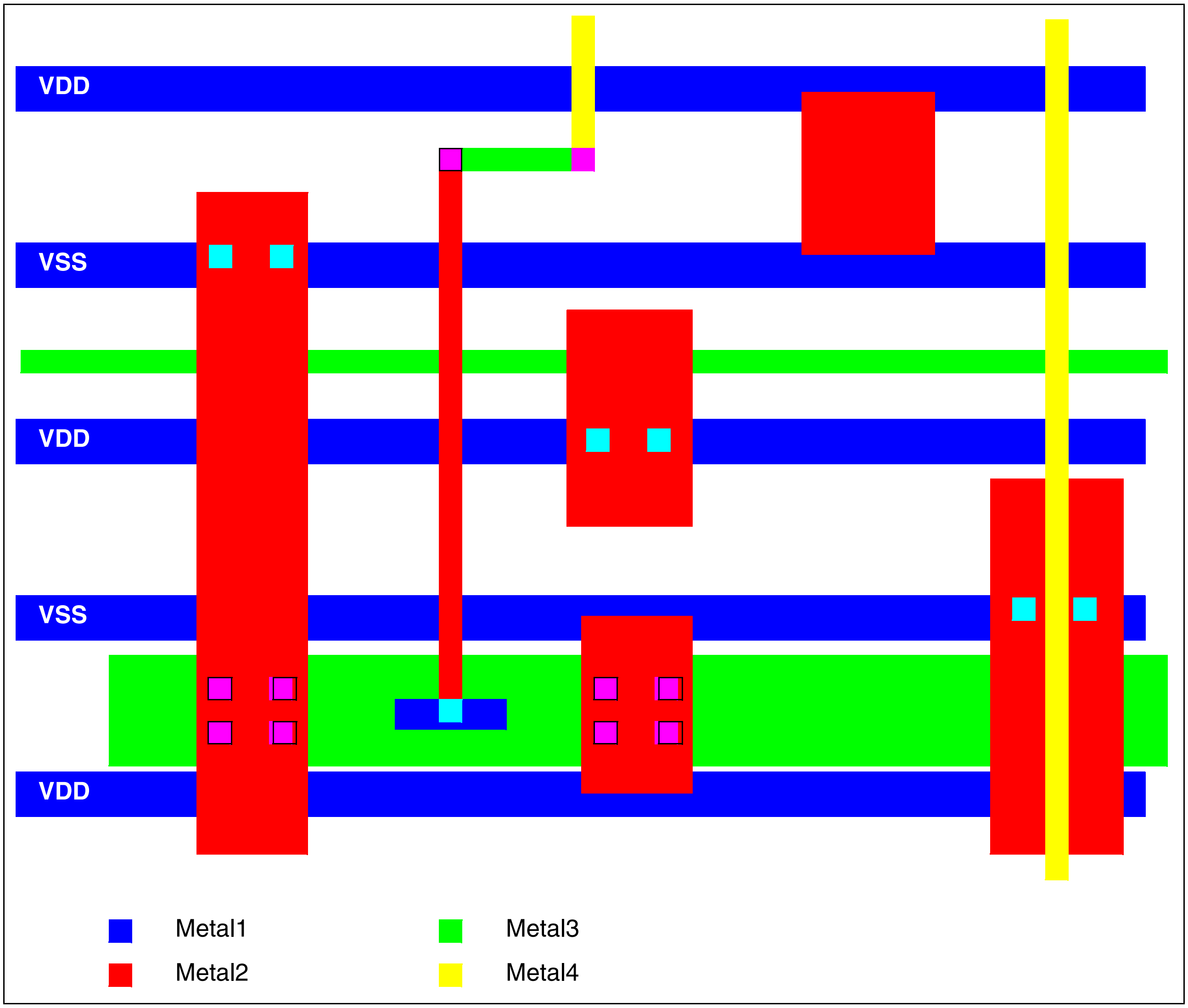


The connections in this example are the same as the previous mesh mode example, with the exception that the Metal2 shape in the center of the figure is connected to VDD. The vias for this shape represent a simple connection that does not affect the IR drop.

Tree Connection Mode

The following figure shows vias added to connect fill shapes to power (VDD) and ground (VSS) nets in tree mode using the following command:

```
connect_fill -mode tree -nets {VSS VDD}
```



While mesh mode tries to maximize connections, tree mode attempts to make connections with the minimum number of cuts required.

Related Information

Tcl Commands	create_fill delete_fill

create_density_fill_keepout

```
create_density_fill_keepout
     -name s_keepoutName
     -group s_groupName
     { -instance s_instanceName
     | -points {f_x1 f_y1 f_x2 f_y2 … f_xn f_yn f_x1 f_y1}
     | -region {f_xlo f_ylo f_xhi f_yhi} }
```

Creates a rectangular or polygonal fill keepout region. This prevents create_fill from adding fill to the given region on layers specified by the `densityFillKeepout` constraint.

To use this command, you must first set the fill keepout layers by doing the following:

1. Create a constraint group.

   ```
   create_constraint_group -name s_groupName
   ```

2. Set the `densityFillKeepout` layer constraint to `true` for each layer on which you want to keep out fill within a region.

   ```
   set_layer_constraint -constraint densityFillKeepout -layer s_layerName
   -BoolValue true -group s_groupName
   ```

You can now set the fill keepout region using `create_density_fill_keepout`. Once set, the fill keepout region boundary will be added as a `fill keepout` object in the artwork, with visibility control in the Object section of the Layer Object Display Panel under `area_boundary`.

You can create multiple constraint groups and use them with `create_density_fill_keepout` to manage different combinations of layers and regions.

After creating a fill keepout region with this command, you can re-enable fill in that region by doing one of the following:

➤ Unset the `densityFillKeepout` constraint for the layers on which you want to re-enable fill for the region.

   ```
   set_layer_constraint -constraint densityFillKeepout -layer s_layerName -group
   s_groupName
   ```

➤ Set the `densityFillKeepout` constraint to `false` for the layers on which you want to re-enable fill for the region.

   ```
   set_layer_constraint -constraint densityFillKeepout -layer s_layerName
   -BoolValue false -group s_groupName
   ```

Important

Changes to the `densityFillKeepout` constraint will affect *all* density fill keepout regions that were created using the constraint’s group.

Arguments

`-group` *`s_groupName`* Specifies the name of the existing constraint group that contains constraints for this keepout region.

`-instance` *`s_instanceName`*

Specifies the name of an instance whose rectangular boundary will be used for the fill keepout region.

`-name` *`s_keepoutName`*

Specifies the name of the fill keepout region.

`-points {`*`f_x1 f_y1 f_x2 f_y2 … f_xn f_yn f_x1 f_y1`*`}`

Specifies coordinate points as X and Y pairs for the polygonal region. The first and last points in the list must be identical.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates for the rectangular region.

Example

The following commands set up a fill keepout for metal layers Metal1, Metal2, and Metal3 in the rectangular region given by `{100 250 400 450}`.

```
create_constraint_group -name grpA
set_layer_constraint -constraint densityFillKeepout -layer Metal1 -BoolValue true
-group grpA
set_layer_constraint -constraint densityFillKeepout -layer Metal2 -BoolValue true
-group grpA
set_layer_constraint -constraint densityFillKeepout -layer Metal3 -BoolValue true
-group grpA
create_density_fill_keepout -name regionA -group grpA -region {100 250 400 450}
```

After these commands are run, create_fill will not be able to add fill shapes in the region `{100 250 400 450}` on layers `Metal1`, `Metal2`, and `Metal3`.

Related Information

Tcl Commands	create_constraint_group create_fill delete_fill set_layer_constraint

create_fill

```
create_fill
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -boundary {f_xlo f_ylo f_xhi f_yhi} ]
     [ -lpp {s_layerlpp …} ]
     [ -window_size f_userunit ]
     [ -step_size f_userunit ]
     [{[ -fill_width {f_min f_max} ]
       [ -fill_length {f_min f_max} ]
       [ -fill_step_size f_userunit ]}
      | -fill_dimensions {{f_width f_length}…} ]
     [ -fill_blockage_spacing f_userunit ]
     [ -fill_boundary_spacing f_userunit ]
     [ -fill_clock_spacing f_userunit ]
     [ -fill_direction {preferred | nonpreferred | any} ]
     [ -fill_fill_spacing f_userunit ]
     [ -fill_keepout_lpp {s_lpp…} ]
     [ -fill_lpp_spacing {{s_layerlpp f_userunit}…} ]
     [ -fill_minarea f_userunit ]
     [ -fill_pg_spacing f_userunit ]
     [ -fill_shape {square | rectangle | both} ]
     [ -fill_signal_spacing f_userunit ]
     [ -fill_style {dense | distributed | regular | greedy} ]
     [ -fill_type {floating | OPC} ]
     [ -fill_via_spacing f_userunit ]
     [ -fill_wire_spacing f_userunit ]
     [ -ignore_illegal_lpp [ true | false ] ]
     [ -max_density f_percent ]
     [ -max_diff_density f_percent_0_to_1 ]
     [ -min_density f_percent ]
     [ -density_range {f_minpercent f_maxpercent}]
     [ -staggered [ true | false ] ]
     [ -target_density f_percent ]
     [ -blockage_density f_percent ]
     [ -boundary_interior_halo {f_x f_y} ]
     [ -boundary_interior_halo_min_density_hole_mult f_multiplier ]
     [ -boundary_interior_halo_target_density f_percent ]
     [ -use_grid {mfg | route} ]
     [ -max_effort [ true |false ] ]
     [ -effort_level {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 0 | 10} ]
     [ -threads i_threads ]
     [ -verbose | -silent ]
```

Checks the metal density for layers in a given area and adds metal fill to achieve optimum metal density required by a specific manufacturing process.

Arguments

`-all` Operates on the entire design. This is the default area.

`-blockage_density` *`f_percent`*

Specifies the blockage percentage to consider when creating fill. By default, the target density is used. If a percentage of 0 is given, then blockages are treated as if there is no metal in the blockage area for density calculations, but the blockage area is still honored when placing fill shapes.

`-boundary` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies an alternate boundary for the design.
Default: the occurrence bounds.

`-boundary_interior_halo` **{***`f_x f_y`***}**

Specifies a band (given by x and y offsets) inside the boundary in which a modified target density can be specified using `-boundary_interior_halo_target_density`, and/ or a modified minimum hole area can be specified using `-boundary_interior_halo_min_density_hole_mult`.

`-boundary_interior_halo_min_density_hole_mult` *`f_multiplier`*

Multiplies the `minDensityHole` constraint by this value to determine the minimum hole area within the boundary halo given by `-boundary_interior_halo`. Value must be greater than 0 and less than 1. This lets you identify and fill holes that are smaller than `minDensityHole` within the boundary halo.

`-boundary_interior_halo_target_density` *`f_percent`*

Specifies the target density percentage (0-100) to use within the boundary halo given by `-boundary_interior_halo`.

`-density_range` **{***`f_minpercent f_maxpercent`***}**

Specifies the minimum and maximum density percentage to use as lower and upper limits. By default, `minDensity` and `maxDensity` constraint values are used, if defined.

`-effort_level` *`i_value`*

Specifies the amount of searching to do when looking for open areas in which to place fill shapes. The larger the number, the more searching is done, and the longer the runtime. Default: 1

`-fill_blockage_spacing` *`f_userunit`*

Specifies the minimum spacing between a fill shape and a blockage. The default value is the fill-signal spacing value.

`-fill_boundary_spacing` *`f_userunit`*

Specifies the minimum spacing between a fill shape and the design boundary. Defaults to the `minFillPatternSpacing` value, if defined, otherwise 0.

`-fill_clock_spacing` *`f_userunit`*

Specifies the minimum spacing between a fill shape and a clock route. The default value is the fill-signal spacing value.

`-fill_dimensions` **{**{*`f_width f_length`*}**…}**

Specifies a list of fill shape width/length pairs to use for fill shape dimensions. Only fill shapes with dimensions in the list will be created.

`-fill_direction {preferred | nonpreferred | any}`

Specifies how fill shapes should be oriented relative to the layer’s preferred routing direction.

`preferred` (Default) Fill is aligned with the layer’s preferred routing direction.

`nonpreferred` Fill is aligned perpendicular to the layer’s preferred routing direction.

`any` Fill can be aligned in either direction.

`-fill_fill_spacing` *`f_userunit`*

Specifies the minimum spacing between two fill shapes. The default is taken from the technology file (`minSameNetSpacing` or `minSpacing`, in that order)

`-fill_keepout_lpp` **{***`s_lpp`***…}**

Prevents fill from being added that would touch any shape on the given layer purpose pairs.

`-fill_length` **{***f_min f_max***}**

Specifies the minimum and maximum fill length. The default is the `minWidth` for the given layer, with no maximum length unless specified in the technology file.

`-fill_lpp_spacing` **{**{*s_layerlpp f_userunit*}…**}**

Specifies the minimum spacing between fill shapes and shapes on the given layers, layer purpose pairs, or purposes. The format of the argument values is:

{*s_layerlpp1 f_spacing1 s_layerlpp2 f_spacing2…*}

`-fill_minarea` *f_userunit*

Specifies the minimum area for a fill shape. The default value is 0, unless specifies in the technology file (`minArea`).

`-fill_pg_spacing` *f_userunit*

Specifies the minimum spacing between a fill shape and a power or ground route. The default value is the fill-signal spacing value.

`-fill_shape {square | rectangle | both}`

Restricts the type of fill shapes created.

`both`	(Default) Fill shape can be square or rectangular.
`rectangle`	Only fill having unequal width and length (non-square) will be created.
`square`	Only fill with equal width and length will be created.

`-fill_signal_spacing` *f_userunit*

Specifies the spacing between a fill shape and a signal route. The default is taken from the technology file (`minFillPatternSpacing` or `minSpacing`, in that order).

`-fill_step_size` *f_userunit*

Specifies the increment to use between the minimum and maximum values for width and length when building fill shapes. Defaults to 1.

`-fill_style {dense | distributed | regular | greedy}`

Controls how fill shapes are placed in a given window.

`dense` Places fill as tightly as possible in each check window. Results will tend to have clusters of fill shapes at regular intervals. This typically runs the fastest of all fill styles.

`distributed` (Default) Places fill more randomly throughout each check window.

`greedy` Places fill as tightly as possible in each check window. Results will tend to have clustering in the lower-left corner of each check window.

`regular` Places fill more uniformly throughout each check window.

`-fill_type {floating | OPC}`

Specifies the purpose that new fill is added to as `fill` (`floating`) or `opcFill` (`OPC`). Default is `floating`.

`-fill_via_spacing` *`f_userunit`*

Specifies the minimum spacing between a fill shape and a via shape on the `via:detail` or `via:redundant` purpose. The default is the fill-signal spacing value.

`-fill_width` **{***`f_min f_max`***}**

Specifies the minimum and maximum fill width. The defaults are the `minWidth` taken from the technology file, with no maximum width unless given in the technology file. Candidate fill shapes will be attempted starting with the width/length that results in the largest area, down to the smallest, subject to minimum area restrictions.

`-fill_wire_spacing` *`f_userunit`*

Specifies the minimum spacing between a fill shape and a wire on the `wire:detail` or `wire:redundant` purpose. The default is the fill-signal spacing value.

`-ignore_illegal_lpp [ true | false ]`

When `true`, any non-existent layers or layer purpose pairs that appear in the command line, other than values for the `-lpp` argument, are treated as non-fatal errors and an error message is output.

Default: (`false`) Non-existent layers and/or layer purpose pairs that appear in the command line cause the command to fail and exit.

`-lpp` **{***`s_layerlpp…`***}**

Specifies the layers and/or layer purpose pairs to process. By default, all routing layers are processed.

`-max_density` *`f_percent`*

Specifies the maximum density allowed. Defaults to the `maxDensity` constraint value, if defined, or 100%. This establishes a hard limit for the amount of fill added. Normally, an extra fill shape is added to each check window area to force the density to meet or exceed the target density. If the target density is equal to the maximum density, the extra shape is not added and maximum density will not be exceeded.

`-max_diff_density` *`f_percent_0_to_1`*

Specifies the maximum density difference allowed between adjacent, non-overlapping check windows. Defaults to the `maxDiffDensity` constraint value, if defined. Must be between 0 and 1.

`-max_effort {true|false ]`

When true, any windows, in which the minimum density (or target density, if minimum density is not given) cannot be achieved, are processed a second time with the highest effort level possible (every grid location is checked). Defaults to `false`.

`-min_density` *`f_percent`*

Specifies the minimum density allowed. Defaults to the `minDensity` constraint value, if defined, or the target density. This establishes a lower limit for the amount of fill added. If the target density cannot be met in a window, and minimum density was not met, a second attempt will be made to meet the minimum if the `-max_effort` argument is specified.

`-region {f_xlo f_ylo f_xhi f_yhi}`

Specifies the boundary of the area to process, given the lower left and upper right coordinates.

`-silent` Outputs only essential messages.

`-staggered [ true | false ]`

When `true`, a staggered fill pattern is used, where each successive row or column is offset from the previous. Refer to Figure 12-1 on page 849 for a comparison of this fill pattern and the default.

Default: (`false`) Fill shapes are added in a more uniform pattern.

`-step_size {f_userunit [f_userunit]}`

Specifies the window step as a single value for X and Y, or as two values ({X Y}). If the value given is larger than the current design, then the window step will automatically be reduced to the bounds of the region/design.

Default: One-half of the window size (`-window_size`) is used.

`-target_density f_percent`

Specifies the desired density. Fill shapes will be added to reach a density equal to or greater than the target density, or the maximum density possible if the target density cannot be met. The default is the minimum density from the technology file.

Valid values are 1.0 through 100.0

`-threads i_threads` Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

`-use_grid {mfg | route}`

Specifies the grid to use for shape placement from the following choices:

`mfg`	Selects the manufacturing grid. This is the default.
`route`	Selects the routing grid.

`-verbose` Outputs additional information while processing.

`-window_size {f_userunit [f_userunit]}`

Specifies the window size as a single value for X and Y, representing a square window, or as two values ({X Y}). If the value given is larger than the current region or design, then the window will automatically be reduced to the bounds of the region/design.

Default: The window size given by the `minDensity` or `maxDensity` constraint is used.

Example

The following example checks the metal density for the metal1 layer using a window size of 100 um, a step size of 50 um, and attempts to add fill shapes in window areas with a metal density less than 50%. Only fill shapes with dimension 4x4 are created.

```
create_fill -lpp metal1 -fill_width {4 4} -fill_length {4 4} -target_density 50
-window_size 100
```

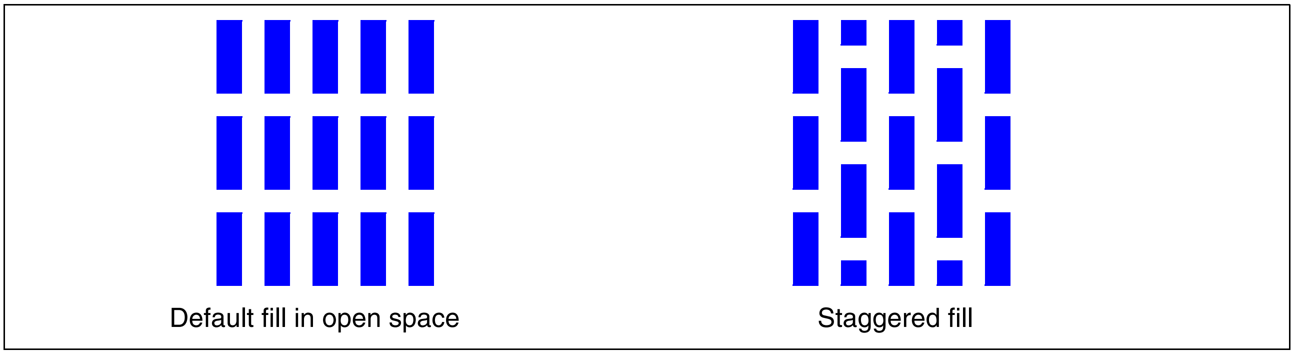
The following example adds fill shapes of dimensions 1x2, 3x4 and 0.5x1.5.

```
create_fill -lpp metal1 -fill_dimensions {1 2 3 4 0.5 1.5} -target_density 50
-window_size 100
```

The following example adds fill shapes of dimensions 1x1, 1.5x1, 2x1. 1x1.5, 1.5x1.5, and 2x1.5.

```
create_fill -lpp metal1 -fill_width {1 2} -fill_length {1 1.5} -fill_step_size 0.5
-target_density 50 -window_size 100
```

Figure 12-1 Normal (default) Fill Pattern versus Staggered (-staggered)



Related Information

Tcl Command	check_density create_net_fill delete_fill

create_net_fill

```
create_net_fill
     -nets {s_netName …}
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -layer {all | {s_layerName …}} ]
     [ -fill_width f_userunit ]
     [ -fill_length f_userunit ]
     [ -fill_fill_clearance f_userunit ]
     [ -fill_signal_clearance f_userunit ]
     [ -fill_pg_clearance f_userunit ]
     [ -fill_clock_clearance f_userunit ]
     [ -use_grid {mfg | route} ]
     [ -no_repaint ]
```

Creates fill shapes around a net.

Arguments

`-fill_clock_clearance` *`f_userunit`*

Specifies the clearance between a fill shape and a clock route. The default value is the fill-signal clearance value.

`-fill_fill_clearance` *`f_userunit`*

Specifies the clearance between two fill shapes. The default is taken from the technology file (`minFillPatternSpacing`, `minSameNetSpacing`, or `minSpacing`, in that order)

`-fill_length` *`f_userunit`*

Specifies the length for fill shapes. If not specified, the minimum length (`minEdgeLength`) for the given layer is used.

`-fill_pg_clearance` *`f_userunit`*

Specifies the clearance between a fill shape and a power or ground route. The default value is the fill-signal clearance value.

`-fill_signal_clearance` *`f_userunit`*

Specifies the clearance between a fill shape and a signal route. The default is taken from the technology file (`minSpacing` or `minClearance`, in that order).

`-fill_width` *`f_userunit`*

Specifies the width for fill shapes. If not specified, the minimum width (`minWidth`) for the given layer is used.

`-layer {all |` **{***`s_layerName …`***}**`}`

Restricts processing to the specified layers. By default or if `all` is specified, all routing layers are processed.

`-nets` **{***`s_netName…`***}**

Specifies the nets to process. Fill shapes are created around the given nets on the shape layer, unless further restricted by the `-layer` argument. The minimum clearance is maintained between the fill shapes and the net, unless overridden by the `-fill_signal_clearance` argument.

`-no_repaint`

Prevents the window from being updated after fill shapes are added. Use this argument in scripts when this command is sequentially repeated to eliminate refreshing and speed up processing until the last command is completed. By default, the window is always refreshed when fill is added.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary of the area to process, given the lower left and upper right coordinates.

`-use_grid {mfg | route}`

Specifies the grid to use for shape placement from the following choices:

`mfg`	Selects the manufacturing grid. This is the default.
`route`	Selects the routing grid.

Example

The following example adds `metal1` fill shapes (length of 0.6 and default width) along the `A_32_INST/i_918` net.

```
create_net_fill -layer metal1 -nets A_32_INST/i_918 -fill_length .6
```

Related Information

Tcl Commands	create_fill delete_fill
Menu Command	*Optimize—Metal Density*

create_pg_fill

```
create_pg_fill
     [ -set d_setObj | -nets {s_netName…} ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -layer {s_layerName…} ]
     [ -lock_via_layers {s_layerName …} ]
     [ -connections [ true | false ] ]
     [ -connections_type [all | via | same_layer] ]
     [ -keep_simple_connections [ true | false ] ]
     [ -silent | -verbose ]
     [ -threads i_threads ]
```

Creates fill shapes connected to power and ground nets to improve IR drop while also increasing metal density. This command performs the following:

- Inserts the maximum number of stripes as possible in the preferred routing direction on each given layer, respecting all the existing power/ground spacing and width constraints:
  - `pgFillWidth` is the width of the power/ground fill stripes to insert. A small width is recommended to minimize the impact on the timing. If `pgFillWidth` is not defined, two times `minWidth` is used.
  - `minPgFillSpacing` is the minimum spacing between two power/ground fill stripes. If `minPgFillSpacing` is not defined, the first defined value is used from the following: `minSameNetSpacing*2` or `minSpacing*2`.
  - `minPgFillSignalSpacing` is the minimum spacing between power/ground fill stripes and signal nets. If `minPgFillSignalSpacing` is not defined, two times `minSpacing` is used.
  - `minPgFillClockSpacing` is the minimum spacing between power/ground fill stripes and clock nets. If `minPgFillClockSpacing` is not defined, `minPgFillSignalSpacing` is used.
  - `minPgFillPgSpacing` is the minimum spacing between power/ground fill stripes and existing power/ground nets. If `minPgFillPgSpacing` is not defined, `minPGFillSignalSpacing` is used.
  - `minPgFillFloatingFillSpacing` is the minimum spacing between power/ ground fill stripes and existing floating fill shapes. If `minPgFillFloatingFillSpacing` is not set, then `minPgFillSpacing` is used.
  - `minPgFillBoundarySpacing` is the minimum spacing between power/ground fill stripes and the design boundary. If `minPgFillBoundarySpacing` is not defined, then `minBoundaryInteriorHalo` is used, and if neither is defined, then the default is 0.

- Connects all stripes to the given power/ground nets (`-nets` or `-set`).
- Removes all fill shapes with less than two connections because these shapes will not help to improve the IR drop (`-connections true -keep_simple_connections false`).

The `create_pg_fill` command does not consider metal density requirements. Following this command, run check_density to verify the metal density, and create_fill to add additional fill shapes as needed.

Arguments

`-all` Processes the entire design. This is the default.

`-connections [ true | false ]`

(Default) If `true`, connects the created fill stripes to the nets specified by `-nets` or `-set`, and all remaining floating fill stripes are removed.

If `false`, creates as many floating fill stripes as possible.

`-connections_type [all | via | same_layer]`

Specifies how power/ground fill shapes can be connected to power/ground rails.

`all`	Both via and same-layer connections are used. This is the default.
`same_layer`	Connects fill stripes directly to same-layer power rails.
`via`	Uses vias to connect fill stripes to power rails.

`-keep_simple_connections [ true | false ] ]`

(Applies only with `-connections true`) If `false`, removes fill stripes that do not help to improve IR drop (stripes that are floating/unconnected or with fewer than two connections).

Default: `false`

`-layer` **{***`s_layerName…`***}**

Specifies the layers to process. By default, all routing layers are processed.

`-lock_via_layers {`*`s_layerName…`*`}`

Prevents this command from adding vias on the specified cut layers. By default, all via layers can be used.

`-nets {`*`s_netName…`*`}`
Connect added fill stripes to the given nets. Specified power/ground nets must be listed in the `SPECIALNETS` section of the DEF file. If several nets are given, Space-based Router and Chip Optimizer attempts to connect to the nets in the order they are given in the list. If neither `-nets` nor -`set` is given, Space-based Router and Chip Optimizer attempts to connect to all power and ground nets.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Specifies the boundary of the area to process, given the lower-left and upper-right coordinates.

`-set` *`d_setObj`*
Connects fill stripes to nets in the set. Specified power/ground nets must be listed in the `SPECIALNETS` section of the DEF file. There is no control on the net priorities to connect the stripes. If neither `-nets` nor -`set` is given, Space-based Router and Chip Optimizer attempts to connect to all power and ground nets.

`-silent`
Outputs only essential messages.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

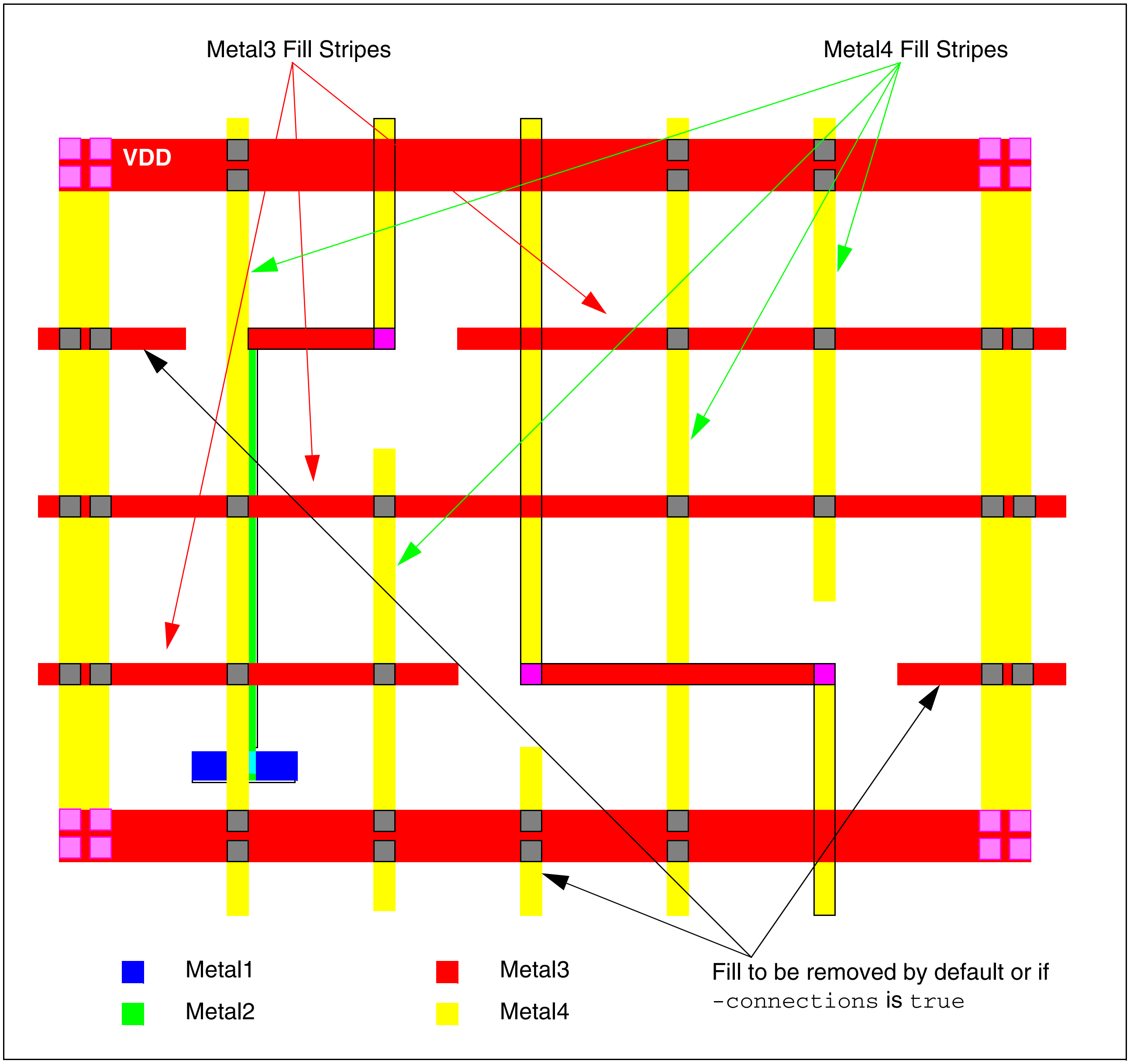
`-verbose`
Outputs additional messages.

Examples

The following commands set power/ground fill constraints for layer `M3`.

```
set_layer_constraint -constraint pgFillWidth -layer M3 -Value 0.1
set_layer_constraint -constraint minPgFillSpacing -layer M3 -Value 0.2
set_layer_constraint -constraint minPgFillSignalSpacing -layer M3 -Value 0.3
set_layer_constraint -constraint minPgFillClockSpacing -layer M3 -Value 0.6
set_layer_constraint -constraint minPgFillPgSpacing -layer M3 -Value 0.4
set_layer_constraint -constraint minPgFillBoundarySpacing -layer M3 -Value 0.5
```

The following figure shows fill stripes added to an area and the power-to-fill and fill-to-fill connections. Some fill stripes are interrupted by existing signal wires that are outlined in black. By default and when `-connections` is set `true`, the added fill stripes with only one connection to the power net will be removed as indicated in the figure. This example uses via connections only (`-connections_type via`).



Related Information

Tcl Commands	create_fill delete_fill

create_step_via

```
create_step_via
     -net s_netName | -set d_setObj
     -layer s_layerName
     -maxStep f_maxStep
     [-viaDef s_viaDefName ] | {-viaDefHorz s_viaDefH -viaDefVert s_viaDefV}]
     [-numCuts i_numCuts]
     [-exceptionPercent f_percent -exceptionWidth f_width]
     [-viaEnclosed [true | false]]
     [-viaRotate [true | false]]
     [-report [true | false]]
```

Inserts vias on paths of the given net or nets in the given set at intervals that are less than or equal to the given maxStep distance. Optional arguments specify the vias to be used, the number of cuts, whether the via must be fully enclosed, whether the via can be rotated, the exceptions, and whether a detailed report should be output.

Arguments

`-exceptionPercent` *`f_percent`* `-exceptionWidth` *`f_width`*

Specifies that when the distance between vias is greater than or equal to ((*`f_percent`*/100) * *`f_maxStep`*) and the width of the path is greater than or equal to *`f_width`*, no additional step via is needed.

`-layer` *`s_layerName`*

Processes paths only on the given layer.

`-maxStep` *`f_masStep`*

Specifies the maximum distance, in user units, between vias measured center-to-center.

`-net` *`s_netName`*

Processes paths on the given net.

`-numCuts` *`i_numCuts`*

Specifies the number of cuts in the via to be inserted.

Default: 1

`-report [true | false]`

When set to `true`, outputs the detailed via insertion information for each processed net. When set to `false`, outputs only the total number of vias generated for each processed net.

Default: `false`

`-set` *`d_setID`*

Processes paths on the nets in the given set.

`-viaDef` *`s_viaDefName`*

Name of the via to be inserted.

Default: When no via definition is specified, standard vias from validRoutingVias with the path layer as the top via layer are considered.

`-viaDefHorz` *`s_viaDefH`*

Name of the via to be inserted for horizontal path segments.

`-viaDefVert` *`s_viaDefV`*

Name of the via to be inserted for vertical path segments.

`-viaEnclosed [true | false]`

When set to `true`, inserts vias only when they are fully enclosed by the existing path shape.

Default. `false`

`-viaRotate [true | false]`

Specifies whether vias can be rotated.

Default: `true`

Example

Inserts double-cut M2_M1 vias when the vias on layer `M2` of net `AA` are greater than 400 microns apart. The inserted vias must be fully enclosed by the `M2` path and cannot be rotated. If there are vias that are less than 400 microns apart but greater than or equal to 160 microns (`40/100*400`) apart on `M2` path segments with width greater than or equal to 25 microns, then no step vias need to be added between those vias.

```
create_step_via -layer M2 -net AA -maxStep 400 -viaDef M2_M1 -numCuts 2
-exceptionPercent 40 -exceptionWidth 25 -viaEnclosed true -viaRotate false
```

critic

```
critic
     [ -layers {s_layerName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -append [ true | false ] ]
     [ -length_threshold i_tracks ]
     [ -max_jog_length f_length ]
     [ -output_file_name s_fileName ]
```

Smoothes wires by removing unnecessary jogs in the entire top cellview, an area, or a set.

Arguments

`-append [ true | false ]` Appends output to the file given by `-output_file_name`.

`-exclude_net {`*`s_netName…`*`}`

Excludes the given nets from processing.

`-exclude_set` *`d_setObj`* Excludes the given set from processing.

`-exclude_type {[power][ground][clock]}`

Excludes one or more given types of nets from processing.

`-layers {`*`s_layerName…`*`}` Limits changes to the specified layers.

Default: All layers can be changed.

`-length_threshold` *`i_tracks`*

Prevents the algorithm from creating segments longer than this value (in tracks) that could otherwise result in timing degradation. Longer wires increase the possibility of lateral capacitance effects that can hurt timing. By default, there is no length restriction.

`-max_jog_length` *`f_length`*

Maximum length of a wire jog, measured center point to center point, that will be straightened.

Default: `maxJogLength` constraint, if set; otherwise, all lengths are considered

`-output_file_name` *`s_fileName`*

Outputs to the given file the net names and coordinates of route sections that were moved.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Processes routes in the area given by the lower left (*`f_xlo f_ylo`*) and upper right (*`f_xhi f_yhi`*) coordinates. By default, the entire top cellview is processed.

`-set` *`d_setObj`* Processes routes in the specified set.

Example

The following example smoothes routes in the selected set.

```
critic -set [get_selection_set]
```

fix_antenna

```
fix_antenna
     -all | -net netName | -set d_setObj
     [ -exclude_nets {s_netName …} ]
     [ -annotate_jumpers [ true | false ] ]
     [ -annotate_diodes [ true | false ] ]
     [ -clear_annotations ]
     [ -check {[PAR] [CAR]} ]
     [ -diff_use_only [ true | false ] ]
     [ -fillcell_distance f_micron ]
     [ -layer_list {s_layerName…} | -layer_range {s_layerName…} ]
     [ -max_wires_to_push [ true | false ] ]
     [ -model {[OXIDE1] [OXIDE2] [OXIDE3] [OXIDE4]} ]
     [ -noIOPinDefault [ true | false ] ]
     [ -push [ true | false ] ]
     [ -push_vias [ true | false ] ]
     [ -use_jumpers [ true | false ] ]
     [ -use_diodes [ true | false ] ]
     [ -use_factors [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -via_rotation [ true | false ] ]
```

Fixes process antenna violations for the entire design, a specific net, or nets in a set by inserting jumpers and/or diodes. You can choose to ignore specific nets. Results are output to the Transcript area.

> *Important*
>
> To use this command, your design data must have been imported from LEF and DEF files that include process antenna keywords for setting values used by this command.

Arguments

`-all` Performs process antenna checks on all nets.

`-annotate_diodes [ true | false ]`

Creates annotations for added diodes. Defaults to `true`.

The annotations are added to the `annotation:viaOpt` purpose for the metal layer and are listed by net as *Antenna Diodes* in the Optimizations page of the Annotation Browser.

`-annotate_jumpers [ true | false ]`

Creates annotations for added jumpers. Defaults to `true`.

The annotations are added to the `annotation:viaOpt` purpose for the metal layer and are listed by net as *Antenna Jumpers* in the Optimizations page of the Annotation Browser.

`-check {[PAR][CAR]}` Determines which checks to perform: `PAR` (Partial Area Ratio), `CAR` (Cumulative Area Ratio).

Default: `PAR` and `CAR`, whichever ratios are defined in the LEF file

`-clear_annotations` Removes existing violation annotations of the same check type before performing this check. If not specified, existing violations are not cleared.

`-diff_use_only [ true | false ]`

When set to `true`, applies area factor to layers when connected to diffusion. If no layers are specified, then area factor is applied to all layers when connected to diffusion.

Default: `false`

`-exclude_nets {`*`s_netName`*` …}`

Ignores the given nets from the check. This argument is ignored if -`net` is specified.

`-fillcell_distance `*`f_micron`*

Specifies the distance in microns from the net to search for filler cells to use for diode insertion locations.

`-layer_list {`*`s_layerName…`*`}`

Specifies the layers that jumpers can be added to. Only `validRoutingLayers` that are also in this list can be used. By default, jumpers can only be added on `validRoutingLayers`.

`-layer_range {`*`s_layerName s_layerName`*`}`

Specifies the range of layers that jumpers can be added to. Only `validRoutingLayers` that are also in this range can be used. By default, jumpers can only be added on `validRoutingLayers`.

`-max_wires_to_push `*`i_count`*

Specifies the number of wires that can be pushed for a single jumper. Default is 2.

`-model`
Specifies which oxide models to use: `OXIDE1`, `OXIDE2`, `OXIDE3`, `OXIDE4`.

Default: `OXIDE1`

`-net` *`s_netName`*
Performs process antenna checks on the given net.

`-noIOPinDefault [ true | false ]`

Prevents the antenna checker from using IO pin default values:

- `ANTENNAINPUTGATEAREA`
- `ANTENNAOUTPUTDIFFAREA`
- `ANTENNAINOUTDIFFAREA`

By default, these values are used for any input that does not have a gate area or any output that does not have diode area.

`-push [ true | false ]`
Permits pushing of neighboring geometry to provide room for jumpers. Default is `true`.

`-push_vias [ true | false ]`

Permits pushing of vias. Default is `true`.

`-set` *`d_setObj`*
Performs process antenna checks on nets in the given set.

`-silent [ true | false ]`

Prevents messages from being output.

Default: `false`

`-use_diodes [ true | false ]`

Adds diodes to remove violations. Default is `true`

`-use_factors`
Uses area factors when computing PAR values.

`-use_jumpers [ true | false ]`

Adds jumpers to remove violations. Default is `true`

`-via_rotation`
Permits jumpers to use rotated vias. Default is `false`.

Value Returned

i_jumperscount *i_diodecount*	Tcl list with the number of jumpers and diodes added (refer to "Processing Tcl Lists" on page 1018).
-1	The command did not run due to a syntax error or a missing required argument.

Example

The following example fixes process antenna violations in the entire design by adding jumpers and diodes and creates annotations for the additions.

```
fix_antenna -all
```

Related Information

Tcl Command	check_antenna

fix_diff_track_errors

```
fix_diff_track_errors
     [ -error_set [ selection set ] ]
     [ -passes [integer] ]
     [ -show_set [ true | false ] ]
     [ -punch_out_diff_track_eol [ true | false ] ]
     [ -punch_for_diff_track_eol [ true | false ] ]
```

Gathers all shapes that violate the EOL rules and fixes these violations by deleting and re-routing these shapes.

Arguments

`-error_set`
Set of objects on which the operation would be performed. If this parameter is not provided, the operation is performed on the entire design and gather shapes that violate EOL rules.

`-passes [ true | false ]`
Number of passes to test how router is converging on diff-track EOL or minSideSpacing errors. It supports a maximum of five passes.

`-show_set [ true | false ]`
This option is for debugging purposes.

`-punch_out_diff_track_eols [ true | false ]`
When enabled, point-to-point punches out diff track EOL area on cut layer.

`-punch_for_diff_track_eols [ true | false ]`
When enabled, point-to-point punches out for cut layers for diff track EOL.

Related Information

Tcl Command	None

fix_litho_errors

```
fix_litho_errors
     [ -allow_hint_adjust [ true | false ] ]
     [ -allow_hintless_fixing [ true | false ] ]
     [ -annotate [ all | none | unfixed | ignored | SPACING | ENCLOSURE | WIDTH
     | LINEEND | SPACEEND | GATE_CD | s_errorTypeName ] ]
     [ -annotation_limit i_count ]
     [ -error_type {all | SPACING | ENCLOSURE | WIDTH | LINEEND | SPACEEND | GATE_CD
     | s_errorTypeName} ]
     [ -force [ true | false ] ]
     [ -incremental_check [ true | false ] ]
     [ -layers {s_layerName…} ]
     [ -mode { 1 | 2 | 3 } ]
     [ -no_repaint ]
     [ -push_limit {1 | 2 | 3 | 4 | 5} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -report s_fileName [ -report_unfixed [ true | false ] ] ]
     [ -severity {i_severityLevel…} ]
     [ -soft_rules_adherence [ true | false ] ]
     [ -top_level_only [ true | false ] ]
     [ -use_fill_shapes [ true | false ] ]
```

Attempts to fix lithography errors based on *Lithography Errors* given in the Violations page of the Annotation Browser. To run this command, you must first run read_litho_errors or check_litho_errors to create the lithography error annotations.

You can optionally incrementally fix and check lithography errors (`-incremental_check`) if you have Litho Physical Analyzer installed, have an LPA license, and first run initialize_lpa to enable Litho Physical Analyzer checking.

Important

For best results when using this command, you should enable gridless routing.

```
set_route_on_grid -on_grid false
```

Arguments

`-allow_hint_adjust [ true | false ]`

When `true`, hints can be adjusted in order to match the manufacturing grid or to transform some edge moves into shape moves.

Default: `true`

`-allow_hintless_fixing [ true | false ]`

When `true`, attempts to fix lithography errors that do not provide hints for fixing and when the provided hints are not successful. When `false`, hintless lithography errors will not be fixed.

Default: `true`

`-annotate` *`s_typeName`*

Specifies the type of adjustments to annotate.

`all`	Creates `Unfixed`, `Fixed`, and `Ignored` annotations as described for the respective conditions below.
`ignored`	Creates annotations for errors that are ignored. An ignored error is an error that Space-based Router and Chip Optimizer does not attempt to fix, because the original geometries that caused the violation have been changed by a previous fix, or because it is a hintless error and fixing hintless errors has not been enabled (-`allow_hintless_fixing`).
`none`	Does not create annotations. This is the default.
`unfixed`	Creates annotations for errors that were attempted, but could not be fixed, by this command.
`ENCLOSURE`	Creates annotations for enclosure errors that were fixed by this command.
`GATE_CD`	Creates annotations for device gate critical dimension errors that were fixed by this command.
`LINEEND`	Creates annotations for lineend errors that were fixed by this command.
`SPACEEND`	Creates annotations for spaceend errors that were fixed by this command.

`SPACING`	Creates annotations for spacing errors that were fixed by this command.
`WIDTH`	Creates annotations for width errors that were fixed by this command.
`s_errorTypeName`	Creates annotations for prototype errors that were fixed by this command.

The annotations are added to the `annotation:viaOpt` purpose of the given layer and are listed as `Unfixed`, `Fixed`, or `Ignored` by layer under *FixLithoErrors* on the Optimizations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that this command can create. This does not restrict the number of fixes attempted. Specify `-annotation_limit -1` to choose no limit.

Default: 1000

`-error_type {all | SPACING | ENCLOSURE | WIDTH | LINEEND | SPACEEND | GATE_CD |` *`s_errorTypeName`*`}`

Specifies the types of lithography errors to process.

Note: When giving specific error types, the names must match the syntax for the names in the Litho error file and are case-sensitive.

Default: `all`

`-force [ true | false ]`

When `true`, forces lithography errors that are marked *checked* (as indicated by the Properties Browser `isChecked` property and the Violations Browser *checked* attribute column for the annotation) to be processed by this command.
After a lithography error is fixed by `fix_litho_errors`, it is automatically marked *checked*.

Default: (`false`) Lithography errors that are marked *checked* are not processed by the command.

`-incremental_check [ true | false ]`

If set to `true`, Litho Physical Analyzer will be run in the region after each fix to check the results. If a lithography error still exists in the changed area, the change is undone and the next hint is attempted and checked, and so on, until successful or all hints have been tried.

Important

> To use this feature, you must first have Litho Physical Analyzer installed with an LPA license and run `initialize_lpa`.

Default: `false` (results are not checked by Litho Physical Analyzer)

`-layers` **{***`s_layerName…`***}**

Specifies the metal layers to operate on.

Default: All layers are operated on.

`-mode {1|2|3}`

Determines the level of fixing that will be attempted. Each higher level increases the methods used for fixing but also increases the disturbance to neighboring shapes.

`-mode 1` — (With and without hints) Permits pushing or partial pushing of wires.

`-mode 2` — (With and without hints) Mode `1` functionality plus pushing of vias.

`-mode 3` — (Without hints) Mode `2` functionality plus incremental re-routing.

(With hints) Mode `2` functionality plus incremental re-routing is applied only if `-allow_hintless_fixing` is `true`. Otherwise, if the hints are not successful in this mode, no other action is taken.

This mode is the default.

`-no_repaint`

Disables screen repaint after fixes. This argument is intended for use in scripts when multiple calls to the command are made back-to-back and a repaint is only desired on the last command.

Default: The screen is always repainted when errors are fixed.

`-push_limit {1 | 2 | 3 | 4 | 5}`

Limits the number of wires that can be pushed when adding space around an error segment.

Default: 2

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Fixes only errors within the specified area.

Default: The entire design is operated on.

`-report` *`s_fileName`*

Creates a Litho Hotspot check file with the given name that reports areas that were checked by this command. If `-report_unfixed` is set to `false`, only fixed areas that were checked are included in the check file, otherwise, unfixed and ignored areas are also included.

This file can be read using read_check_area to create `Changed` area annotations that can be used by `check_litho_errors`.

`-report_unfixed [ true | false ]`

If `true`, unfixed errors and ignored areas are included in the output Litho Hotspot check file.

Default: `true`

`-severity` **{***`i_severityLevel…`***}**

Specifies the order in which lithography errors will be fixed, according to their severity level.

Default: Will attempt to fix errors of all severity levels that are identified by lithography error annotations.

`-soft_rules_adherence [ true | false ]`

When set to `true`, uses soft constraints when performing design rule checks.

Default: Uses hard constraints for design rule checks.

`-top_level_only [ true | false ]`

When set to `true` and by default, performs design rule checks on top level shapes only. When set to `false,` design rule checks are performed on all levels.

`-use_fill_shapes [ true | false ]`

(Applies to width and line end errors only) When `false`, wires will not be widened or lengthened. The violations will be fixed by adding space around the violation where possible using push (all `-mode` values) or incremental routing (if `-mode 3` without hints, or `-mode 3` with hints and -`allow_hintless_fixing true`). This prevents fill shapes from being added to the design, allowing for easier implementation of ECOs and validation tasks.

When `true`, wires can be widened or lengthened using fill shapes to correct width and line end errors.

Default: `true`

Value Returned

`0`	Command was run.
`-1`	A syntax error occurred. The command failed.

During processing, the following messages might occur:

- `Error: LPA layer mapping failure: fail to map layer <layerName>`

  The layer, declared in the layer map file, does not exist in the OpenAccess database. Remove the layer from the layer map file.

- `Warning: Change detected on layer <layerName> but LPA has not been initialized for this layer`

  A change was made in the given layer but cannot be check for lithography errors because LPA has not been initialized for this layer. The layer is not declared in the LPA configuration file or there is no technology file associated with the layer in the LPA configuration file.

Note: Via layers are not currently supported in the technology file.

Examples

Example 1—Fix all width violations using only hints

```
fix_litho_errors -error_type width -allow_hintless_fixing false
```

Example 2—Fix all width violations using hints and hintless methods

```
fix_litho_errors -error_type width -allow_hintless_fixing true
```

Example 3—Fix all width violations without adding fill shapes and only using hints

```
fix_litho_errors -error_type width -allow_hintless_fixing false -use_fill_shapes
false
```

Example 4—Fill all violations using hintless methods including pushing wires and vias, and incremental wiring

```
fix_litho_errors -error_type all -allow_hintless_fixing true -mode 3
```

Example 5—Fix all violations and create annotations for unfixed errors

```
fix_litho_errors -annotate unfixed
```

Example 6—Incrementally fix all violations using only hints and create annotations for fixed, unfixed, and ignored errors, output all checked areas to a Litho check file

```
fix_litho_errors -annotate all -incremental_check true \
  -allow_hintless_fixing false -report area_check.hif
```

The following is an example of the Lithography Hotspots Fixing Summary that is output to the Transcript area.

```
Lithography Hotspots Fixing Summary:
# + LAYER DETAILS ===========================================================
# | Layer          |           |          Fixed         |         |         |
# |    (Severity)  | Analyzed  |  w/hints | w/o hints   | Ignored | Unfixed |
# |---------------------------------------------------------------------------|
# | metal2         |      63   |     46   |        16   |     0   |     1   |
# |             (1) |      (6) |      (3) |         (3) |     (0) |     (0) |
# |             (2) |     (19) |     (13) |         (5) |     (0) |     (1) |
# |             (3) |     (38) |     (30) |         (8) |     (0) |     (0) |
# | metal3         |      12   |      9   |         3   |     0   |     0   |
# |           (1) |      (6) |      (4) |       (2) |     (0) |     (0) |
# |           (2) |      (1) |      (1) |       (0) |     (0) |     (0) |
# |           (3) |      (5) |      (4) |       (1) |     (0) |     (0) |
# |---------------------------------------------------------------------|
# | Totals        |      75  |      55  |       19  |      0  |      1  |
# |           (1) |     (12) |      (7) |       (5) |     (0) |     (0) |
# |           (2) |     (20) |     (14) |       (5) |     (0) |     (1) |
# |           (3) |     (43) |     (34) |       (9) |     (0) |     (0) |
# =======================================================================
```

Related Information

Tcl Command	check_litho_errors read_check_area read_litho_errors

initialize_lpa

```
initialize_lpa
     -conf s_fileName
     [ -output_directory s_dirName ]
```

Checks out an Litho Physical Analyzer (LPA) license and loads a part of the LPA configuration (`.conf`) file. If this step succeeds, Litho Physical Analyzer checking is enabled for use within Chip Optimizer. You must successfully run this command before any of the following commands can be used:

- check_litho_errors
- fix_litho_errors `-incremental_check`

Arguments

`-conf` *`s_LPAConfFileName`*

Specifies the name of the LPA configuration file to be loaded.

`-output_directory` *`s_dirName`*

Specifies the results output directory.

Default: `.LpaCcoResults`

Value Returned

`0`	Litho Physical Analyzer is enabled for use.
`-1`	Either the license checkout or the LPA configuration file load failed. Litho Physical Analyzer checking cannot be run from Space-based Router and Chip Optimizer.

Related Information

Tcl Command	check_litho_errors fix_litho_errors

map_redundant_via

```
map_redundant_via
     -master_lib s_libName
     -master_cell s_cellName
     -master_view s_viewName
     -redundant_cell s_cellName
     -direction {UP | DOWN | RIGHT | LEFT}
     [ -redundant_lib s_libName ]
     [ -redundant_view s_viewName ]
     [ -no_save ]
```

Maps existing redundant vias to existing master vias. If you establish these mappings with this command before you remaster via instances using the `remaster_via` command, and prior to reverting remastered vias using unclone_via, only vias given by the map_redundant_via command will be used. To return to the preferred method for specifying remaster vias which uses the current setting for the `validRoutingVias` constraint, you must issue the `clear_redundant_via_mapping` command.

Arguments

`-direction {UP | DOWN | RIGHT | LEFT}`

Specifies the direction of the redundant via, relative to the original via, in reference to a transform of R0.

`UP`	Indicates the redundant via is located in a positive X direction and at the same Y coordinate as the original via.
`DOWN`	Indicates the redundant via is located in a negative X direction and at the same Y coordinate as the original via.
`RIGHT`	Indicates the redundant via is located in a positive Y direction and at the same X coordinate as the original via.
`LEFT`	indicates the redundant via is located in a negative Y direction and at the same X coordinate as the original via

`-master_cell` *`s_cellName`*

Specifies the original via master cell name.

`-master_lib` *`s_libName`*

Specifies the original via master library name.

`-master_view` *`s_viewName`*

Specifies the original via master view name.

`-no_save` Indicates the redundant via information is only active in the current session. If this argument is not given, the redundant via information is saved to disk.

`-redundant_cell` *`s_cellName`*

Specifies the new redundant via master cell name.

`-redundant_lib` *`s_libName`*

Specifies the new redundant via master library name.

`-redundant_view` *`s_viewName`*

Specifies the new redundant via master view name.

Example

The following Tcl commands map an original via master to four redundant via cellviews, one for each direction of UP, DOWN, RIGHT and LEFT.

```
map_redundant_via -master_lib olib -master_cell V121 -master_view via
-redundant_cell V126 -direction UP
map_redundant_via -master_lib olib -master_cell V121 -master_view via
-redundant_cell V125 -direction DOWN
map_redundant_via -master_lib olib -master_cell V121 -master_view via
-redundant_cell V124 -direction RIGHT
map_redundant_via -master_lib olib -master_cell V121 -master_view via
-redundant_cell V123 -direction LEFT
```

Related Information

Tcl Command	clear_redundant_via_mapping remaster_via show_redundant_via_mapping

net_strap

```
net_strap
     -set d_setObj
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -do_via_layers {s_layerName …} ]
     [ -add_vias [ true | false ] ]
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -maximize_cuts [ true | false ] ]
     [ -use_larger_cut_spacing [ true | false ] ]
     [ -auto_use_larger_cut_spacing [ true | false ] ]
     [ {-use_cut_classes [ true | false ] || -cut_class_names {s_cutClassName …}
     [ -cut_class_names_only [ true | false ] ] } ]
```

Determines if there are enough vias strapping down the power and ground buses and optionally adds vias where needed.

Note: To use this command to add vias, `minWidth`, `minSpacing` and an extension rule (`minExtension` or `minDualExtension`) must be set for the via layers processed.

Arguments

`-add_vias [ true | false ]`

Adds vias where needed.

`-annotate [ true | false ]`

Creates rectangle annotations in the artwork that represent the boundaries for areas where vias can be added. The annotations are listed under *Net Strapping* in the Optimizations page of the Annotation Browser and are added to the `annotation:viaOpt` purpose of the cut layer. By default, annotations are not added.

`-annotation_limit i_count`

Specifies the maximum number of annotations that can be added for a net on a cut layer. The default value is 1000.

`-auto_use_larger_cut_spacing [ true | false ]`

If true, the largest cut spacing (for example, the distance parameter for minAdjacentViaSpacing) is used for vias that are close enough to impact the placement of nearby vias. Otherwise, the smallest cut spacing is used.

Default: false

-clear_annotations

Removes existing net strap annotations before adding new ones.

-cut_class_names **{***s_cutClassName* …**}**

Uses vias with cuts taken from the given list of cut class names, in the order given. Regular expressions, such as *_Bar, can be used.

-cut_class_names_only [true | false]

If true, only cuts from the given list of cut classes are used. If false, any unspecified smaller cut classes are used after the named ones.

Default: false

-do_via_layers **{***s_layerName* …**}**

Restricts processing to the specified cut layers. If not specified, all via layers are processed.

-maximize_cuts [true | false]

When true, places the maximum possible cuts in any given intersection. By default (false), only the largest rectangle in any given metal intersection is filled.

-region **{***f_xlo f_ylo f_xhi f_yhi***}**

Limits processing to the given area (defined by the lower left and upper right coordinates). If not specified, the entire design is processed.

-set *d_setObj*

Specifies the set that contains the nets to be strapped.

-use_cut_classes [true | false]

Uses vias with cuts taken from the cut class constraints, from largest to smallest. By default, only the smallest cuts are used.

-use_larger_cut_spacing [true | false]

Uses the largest cut spacing (for example, the distance parameter for `minAdjacentViaSpacing`) when adding vias. By default, the minimum allowed cut spacing is used.

Example

The following command checks the nets in the selected set, adds up to 5 annotations per net for via layers that show the boundaries of areas that need additional strapping, then adds vias to those areas.

```
net_strap -set [get_selection_set] -annotate -annotation_limit 5
```

Related Information

Tcl Command	create_fill

pg_tap

```
pg_tap
     {-inst s_instName -term s_termName} | -set d_setObj
     -find_distance f_userunit
     [ -route_spec s_routeSpec ]
     [ -target_layer s_layerName ]
     [ -tap_signal_clearance f_userunit ]
     [ -result_set d_setObj ]
     [ -allow_violations ]
```

Adds direct power/ground connection to cells to reduce voltage drop issues associated with current flow (`I`) or electrical resistance (`R`). Space-based Router and Chip Optimizer finds areas where a power or ground route can be added, then adds and connects the extra route.

Arguments

`-allow_violations` Ignores violations when routing the tap. By default, the tap is routed following DRC rules.

`-find_distance` *`f_userunit`*

Specifies the maximum distance from the instTerm to search for the net to tap into. This defines an octagonal bounding box as the search area.

`-inst` *`s_instName`* Specifies the name of the instance to add the tap to.

`-result_set` *`d_setObj`* Adds tap connection shapes to the given set.

`-route_spec` *`s_routeSpec`*

Specifies the route spec for the tap.

`-tap_signal_clearance` *`f_userunit`*

Specifies the minimum clearance to the signal nets. The default value is the minimum spacing rule from the technology file is used.

`-target_layer` *`s_layerName`*

Specifies the name of the layer to connect the instTerm to. By default, the closest sequential layer that is within the search area.

`-term` *`s_termName`* Specifies the name of the instTerm to add the tap to.

Example

In this example, terminal A is a ground net term. The following command searches layer M4 around terminal A of instance SCLK_box_0_15 for an area within a10 micron distance to add a route connecting the terminal to the ground net. The route will use the single route spec.

```
pg_tap -inst SCLK_box_0_15 -term A -route_spec single -target_layer M4
-find_distance 10
```

read_check_area

```
read_check_area
     -file s_fileName
     [ -annotation_limit i_count ]
     [ -layers {s_layerName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
```

Reads a Litho Hotspot check file and creates annotations for check areas in the file. Annotations are added as layer `annotation:viaOpt` objects that are listed by layer as *Changed area* annotations in the Optimizations page of the Annotation Browser and indicate the source file.

Arguments

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that this command can create.

Default: 1000 per error type.

`-file` *`s_fileName`* Specifies the name of the Litho Hotspot Check file.

`-layers` **{***`s_layerName…`***}** Creates only annotations for the given layers.

Default: All layers are included.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Creates annotations only for checks areas in the Litho check file that are partially or fully within the given region.

Value Returned

`0`	The file was read successfully.
`-1`	An error occurred while reading the file.

Related Information

Tcl Command	fix_litho_errors

reduce_vias

```
reduce_vias
     [ -check_rule
       {all
       | none
       |[grid][minarea][minedge ][minenclosedarea][minwidth][numcuts][samenet]
       [samenet_crossing][samenet_portshort]} ]
     [ -check_antenna [ true | false ] ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -set d_setObj ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -critic [ true | false ] ]
     [ -threads i_threads ]
     [ -use_double_cut_vias [ true | false ] ]
```

Removes unnecessary vias and associated wiring from the design.

Note: Since this process can be time-consuming, it can be beneficial to limit the scope to a region or to nets in a set.

Arguments

`-check_antenna [ true | false ]`

If `true`, checks new routes created by this command for antenna violations and, if found, the new routes are discarded and the original route connections are kept. Default: `false`

`-check_rule {all | none | {s_check…}}`

Checks new routes created by this command for the specified rule violations.

`all`	(default) Performs all checks (`grid, minarea, minedge, minenclosedarea, minwidth, numcuts, samenet, samenet_crossing, samenet_portshort).`
`none`	No checks are performed.
{*s_check…***}**	Performs the checks in the list.

`-critic [ true | false ]`

Straightens wires where possible after routing. Defaults to `false`.

`-exclude_net` **{***`s_netName`*…**}**

Excludes the given nets from processing. Nets that are not fixed or locked in this list can be shifted while routing other nets.

`-exclude_set` *`d_setObj`* Excludes nets in the given set from processing.

`-exclude_type` **{**`[power][ground][clock]`**}**

Excludes the given types of nets from processing. Nets that are not fixed or locked in this list can be shifted while routing other nets.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Processes routes in the area given by the lower left (*`f_xlo f_ylo`*) and upper right (*`f_xhi f_yhi`*) coordinates.

Default: The entire top cellview is processed.

`-set` *`d_setObj`* Processes nets in the given set. If this argument is not specified, the entire top cell is processed.

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

`-use_double_cut_vias [ true | false ]`

If `true`, will route using double cut vias when possible.

Default: `false`

Example

The following example removes unnecessary vias and associated wiring within the given region.

```
reduce_vias -region [get_window_area]
```

remaster_via

```
remaster_via
     [ -window_id i_windowID ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj ]
     [ -do_via_layers {s_layerName …} ]
     [ -lock_layers {s_layerName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -exclude_via_pin [ true | false ] ]
     [ -output_pins_only [ true | false ] ]
     [ -insertion_mode {{on_wire | off_wire | on_wire_push | off_wire_push | all}…}
     ]
     [ -annotate all | unfixed | skipped | remastered | none ]
     [ -annotation_limit i_count ]
     [ -allow_num_cut_downsizing [ true | false ] ]
     [ -check_antenna [ true | false ] ]
     [ -clear_annotations ]
     [ -critic [ true | false ] ]
     [ -double_cut_only [ true | false ] | -single_cut_only [ true | false ]
     [ -ignore_active_route_status [ true | false ] ]
     [ -optimize_multi_cut_via_enclosure i_numCut ]
     [ -push_limit {1 | 2 | 3 | 4 | 5} ]
     [ -silent [ true | false ] ]
     [ -threads i_threads ]
     [ -use_grid {mfg | route} ]
     [ -valid_routing_vias {s_viaName…}
       [ -override_valid_routing_vias [ true | false ] ] ]
     [ -via_order {s_viaName…} [ -via_order_only [ true | false ] ] ]
     [ -lock_pin_metal [ true | false ] ]
```

Remasters vias in the target area of the cellview. You can optionally add annotations for remastered, skipped, or unfixed vias to the design on the `annotation:viaOpt` purpose of the via layer. On completion of this command, the Redundant Via Insertion Summary is output to the Transcript area and remastered vias are placed on the `via:redundant` purpose.

By default, all available vias given by the `validRoutingVias` constraint (including metal1/poly vias) are considered. To customize the list of considered vias, use `-valid_routing_vias`. To specify a list of vias to consider first, use `-via_order`.

Note: If the map_redundant_via command has been issued prior to using this command, only the via mappings given by the map_redundant_via command (typically sourced in a Tcl *mapping file* script that you must create) will be used to remaster vias. In the preferred method, which does not use a mapping file, Space-based Router and Chip Optimizer will attempt to remaster using vias given by the `validRoutingVias` constraint, starting with double cut vias and then larger enclosure single cut vias. If you translated your design from

LEF, `validRoutingVias` should automatically be set and include all of the vias in the library. If you have customized your design and added additional via cellviews, you must ensure that `validRoutingVias` includes the vias that you want to use for remastering. To return to the preferred method after using a mapping file, you must issue clear_redundant_via_mapping.

Arguments

`-all`

Sets the target area to the entire cellview. Attempts to remaster all vias in the cellview. This is the default.

`-allow_num_cut_downsizing [ true | false ]`

Permits vias to be remastered to a master with fewer cuts (for example, from two cuts to one), if the `-via_order` argument contains such an order. By default, the number of cuts on a via being remastered will increase or stay the same.

`-annotate all | unfixed | skipped | remastered | none`

Creates rectangle annotations in the via color and adds an entry for each via to the Annotation Browser Optimizations section, depending on the given argument:

`all`	Creates annotations for remastered, unfixed and skipped vias.
`none`	Creates no annotations. This is the default.
`remastered`	Creates annotations for newly remastered vias.
`skipped`	Creates annotations for vias that already have redundant cuts nearby.
`unfixed`	Creates annotations for unfixed vias only.

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that will be created for each annotation group. The default is 1000.

`-check_antenna [ true | false ]`

Checks for and avoids creating new antenna violations. Defaults to `false`.

`-clear_annotations`

Clears all existing optimized via annotations before adding new ones. By default, existing annotations are not cleared.

`-critic`

Causes pushed route segments to be straightened or cleaned up, whenever possible, during processing. Specifying this option has a minor negative impact on processing time. By default, this option is not enabled.

`-do_via_layers` **{***`s_layerName`* **…}**

Limits the remastering to the via layers in the given list. If this argument is not specified, all via layers are processed.

`-double_cut_only [ true | false ]`

Restricts processing to double cut vias only. By default, when no via mappings are given, larger enclosure single cut vias are tried after all double cut vias have been tried.

`-exclude_net` **{***`s_netName`* **…}**

Specifies the names of nets to exclude from processing.

`-exclude_set` *`d_setObj`*

Excludes from processing vias, routes and nets in the given set.

`-exclude_type` **{**`[power][ground][clock]`**}**

Specifies one or more net types to exclude from processing.

`-exclude_via_pin [ true | false ]`

When `true`, excludes via pins (top level only) from being remastered.

Default: `true`

`-ignore_active_route_status [ true | false ]`

Processes vias belonging to fixed routes. By default, these vias are skipped.

`-insertion_mode`

Specifies the insertion mode or a list of insertion modes to use. If a list of modes is specified, the modes will be processed in the order given in the list.

`on_wire` Puts remastered vias on the connecting wire without pushing routes.

`off_wire`	Puts remastered vias off the connecting wire without pushing routes.
`on_wire_push`	Puts remastered vias on the connecting wire, pushing surrounding shapes, if needed.
`off_wire_push`	Puts remastered vias off the connecting wire, pushing surrounding shapes, if needed.
`all`	Uses insertion mode techniques in the following order: `on_wire`, `off_wire`, `on_wire_push`, `off_wire_push`. If a via cannot be remastered using one technique, the next technique is attempted, and so on.

`-lock_layers` **{***`s_layerName`…***}**

Specifies the metal layers that must not be changed during this operation.

`-lock_pin_metal [ true | false ]`

If `true`, no additional metal will be added if the via touches a pin. If a pin shape is present at the via on the above/ below layer, then that metal layer is considered locked for the purpose of remastering that particular via. The above/ below layers are considered independently, so locking on one does not affect the other.

Default: `false`

`-optimize_multi_cut_via_enclosure` *`i_numCut`*

Specifies the maximum number of cuts in a multi-cut via for which larger enclosure via masters will be considered.

Default: 1 (Vias that already have two or more cuts will not be processed)

`-output_pins_only [ true | false ]`

Limits processing to vias on output pins.

Default: `false`

`-override_valid_routing_vias [ true | false ]`

When `true`, permits a via master that is defined in the technology file and specified in the `-valid_routing_vias` argument to be used for remastering even if it is not included in the `validRoutingVias` or `extendedValidRoutingVias` constraint for the net.

Default: `false`

`-push_limit {1|2|3|4|5}` Limits the number of routes that can be pushed when remastering a via using a push insertion mode.

Default: 2

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Attempts to remaster vias only in the specified area of the cellview, given by the lower left and upper right coordinates.

`-set` *`d_setObj`* Attempts only to remaster vias in the specified set.

`-silent [ true | false ]`

When `true`, outputs only essential messages. By default and when `false`, additional status messages are output.

`-single_cut_only [ true | false ]`

Restricts processing to single-cut vias only. By default, when no via mappings are given, larger enclosure single cut vias are tried after all double cut vias have been tried.

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled (enable_multithreading), the session threads are used, otherwise, one processor is used.

`-use_grid {mfg | route}`

Specifies the grid to snap to for vias and pushed wires:

`mfg` Uses the manufacturing grid from the technology file. This is default setting.

`route` Uses the routing grid from the technology file. This is the coarsest grid. Snapping to this grid reduces the redundancy rate.

`-valid_routing_vias` **{***`s_viaName…`***}**

Specifies the list of via masters to consider for remastering a given via, in no particular order. String patterns are not considered. Only via masters in the `validRoutingVias` or `extendedValidRoutingVias` constraint for the net will be considered, unless `-override_valid_routing_vias` is set to `true`.

Default: All available via masters are considered.

`-via_order` **{***`s_viaName…`***}**

Specifies the list of via masters, in the preferred order, to consider first when remastering a given via. Other potential via masters that are not in this list are considered next. String patterns, such as “`v*_2cut_*`”, which could match `v01_2cut_N`, `v01_2cut_S`, `v02_2cut_N`, and so on, are supported.

`-via_order_only [ true | false ]`

When `true`, limits remastering to new via masters that are given by the `-via_order` argument and are also included in the `-valid_routing_vias` list. By default, all via masters are considered, with priority given to those in the `-via_order` list.

`-window_id` *`i_windowID`* Specifies the window to process. If this argument is not specified, the active artwork window is used.

Example

The following example remasters vias on-wire for via layers `V1` and `V2` of entire design.

```
remaster_via -all -do_via_layers {"V1" "V2"} -insertion_mode on_wire
```

The following example inserts redundant vias for all vias in the design using all insertion mode techniques.

```
remaster_via -all -insertion_mode all
```

Related Information

Tcl Command	clear_redundant_via_mapping map_redundant_via show_redundant_via_mapping report_via_stats

show_redundant_via_mapping

```
show_redundant_via_mapping
     [ -command_form ]
```

Lists all current redundant via mappings in the Transcript area.

Arguments

`-command_form`

Outputs the information as map_redundant_via commands. By default, the information is output as follows:

master lib/cell/view direction -> redundant lib/cell/view

Example

The following example shows output for `show_redundant_via_mapping`:

```
mylib/M2_M1/via LEFT  -> mylib/M2_M1_x2_west/via
mylib/M2_M1/via RIGHT -> mylib/M2_M1_x2_east/via
mylib/M2_M1/via DOWN  -> mylib/M2_M1_x2_south/via
mylib/M2_M1/via UP    -> mylib/M2_M1_x2_north/via
```

The following example shows output when using `-command_form`:

```
map_redundant_via -master_lib mylib -master_cell M2_M1 -master_view via
-redundant_cell M2_M1_x2_west -direction LEFT

map_redundant_via -master_lib mylib -master_cell M2_M1 -master_view via
-redundant_cell M2_M1_x2_east -direction RIGHT

map_redundant_via -master_lib mylib -master_cell M2_M1 -master_view via
-redundant_cell M2_M1_x2_north -direction UP

map_redundant_via -master_lib mylib -master_cell M2_M1 -master_view via
-redundant_cell M2_M1_x2_south -direction DOWN
```

Related Information

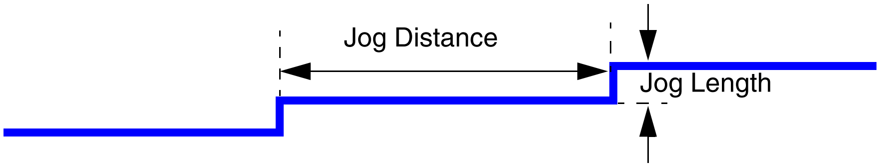
Tcl Command	clear_redundant_via_mapping map_redundant_via

spread_wire

```
spread_wire
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -layer {s_layerName…} ]
     [ -lock_vias [ true | false ] ]
     [ -min_jog_distance f_tracks ]
     [ -soft_rules_adherence [ true | false ] ]
     [ -target_spacing f_multiplier ]
     [ -threads i_threads ]
     [ -x_step f_step ]
     [ -y_step f_step ]
```

Spreads and bends wire for yield optimization while maintaining the current dimensions. Use the `-target_spacing` argument to set the targeted spread spacing as a multiple of the foundry `minSpacing` for each layer. A smaller `min_jog_distance` value results in more spreading, reducing the likelihood of shorts, at the potential cost of an increased number of jogs.

Use the `minJogLength` constraint to control the minimum height of the jog.



Use `-min_jog_distance` to set the minimum jog distance. Use the `minJogLength` constraint to set the minimum jog length.

Arguments

`-all` Operates on the entire design. This is the default.

`-exclude_type {[power][ground][clock]}`

Specifies one or more net types to exclude from processing. By default, no types are excluded.

`-layer {s_layerName…}`

Permits only shapes and vias on the specified layers to be moved. Default: All routing layers are processed.

`-lock_vias [ true | false ]`

If set `true`, vias are not moved.

Default: Vias can be moved when spreading wires.

`-min_jog_distance` *`f_tracks`*

Specifies the minimum distance, in tracks, between two jogs on the same wire. A smaller distance can result in more spreading, at the potential cost of an increased number of jogs.

Default: 2 tracks (one track is `minSpacing` + `minWidth`)

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the target area for this command.

Default: The entire design is processed.

`-soft_rules_adherence [ true | false ]`

When `true`, soft spacing rules are obeyed.

Default: Hard rules are obeyed.

`-target_spacing` *`f_multiplier`*

Specifies the desired amount of spacing between wires as a multiple of the foundry `minSpacing` constraint.

Default: 2 (The default target spacing is 2*`minSpacing`.)

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-x_step` *`f_userunit`* Specifies the spread step size, in user units. By default, the routing grid step is used for gridded routing, and the manufacturing step size is used for gridless routing.

`-y_step` *`f_userunit`* Specifies the spread step size, in user units. By default, the routing grid step is used for gridded routing, and the manufacturing step size is used for gridless routing.

Example

The following example spreads wires on `M2` in the active workspace. Jogs must be at least 3 µm apart.

```
spread_wire -region [get_window_area] -layer {M2} -min_jog_distance 3
```

Related Information

Menu Command	*Optimize—Wires*

terminate_lpa

```
terminate_lpa
```

Releases the LPA license and frees the LPA model. After this command is run, Litho Physical Analyzer checking will no longer be enabled. To use lithography commands, you must run initialize_lpa again.

Arguments

None

Related Information

Tcl Command	initialize_lpa

unclone_via

```
unclone_via
     {-set d_setObj
     | -all [ true | false ]
     | {-violators
       [ -rules {s_ruleName …} ]
       [ -expansion_distance f_userunit ]
       [ -max_vias_per_violation i_count ]}}
     [ -uncloned_set d_setObj ]
     [ -ignore_active_route_status [ true | false ] ]
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -silent [ true | false ] ]
```

Reverts remastered vias to their original masters.

Important

> `unclone_via` will revert vias that were remastered in the current session. To use this command in a later session for remastered vias in the design, you must set the environment variable, `db.save_original_via_master_property`, to `true` prior to saving the design with the newly remastered vias, and again before opening the saved view. Setting this variable before saving the design causes the original via properties to be saved for each remastered via, and setting the variable before opening the saved design causes those properties to be loaded when the design is reopened.
>
> ```
> setvar db.save_original_via_master_property true
> ```

You can optionally replace remastered vias with the via given by the `lib`, `cell`, and `view` arguments.

Arguments

`-all [ true | false ]` When `true`, reverts all remastered vias to their original masters.

`-expansion_distance` *`f_userunit`*

Distance to expand the Mentor Graphics® Calibre® error bounding box so that it overlaps the violating via. The default is 0.0.

`-ignore_active_route_status [ true | false ]`

Reverts vias on fixed routes. By default, these vias are skipped.

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

(Optional) Specifies the library, cell and view of the replacement via.

Caution

This option replaces all vias in the specified set with the given via.

`-max_vias_per_violation` *`i_count`*

Specifies the maximum number of vias to revert for each processed Calibre violation if more than one remastered via overlaps the error bounding box. By default, only one via is processed per Calibre violation.

`-rules` **{***`s_ruleName`* **…}** Restricts processing of Calibre errors to the given rules. If not specified, all rules are processed.

`-set` *`d_setObj`* Specifies the set that contains the remastered vias to revert.

`-silent [ true | false ]`

When `true`, outputs only essential messages. By default and when `false`, additional status messages are output.

`-uncloned_set` *`d_setObj`* Puts all reverted vias into this set. Can be used only with the `-violators` argument.

`-violators` Reverts any remastered vias that cause violations as reported by the checker or a Calibre error file. You must run the check commands and/or load the Calibre files prior to using this command.

Value Returned

`i_count` Is the total number of remastered vias that are reverted.

`-1` The command failed due to a syntax error.

Example

The following example reverts remastered vias in the selected set to their original master vias.

```
unclone_via -set [get_selection_set]
```

Related Information

Tcl Command	map_redundant_via remaster_via

widen_jogs

```
widen_jogs
     {-jog_target_width f_userunit | -check_only [ true | false ]}
     [ -jog_width f_userunit ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj
     | -nets {s_netName…} ]
     [ -lpp {s_layerlpp …} ]
     [ -annotate all | unfixed | partial | optimal | none ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -max_jog_length f_userunit ]
     [ -min_jog_length f_userunit ]
     [ -non_preferred_direction_only [ true | false ] ]
     [ -prop_name s_propertyName -prop_value s_propertyValue ]
     [ -use_fill_purpose ]
```

Widens jogs that are smaller than or equal to the given jog width (`-jog_width`) or, by default, the minimum layer width. Can be further qualified by length (`-min_jog_length` and `-max_jog_length`). Shapes are added to the `gapFill` (default) or `fill` purpose of the routing layer.

Arguments

`-all` Sets the target area to the entire cellview. Attempts to widen all jogs that meet the criteria in the cellview.

`-annotate all | unfixed | partial | optimal | none`

Creates rectangle annotations and adds an entry for each annotation to the Annotation Browser Optimizations section, depending on the given argument:

`all`	Creates annotations for widened and unfixed jogs.
`none`	Creates no annotations. This is the default.
`partial`	Creates annotations for jogs that were widened but not to the target width.
`optimal`	Creates annotations for jogs that were widened to the target width.

`unfixed` Creates annotations for jogs that met the criteria but could not be widened.

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that will be created for each annotation group. The default is 1000.

`-check_only` Prevents jogs from being widened but reports the number of jogs found that meet the criteria as one of the following: `unfixed` (cannot be widened due to DRC limitations), `partial` (can be widened, but not to target width due to DRC limitations), or `optimal` (can be widened to target width).

`-clear_annotations` Clears all existing Jog Width Optimizations annotations before adding new ones. By default, existing annotations are not cleared.

`-exclude_net` **{***`s_netName`* …**}**

Specifies the names of nets to exclude from processing.

`-exclude_set` *`d_setObj`* Excludes from processing vias, routes, and nets in the given set.

`-exclude_type` **{**`[power][ground][clock]`**}**

Specifies one or more net types to exclude from processing.

`-jog_target_width` *`f_userunit`*

Specifies the desired width for jogs.

`-jog_width` *`f_userunit`* Specifies that any segment that is smaller than or equal to this width be considered for widening. If this argument is not given, the minimum width for the layer is used.

`-lpp` **{***`s_layerlpp`* …**}**

Restricts processing to the specified layers. If not specified, all routing layers are processed.

`-max_jog_length` *`f_userunit`*

Specifies that any segment that is less than or equal to this length, measured in user units from outside edge to outside edge, will be considered for widening.

Default: No length restriction

`-min_jog_length` *f_userunit*

Only considers segments that are greater than or equal to this length, measured in user units from outside edge to outside edge.

Default: If the `minJogLength` constraint is set, the default is the `minJogLength` constraint + `minWidth`; otherwise the default is `minWidth*3`.

`-nets` **{***s_netName…***}**

Operates only on the nets in the list.

`-non_preferred_direction_only [ true | false ]`

Prevents jog segments that are in the preferred direction from being widened. By default, all jogs are processed regardless of orientation, except for jogs that are connected, such as a stair-step or U-shaped configuration; in these cases, the preferred direction segments in the cluster are skipped.

`-prop_name` *s_propertyName*

Attaches the given property to the added fill shapes. Must be specified with `-prop_value`. The property can be viewed using the Properties Browser or queried using inspect_getprop or inspect_prop.

`-prop_value` *s_propertyValue*

Assigns the value to the property given by `-prop_name`. The property can be viewed using the Properties Browser or queried using inspect_getprop or inspect_prop.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**

Attempts to widen jogs only in the specified area of the cellview, given by the lower left and upper right coordinates.

`-set` *d_setObj*

Operates only on objects in the given set.

`-use_fill_purpose`

Widens jogs by adding `fill` purpose shapes. By default, shapes are added to the `gapFill` purpose.

Example

The following example widens jogs that are smaller than 0.5 user units in length and larger than or equal to `minJogLength` plus `minWidth` width (or `minWidth*3` if `minJogLength` is not set) for the Metal2 layer to a desired width of 0.65.

```
widen_jogs -lpp Metal2 -max_jog_length 0.5 -jog_target_width 0.65
```

widen_wire

```
widen_wire
     [ -all | -set d_setObj ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -layer {s_layerName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -asymmetric_allowed [ true | false ] ]
     [ -ignore_active_route_status [ true | false ] ]
     [ -soft_rules_adherence [ true | false ] ]
     [ -split [ true | false ] ]
     [ -threads i_threads ]
```

Widens route segments by adding rectangle shapes on the `gapFill` purpose. You can limit the scope of processing to one or more layers, objects in a set, or in a given region. By default, all route segments in the entire design are processed. The `gapFill` shapes are added without modifying the existing route segments, and, when exported, will appear as `DRCFILL` in the `SPECIALNETS` section of the DEF.

The `wideningTargetWidth` layer constraint determines the amount of widening to attempt. If this constraint is not set, Space-based Router and Chip Optimizer attempts to widen route segments by 110% of the current wire width. Widening is performed using the following strategies: splitting, symmetrical widening, and asymmetrical widening (can be disabled).

The `wideningMinSplitValue` layer constraint defines the minimum length, in microns, for a split. The minimum and default value is 1 track. A value smaller than one track will be ignored.

To widen wires on fixed routes, use `-ignore_active_route_status true`. Otherwise, these wires are ignored.

Arguments

`-all` Attempts to widen all route segments in the design.

`-asymmetric_allowed [ true | false ]`

If set to `true` and by default, symmetric and asymmetric widening is performed. If set to `false`, only symmetric widening is performed.

`-exclude_type {[power][ground][clock]}`

Specifies one or more net types to exclude from processing.

`-ignore_active_route_status [ true | false ]`

If set to `true`, wires on fixed routes will also be processed. By default and when set to `false`, wires on fixed routes are not processed.

`-layer` **{***`s_layerName…`***}** Specifies the metal layers to operate on.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Limits processing to route segments in the given boundary. By default, the entire design is operated on.

`-set` *`d_setObj`* Attempts to widen route segments only for objects in the set.

`-soft_rules_adherence [ true | false ]`

When set to `true`, soft spacing rules are obeyed and widening that causes violations is not permitted.

Default: `(false`) Hard spacing rules are obeyed.

`-split [ true | false ]`

Specifies whether widening must be done on the entire length of a route segment (`false`), or can be split (`true`) where necessary to avoid DRC and connectivity violations. By default, splitting is permitted.

`-threads` *`i_threads`* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

Example

The following example widens route segments on the Metal2 layer in the current active window.

```
widen_wire -layer Metal2 -region [get_window_area]
```

Related Information

Tcl Command	`report_yield`

13

InPlace Cover Obstruction Commands

The InPlaceCoverObstruction Tcl commands are used to create or destroy InPlaceCoverObstruction objects and to retrieve or set InPlaceCoverObstruction attributes. This chapter describes the following InPlaceCoverObstruction Tcl commands:

- create_inplace_cover_obstruction
- get_inplace_cover_obstruction_bloat
- get_inplace_cover_obstruction_blockage_attribute
- get_inplace_cover_obstruction_blockage_model
- get_inplace_cover_obstruction_doughnut_halo
- get_inplace_cover_obstruction_layers
- get_inplace_cover_obstruction_min_mask
- get_inplace_cover_obstruction_max_mask
- get_inplace_cover_obstruction_pin_remodeling
- get_inplace_cover_obstruction_spacing_model
- has_inplace_cover_obstruction
- remove_inplace_cover_obstruction
- set_inplace_cover_obstruction_bloat
- set_inplace_cover_obstruction_blockage_attribute
- set_inplace_cover_obstruction_blockage_model
- set_inplace_cover_obstruction_doughnut_halo
- set_inplace_cover_obstruction_pin_remodeling
- set_inplace_cover_obstruction_spacing_model

create_inplace_cover_obstruction

```
create_inplace_cover_obstruction(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -rangeType s_rangeType ]
     [ -bound1 l_bound1 -bound2 l_bound2 ]
```

Creates an InPlaceCoverObstruction object on the specified cellview of the library.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

`-bound1` *`l_bound1`* `-bound2` *`l_bound2`*

Integer mask range (`minMask`, `maxMask`) for the InPlaceCoverObstruction to be created in the cellview.

`-rangeType` *`s_rangeType`*

Specifies the type of mask range. The valid values are: `lessThan`, `lessThanEqual`, `greaterThan`, `greaterThanEqual`, `closed`, `open`, `openLeft`, and `openRight`.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` for which the InPlaceCoverObstruction is to be created.

```
create_inplace_cover_obstruction -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -rangeType closed -bound1 0 -bound2
4
```

get_inplace_cover_obstruction_bloat

```
get_inplace_cover_obstruction_bloat(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
```

Returns value of the InPlaceCoverObstruction bloat attribute from the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` and layer name `L3` for which the InPlaceCoverObstruction bloat attribute value is returned.

```
get_inplace_cover_obstruction_bloat  -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L3
```

get_inplace_cover_obstruction_blockage_attribute

```
get_inplace_cover_obstruction_blockage_attribute(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
```

Returns value of the InPlaceCoverObstruction blockage attribute from the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` and layer name `L0` for which the InPlaceCoverObstruction blockage attribute value is returned.

```
get_inplace_cover_obstruction_blockage_attribute -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0
```

get_inplace_cover_obstruction_blockage_model

```
get_inplace_cover_obstruction_blockage_model(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
```

Returns value of the InPlaceCoverObstruction blockage model attribute from the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` and layer name `L0` for which the InPlaceCoverObstruction blockage model attribute value is returned.

```
get_inplace_cover_obstruction_blockage_model -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0
```

get_inplace_cover_obstruction_doughnut_halo

```
get_inplace_cover_obstruction_doughnut_halo(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
```

Returns value of the InPlaceCoverObstruction doughnut halo attribute from the specified cellview on the given layer.

Arguments

`-lib s_libName -cell s_cellName -view s_viewName`

Specifies the library name, cell name, and view name.

`-layer s_layerName`

Name of a physical layer specified in the technology file.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` and layer name `L3` for which the InPlaceCoverObstruction doughnut halo attribute value is returned.

```
get_inplace_cover_obstruction_doughnut_halo  -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L3
```

get_inplace_cover_obstruction_layers

```
get_inplace_cover_obstruction_layers
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Returns a list of physical layer names in the specified cellview of the library with mask value lying between the `minMask` and `maxMask` values.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

Example

The following example specifies the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout` for which a list of physical layer names is returned.

```
get_inplace_cover_obstruction_layers -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

get_inplace_cover_obstruction_max_mask

```
get_inplace_cover_obstruction_max_mask(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Returns value of the InPlaceCoverObstruction max mask attribute from the specified cellview of the library layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

Example

The following example specifies the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout` for which the InPlaceCoverObstruction max mask attribute value is returned.

```
get_inplace_cover_obstruction_max_mask -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

get_inplace_cover_obstruction_min_mask

```
get_inplace_cover_obstruction_min_mask(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Returns value of the InPlaceCoverObstruction min mask attribute from the specified cellview of the library layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and view name.

Example

The following example specifies the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout` for which the InPlaceCoverObstruction min mask attribute value is returned.

```
get_inplace_cover_obstruction_min_mask -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

get_inplace_cover_obstruction_pin_remodeling

```
get_inplace_cover_obstruction_pin_remodeling
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Returns value of the InPlaceCoverObstruction pin remodeling attribute from the specified cellview of the library.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

Example

The following example specifies the given cellview `mylib/mycell/layout` for which the InPlaceCoverObstruction pin remodeling attribute value is returned.

```
get_inplace_cover_obstruction_pin_remodeling -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

get_inplace_cover_obstruction_spacing_model

```
get_inplace_cover_obstruction_spacing_model(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
```

Returns value of the InPlaceCoverObstruction spacing model attribute from the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

Example

The following example specifies the given cellview `unitTestLib/ixInPlaceCoverObstruction_master1/layout` and layer name `L0` for which the InPlaceCoverObstruction spacing model attribute value is returned.

```
get_inplace_cover_obstruction_spacing_model -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0
```

has_inplace_cover_obstruction

```
has_inplace_cover_obstruction
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Checks whether an InPlaceCoverObstruction object is present on the specified cellview.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

Example

The following example checks the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout` for InPlaceCoverObstruction.

```
has_inplace_cover_obstruction -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

remove_inplace_cover_obstruction

```
remove_inplace_cover_obstruction(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
```

Removes an InPlaceCoverObstruction object from the specified cellview of the library.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

Example

The following example specifies the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout` for which the InPlaceCoverObstruction is to be removed.

```
has_inplace_cover_obstruction -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout
```

set_inplace_cover_obstruction_bloat

```
set_inplace_cover_obstruction_bloat(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName} ]
     [ -bloat i_value ]
```

Sets value of the InPlaceCoverObstruction bloat attribute on the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

`-bloat` *`i_value`* Value of bloat attribute that needs to be set.

Example

The following example sets the value of the InPlaceCoverObstruction bloat attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_bloat  -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L3 -bloat 7
```

set_inplace_cover_obstruction_blockage_attribute

```
set_inplace_cover_obstruction_blockage_attribute(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
     [ -attribute f_attributeValue ]
```

Sets value of the InPlaceCoverObstruction blockage attribute on the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

`-attribute` *`f_attributeValue`*

Value of blockage attribute that needs to be set.

Example

The following example sets the value of the InPlaceCoverObstruction blockage attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_blockage_attribute -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0 -attribute 5.5
```

set_inplace_cover_obstruction_blockage_model

```
set_inplace_cover_obstruction_blockage_model(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
     [ -model s_modelName ]
```

Sets value of the InPlaceCoverObstruction blockage model attribute on the specified cellview of the library on the specified layer.

Arguments

`-lib s_libName -cell s_cellName -view s_viewName`

Specifies the library name, cell name, and the view name.

`-layer s_layerName`

Name of a physical layer specified in the technology file.

`-model s_modelName`

Name of a valid blockage model. The valid values are: `fullCover`, `detailed`, `horizontal`, `vertical`, `doughnut`, and `shrinkAndWrap`.

Example

The following example sets the value of the InPlaceCoverObstruction blockage model attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_blockage_attribute -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0 -attribute 5.5
```

set_inplace_cover_obstruction_doughnut_halo

```
set_inplace_cover_obstruction_doughnut_halo(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
     [ -halo i_haloValue ]
```

Sets value of the InPlaceCoverObstruction doughnut halo attribute on the specified cellview on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

`-halo` *`i_haloValue`* Specifies the doughnut halo value that needs to be set.

Example

The following example sets the value of the InPlaceCoverObstruction doughnut halo attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_doughnut_halo  -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L3 -halo 7
```

set_inplace_cover_obstruction_pin_remodeling

```
set_inplace_cover_obstruction_pin_remodeling(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -model s_remodelingName ]
```

Sets value of the InPlaceCoverObstruction pin remodeling attribute on the specified cellview of the library.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-model` *`s_remodelingName`*

Name of a valid pin remodeling. The valid values are: `none`, `connectedShapes`, `wholeNet`, and `perimeterOnly`.

Example

The following example sets the value of the InPlaceCoverObstruction pin remodeling attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_pin_remodeling -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -model wholeNet
```

set_inplace_cover_obstruction_spacing_model

```
set_inplace_cover_obstruction_spacing_model(
     [ -lib s_libName -cell s_cellName -view s_viewName ]
     [ -layer {s_layerName…} ]
     [ -model s_spacingModelName ]
```

Sets value of the InPlaceCoverObstruction spacing model attribute on the specified cellview of the library on the given layer.

Arguments

`-lib` *`s_libName`* `-cell` *`s_cellName`* `-view` *`s_viewName`*

Specifies the library name, cell name, and the view name.

`-layer` *`s_layerName`*

Name of a physical layer specified in the technology file.

`t_spacingModelName`

Name of a valid spacing model. The valid values are: `spacing`, `effectiveWidth`, `computedEffectiveWidth`, and `asIs`.

Example

The following example sets the value of the InPlaceCoverObstruction spacing model attribute for the given cellview `unitTestlib/ixInPlaceCoverObstruction_master1/layout`.

```
set_inplace_cover_obstruction_spacing_model -lib unitTestLib -cell
ixInPlaceCoverObstruction_master1 -view layout -layer L0 -model
computedEffectiveWidth
```

14

Annotation Commands

This chapter describes the annotation object helper commands. The commands are presented in alphabetic order:

add_annotation

```
add_annotation
     -object CtuId
     -parent_group CtuId
     [ -make_group [ true | false ] ]
     [ -message messagestring ]
     [ -name namestring ]
```

Annotates the specified object.

Arguments

`-make_group [ true | false ]`	Creates group to contain new annotations.
`-message` *`messagestring`*	Message text for the new annotation.
`-name` *`namestring`*	Name of the new annotation.
`-object` *`CtuId`*	Object to be annotated.
`-parent_group` *`CtuId`*	Parent group for the new annotation or group.

add_net_annotation

```
add_net_annotation
     { [ -net string ] | [ -net1 string ] | [ -net2 string ] }
     { [ -from Coord] | [ -rect BBox ] | [ -to Coord ] }
     -parent_group CtuId
     [ -make_group [ true | false ] ]
     [ -message messagestring ]
     [ -name namestring ]
```

Annotates one or a pair of nets with a marker.

Arguments

`-from` *Coord*	Start point of line used to mark the nets.
`-make_group [ true | false ]`	Creates group to contain new annotations.
`-message` *messagestring*	Message text for the new annotation.
`-name` *namestring*	Name of the new annotation.
`-net` *string*	Net to be marked.
`-net1` *string*	First net to be marked.
`-net2` *string*	Second net to be marked.
`-parent_group` *CtuId*	Parent group for the new annotation or group.
`-rect` *BBox*	Dimensions of rectangle to mark the nets.
`-to` *Coord*	End point of line used to mark the nets.

delete_annotation

```
delete_annotation
     { [ -id CtuId ] | [ -set SelectionSet ] }
```

Deletes the specified annotation object by ID or set.

Arguments

`-id` *`CtuId`*	ID of the annotation object to delete.
`-set` *`SelectionSet`*	Set containing annotation objects to delete.

delete_annotations

```
delete_annotations
     { [ -ids CtuIdList ] | [ -set SelectionSet ] }
```

Deletes the specified annotation object(s) by ID or set.

Arguments

`-ids` *`CtuIdList`*	List of annotation object IDs to delete.
`-set` *`SelectionSet`*	Set containing annotation objects to delete.

find_annotations

```
find_annotations
     [ { [ -find_message messagestring ] | [ -find_name namestring ] }
     [ -group CtuId | -optimizations [ true | false ] | -user [ true | false ] |
     -violations [ true | false ] ]
     [ -no_occ [ true | false ] | -occurrence occurrenceId ]
     [ -groups [ true | false ] ]
     [ -recurse [ true | false ] ]
```

Finds annotation objects by regular expression.

Arguments

`-find_message` *`messagestring`*	Regular expression for annotation message search.
`-find_name` *`namestring`*	Regular expression for annotation name search.
`-group` *`CtuId`*	Searches in group.
`-groups [ true | false ]`	Searches groups (name, message) and annotations.
`-no_occ [ true | false ]`	Selects from unaffiliated annotations.
`-occurrence` *`occurrenceId`*	Selects from specified occurrence.
`-optimizations [ true | false ]`	Selects from optimizations group.
`-recurse [ true | false ]`	Recurse.
`-user [ true | false ]`	Selects from user annotations group.
`-violations [ true | false ]`	Selects from violations group.

get_annotation_group

```
get_annotation_group
     [ -optimizations [ true | false ] | -user [ true | false ] | -violations [
     true | false ] ]
     [ -no_occ [ true | false ] | -occurrence occurrenceId ]
```

Gets the top group node ID for the specified tab in the browser, for the specified occurrence, or for the unassociated items. If no `–occurrence` or `–no_occ` option is specified, the active design occurrence is used.

Arguments

`-no_occ [ true | false ]`	Selects from unaffiliated annotations.
`-occurrence` *`occurrenceId`*	Selects from specified occurrence.
`-optimizations [ true | false ]`	Selects from optimizations group.
`-user [ true | false ]`	Selects from user annotations group.
`-violations [ true | false ]`	Selects from violations group.

group_annotations

```
group_annotations
     { [ -ids CtuIdList ] | [ -set SelectionSet ] }
     -parent_group CtuId
     [ -make_group [ true | false ] ]
     [ -message messagestring ]
     [ -name namestring ]
```

Groups specified annotations into a group.

Arguments

`-ids` *`CtuIdList`*	List of annotation object IDs to group.
`-make_group [ true | false ]`	Creates group to contain new annotations.
`-message` *`messagestring`*	Message text for the new group.
`-name` *`namestring`*	Name of the new group.
`-parent_group` *`CtuId`*	Parent group for the new annotation or group.
`-set` *`SelectionSet`*	Set containing annotation objects to group.

15

Analyze Commands

The Cadence® Chip Optimizer Analyze commands let you perform Critical Area Analysis to determine the probability of faults and the expected yield based on the technology you are using.

The analysis commands are presented in alphabetic order:

report_yield

```
report_yield
     [ -yld s_fileName ]
     [ -report s_fileName ]
     [ -gridX i_micron ]
     [ -gridY i_micron ]
     [ -detail [ true | false ] ]
```

Reports the expected yield for the design using critical area analysis, based on the technology you are using and the probability of failure of the cells, vias, and point defects in routing. Uses data from a yield technology file supplied by the fabricator. Generates a yield report and a yield map.

To display the generated yield map graphically, use display_color_map with one of the following `-name` arguments: `YldCell`, `YldVia`, `YldShort`, `YldOpen`, `YldRouting`.

Arguments

`-detail [ true | false ]`

Specifies a detailed report. The detailed report contains the information included in the non-detailed report, plus statistics by cell, via, and grid area. By default, a non-detailed report is generated.

`-gridX` *`i_micron`*

Specifies the step size of the yield map, in microns, in the horizontal direction. Default is 50.

`-gridY` *`i_micron`*

Specifies the step size of the yield map, in microns, in the vertical direction. Default is 50.

`-report` *`s_fileName`*

Specifies the yield report name. The report includes an overall yield result that you can use to estimate the improvement in yield from different optimization techniques. The default is *`cellName`*`.yld.rpt`.

`-yld` *`s_fileName`*

Specifies the name of the yield technology file. The default is *`cellName`*`.yld`.

Example

The following example creates a detailed yield report named `yield.rpt`.

```
report_yield -detail -report yield.rpt
```

Related Information

Tcl Commands	display_color_map

16

Verify Commands

The Verify commands let you examine connectivity in the design and check for DRC errors.

The commands are presented in alphabetic order:

check_antenna

```
check_antenna
     -all | -net netName | -set d_setObj
     [ -error_set d_setObj ]
     [ -exclude_nets {s_netName …} ]
     [ -annotate [ true | false ] ]
     [ -check {[PAR] [CAR]} ]
     [ -clear_annotations ]
     [ -detailed ]
     [ -diff_use_only [ {s_layerName…} ] ]
     [ -file [s_fileName ] ]
     [ -model {[OXIDE1] [OXIDE2] [OXIDE3] [OXIDE4]} ]
     [ -noIOPinDefault [ true | false ] ]
     [ -use_factors [ true | false ] ]
     [ -short_format [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -vioLimit i_count ]
```

Checks for process antenna violations for the entire design, a specific net, or nets in a set. You can choose to ignore specific nets or add violating nets to a set. Results are output to the Transcript area or to a file.

Important

> To use this command, your design data must have been imported from LEF and DEF files that include process antenna keywords for setting values used by this check.

Arguments

-all

Performs process antenna checks on all nets.

-annotate [true | false]

Creates annotations for violations.

The annotations are added to the annotation:violation purpose for the metal layer and are listed by model and net as *Process Antenna Violations* in the Violations page of the Annotation Browser.

-check **{**[PAR][CAR]**}**

Determines which checks to perform: PAR (Partial Area Ratio), CAR (Cumulative Area Ratio).

Default: PAR and CAR, whichever ratios are defined in the LEF file

-clear_annotations

Removes existing violation annotations of the same check type before performing this check. If not specified, existing violations are not cleared.

-detailed

Includes non-violating nets in the report. By default, only nets with violations are reported.

-diff_use_only [**{***s_layerName…***}**]

Specifies layers to apply area factor to when connected to diffusion. If no layer is given, then area factor applies to all layers when connected to diffusion.

Default: Area factor is not applied to any layer.

-error_set *d_setObj*

Adds violating nets to this set.

-exclude_nets **{***s_netName* …**}**

Ignores the given nets from the check. This argument is ignored if -net is specified.

-file [*s_fileName*]

Outputs the report to the given file. If you do not specify a filename, the report is output to antenna*yymmdd.hhmmss.log*.

-model

Specifies which oxide models to use: OXIDE1, OXIDE2, OXIDE3, OXIDE4.

Default: OXIDE1

`-net` *`s_netName`*

Performs process antenna checks on the given net.

`-noIOPinDefault [ true | false ]`

Prevents the antenna checker from using IO pin default values:

- ANTENNAINPUTGATEAREA
- ANTENNAOUTPUTDIFFAREA
- ANTENNAINOUTDIFFAREA

By default, these values are used for any input that does not have a gate area or any output that does not have diode area.

`-set` *`d_setObj`*

Performs process antenna checks on nets in the given set.

`-short_format [ true | false ]`

Reports only error nodes. By default, all terminals for each instance are reported.

`-silent [ true | false ]`

When set to `true`, prevents messages from being output.

Default: `true`

`-use_factors`

Uses area factors when computing PAR values.

`-vioLimit` *`i_count`*

Specifies the maximum number of violations to report. Default is 1000.

Value Returned

`i_netcount`
`i_violationcount`
`i_pincount`

Tcl list with the number of violating nets, violations, and violated pins found (refer to "Processing Tcl Lists" on page 1018).

`-1`

The command did not run due to a syntax error or a missing required argument.

Example

The following example performs `PAR` checks on `myNet` and reports violations to the `checkmyNet` file.

```
check_antenna -net myNet -file checkmyNet -check PAR
```

Related Information

Tcl Command	fix_antenna

check_connectivity

```
check_connectivity
     [ -all | -net {s_netName …} | -set d_setObj ]
     [ -annotate [ -annotation_limit i_count ] ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -verbose ]
```

Checks nets for loops and dangles by examining the graph-based connectivity of terminals, Steiner points, and routes. The scope of the check can be the entire design in the active window, specified nets, or nets in the specified set. If the scope is not given, the selected set is operated on.

Arguments

`-all` Checks the connectivity of all nets in the active window. By default, all power and ground nets are ignored.

`-annotate` Adds annotations for loops and dangles that are found.

The annotations are added to the `annotations:extern viols` objects and are listed by net as *Connectivity Checks-Nets With Loops* or *Connectivity Checks-Nets With Dangles* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of annotations that can be created for this operation. The default value is 1000.

`-exclude_net` **{***`s_netName`* **…}**

Excludes the named nets from processing.

`-exclude_set` *`d_setObj`*

Excludes nets in the given set from processing.

`-exclude_type` **{**`[power][ground][clock]`**}**

Excludes nets of the specified types from processing.

`-net` **{***`s_netName`***…}** Checks the connectivity of the named nets.

`-set` *`d_setObj`* Checks the connectivity on nets in the set.

`-verbose` Outputs to the Transcript area the names of checked nets with dangles and/or loops and the number of loops and dangles found in each. If this argument is not specified, only a summary of the number of nets checked and the total number of dangles and loops found is reported.

Value Returned

`i_count` Indicates the number of nets found with loops and/or dangles. This is the default output format.

`-1` The command did not run due to a syntax error.

Example

The following example looks for loops and dangles in nets in the selected set, reports the names of nets checked, and creates annotations for them that can be viewed in the Violation Browser.

```
check_connectivity -set [get_selection_set] -annotate -verbose
```

The following is an example of output to the Transcript area for this command.

```
Connectivity Check Summary:
Net Name                                  Loops     Dangles
TDSP_CORE_INST_opb#5b8#5d                     1           0
Processed 5 net(s) in 0.02 seconds (320.00 nets/second) with 4 success(es) and 1
failure(s) (1 loop(s), 0 dangle(s))
Elasped Time: 0.0 seconds    Current Memory Usage: 36.7 MB    Peak Memory Usage:
37.1 MB
1
```

Related Information

Tcl Command	update_net_connectivity verify_connectivity
Menu Command	*Verify—Connectivity*

check_density

```
check_density
     [ -lpp {s_layerlpp …} ]
     [ -output_lpp s_lpp ]
     [ -boundary {f_xlo f_ylo f_xhi f_yhi} ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -window_size {f_userunit [f_userunit]} ]
     [ -step_size {f_userunit [f_userunit]} ]
     [ -min_density f_percent ]
     [ -max_density f_percent ]
     [ -max_diff_density f_percent_0_to_1 ]
     [ -density_range {f_minpercent f_maxpercent} ]
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -blockage_density f_percent ]
     [ -boundary_interior_halo {f_x f_y} ]
     [ -boundary_interior_halo_min_density_hole_mult f_multiplier ]
     [ -boundary_interior_halo_target_density f_percent ]
     [ -no_sync ]
     [ -silent | -verbose ]
     [ -error_limit i_count ]
     [ -threads i_threads ]
```

Checks the metal density for a layer or layer purpose. You can check a specific region or the entire design in the active window. The area is checked one section at a time in partitions given by the `window_size` argument, beginning from the bottom-left of the area and moving across and up the area in `step_size` steps until the entire area has been checked. By default,

- The density for each partition (check window) is checked against the `minDensity` and `maxDensity` constraint values.
- The density difference between adjacent, non-overlapping partitions is checked against the `maxDiffDensity` constraint value.
- The window sizes/step sizes given by the density constraints are used.

The results are reported in the Transcript area for each layer checked. You can also create annotations for window sections that are not within the required densities and/or output the violating windows to a specific layer purpose.

If the `minDensityHole` constraint is set, this command will also check for holes whose effective width and area are larger than or equal to the minimum values defined by the constraint. The number of holes found will be included under the Violation Count heading in the Transcript area report. Polygon annotations can be created that outline the hole areas.

To identify holes that straddle two abutting blocks but do not qualify as holes within the individual blocks, use `-boundary_interior_halo` to specify a band around the block boundaries and choose a `minDensityHole` multiplier (`-boundary_interior_halo_min_density_hole_mult`) greater than 0 and less than 1.0 to recognize a smaller area as a hole within each block boundary.

To increase metal density, use create_fill.

Arguments

`-all`

When specified and by default, processes the entire design.

`-annotate [ true | false ]`

Chooses whether to create annotations to mark holes (if the `minDensityHole` constraint is set) and density check violations (sections that do not meet the density requirements). The annotations are added to the `annotation:violation` purpose of the given layers and are listed by layer under *Density-MinDensity Holes*, *Density-Max Density, Density-Min Density*, or *Max diff density violation* in the Violations page of the Annotation Browser. By default, annotations are created.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. The default is `1000`. A value of `-1` specifies no limit.

`-blockage_density` *`f_percent`*

Specifies the metal density to use for blockages. If a percentage of 0 is given, then blockages are treated as if there is no metal in the blockage area.

Default: target density

`-boundary` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Overrides the default behavior of determining the boundary used for checking density. The default is 1) Place and Route boundary and 2) occurrence bounds.

`-boundary_interior_halo` **{***`f_x f_y`***}**

Specifies a band (given by x and y offsets) inside the boundary in which a modified target density can be specified using `-boundary_interior_halo_target_density`, and/or a modified minimum hole area can be specified using `-boundary_interior_halo_min_density_hole_mult`.

`-boundary_interior_halo_min_density_hole_mult` *`f_multiplier`*

Multiplies the `minDensityHole` constraint by this value to determine the minimum hole area within the boundary halo given by `-boundary_interior_halo`. Value must be greater than 0 and less than 1. This lets you identify holes that are smaller than `minDensityHole` within the boundary halo.

`-boundary_interior_halo_target_density` *`f_percent`*

Specifies the target density percentage (0-100) to use within the boundary halo given by `-boundary_interior_halo`.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-density_range` **{***`f_minpercent f_maxpercent`***}**

Specifies the minimum and maximum density percentage to use as lower and upper limits. If not specified, the values from the technology database are used. If these values are not available, the command fails.

`-error_limit` *`i_count`*

Specifies the maximum number of errors to detect before stopping. By default, no error limit is set. A value of `-1` also specifies no limit.

`-lpp` **{***`s_layerlpp…`***}**

Specifies the layers to check.

`-max_density` *`f_percent`*

Specifies the maximum density percentage. Defaults to the `maxDensity` constraint value, if defined, otherwise 100%. Valid values: 0 to 100

`-max_diff_density` *`f_percent_0_to_1`*

Specifies the maximum density difference allowed between adjacent, non-overlapping check windows. Defaults to the `maxDiffDensity` constraint value, if defined. Must be between 0 and 1.

`-min_density` *`f_percent`*

Specifies the minimum density percentage. Defaults to the `minDensity` constraint value, if defined, otherwise 0%. Valid values: 0 to 100

`-no_sync`

Prevents the OpenAccess database from being updated.

`-output_lpp` *`s_lpp`*
Specifies the layer purpose to add violating window sections to.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**
Specifies the area to check, given the lower-left and upper-right coordinates. If not specified, the entire design in the active artwork window is checked.

`-silent`
When selected, turns off detailed messages during the check.

`-step_size` **{***`f_userunit`* `[`*`f_userunit`*`]`**}**
Specifies the window step as a single value for X and Y, or as two values ({X Y}). By default, one-half of the window size (`-window_size`) is used.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-verbose`
Outputs a message for each error found during the check.

`-window_size` **{***`f_userunit`* `[`*`f_userunit`*`]`**}**
Specifies the checking window size as a single value for X and Y, representing a square window, or as two values ({X Y}). By default, the window size given by the `minDensity` or `maxDensity` constraint is used.

Value Returned

`i_count`
Indicates the number of window sections found that do not meet the density criteria.

`-1`
The command did not run due to a syntax error or a missing required argument.

Example

The following example sets the window size to 200 microns (the step size defaults to one-half of the window size or 100 microns). The entire design in the active window will be checked. Annotations will be created for window sections with a `met1` density that is less than 25% or greater than 90%. The annotations will be listed in the Annotation Browser Violations page.

```
check_density -lpp met1 -density_range {25 90} -window_size 200
```

Example output from this command to the Transcript area:

```
      Window Window  Min      Max    Violation Count Density Percentage Stats  Std
Layer  Size   Step  Density Density   Min       Max  lowest highest median ave Dev
===== ====== ====== ======= ======= =============== ========================= ===
met1  200.00 100.00  25.0%   90.0%    127         0  0.00%  53.17%   2.49% 7.94% 2
127
```

Related Information

Tcl Command	create_fill enable_multithreading
Menu Command	*Verify—Metal Density*

check_discrete_width

```
check_discrete_width
     -lpp s_layerlpp
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -threads i_threads ]
     [ -top_level_only ]
     [ -trim_corners ]
     [ -verbose [ true | false ] ]
```

Checks the unbroken distances between opposing edges for *effective* (maximal) shapes on a layer. The width is considered to be the shorter dimension of a shape and is compared with discrete widths given in a one-dimensional table. A violation occurs if the measured width cannot be found in the table.

To use this command, you must first set the `discreteWidth` layer constraint, and you can optionally set the `length` parameter to exclude from checking any shapes shorter than the given length.

Arguments

`-annotate [all | none | dim | rect]`

Chooses whether to mark violations with annotations. The annotations are added to the `annotation:violation` purpose of the given layer and are listed by layer under *WidthChecks—Discrete Width Checks* in the Violations page of the Annotation Browser.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, up to 1000 annotations are created. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations` Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-lpp` *`s_layerlpp`* Specifies the list of layers and/or layer purpose pairs to check.

`-output_lpp` *`s_lpp`* Specifies the layer purpose pair to which violating shapes will be added.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`
Specifies that top-level shapes and shapes connected to top-level shapes be checked. By default, all shapes are checked.

`-trim_corners`
Specifies that corners be trimmed for checking.

`-verbose`
Outputs a message for each error found during the check.

Example

The following example limits discrete width checking to `Metal1` shapes that are longer than 3.5 user units. Violating shapes have widths (the shorter dimension) other than 0.20, 0.30, 0.4 or 1.00 user units.

```
set_constraint_parameter -name length -Value 3.5
set_layer_constraint -layer Metal1 -constraint discreteWidth -hardness hard
-OneDTblValue {0 0.20 1 0.30 2 0.4 3 1.00}
check_discrete_width -region [get_window_area] -lpp Metal1
Running discete width check
Metal1:ALL -- 2 (between 0 and 0.2 microns) discrete width violations found.
Metal1:ALL -- 0 (between 0.2 and 0.3 microns) discrete width violations found.
Metal1:ALL -- 0 (between 0.3 and 0.4 microns) discrete width violations found.
Metal1:ALL -- 0 (between 0.4 and 1.00 microns) discrete width violations found.
check_discrete_width completed in = 0.0s/0.0s/0.0s
2
```

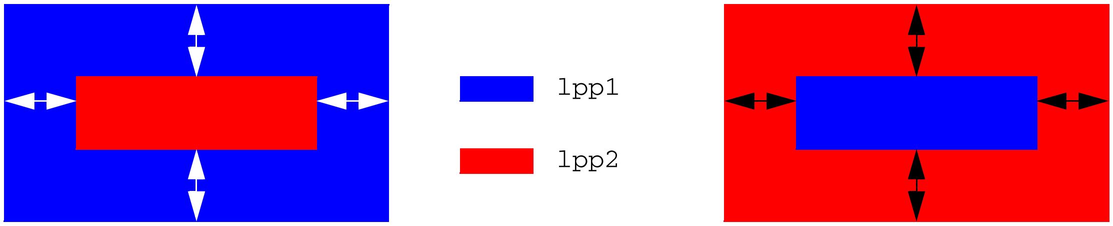
Related Information

Tcl Commands   `check_width`

check_extensions

```
check_extensions
     -lpp1 s_layerlpp
     -lpp2 s_layerlpp
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -top_level_only ]
     [ -error_limit i_count ]
     [ -silent | -verbose ]
     [ -trim_corners ]
     [ -min_bound_extent [f_userunit] ]
     [ -min_extent1 [f_userunit] ]
     [ -max_extent1 [f_userunit] ]
     [ -min_extent2 [f_userunit] ]
     [ -max_extent2 [f_userunit] ]
     [ -coincident_allowed1 [ true | false ] ]
     [ -coincident_allowed2 [ true | false ] ]
     [ -subtotals [ true | false ] ]
```

Checks enclosures by measuring the distance between the outside edge of one shape and the inside edge of another shape, or between the outside edge of a shape and the PRBoundary. Specify the appropriate argument for the check that you want to perform (`max_extent1`, `max_extent2`, `min_extent1`, `min_extent2`, `min_bound_extent`).



Checks the distance between the inner edge of `lpp1` beyond outer edge of `lpp2` by specifying `-min_extent1` and `-max_extent1`.

Checks the distance between the inner edge of `lpp2` beyond outer edge of `lpp1` by specifying `-min_extent2` and `-max_extent2`.

Arguments

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given layers and are listed by layer under *Minimum Extension* or *Maximum Extension* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-coincident_allowed1 [ true | false ]`

When set to `true`, allows coincident edges when checking the layer purpose given by `-lpp1` extending beyond the layer purpose given by `-lpp2`. Default is true,

`-coincident_allowed2 [ true | false ]`

When set to `true`, allows coincident edges when checking the layer purpose given by `-lpp2` extending beyond the layer purpose given by `-lpp1`.

`-error_limit` *`i_count`*

Specifies the maximum number of errors for each of `minExtension` and `maxExtension` to detect before stopping.

`-lpp1` *`s_layerlpp`*

Checks shapes on this layer (optionally restricted to a specified purpose). Defaults to the extended layer.

`-lpp2` *`s_layerlpp`*

Checks shapes on this layer (optionally restricted to a specified purpose). Defaults to the unextended layer.

`-max_extent1 [`*`f_userunit`*`]`

Checks the maximum extension rule of the layer purpose given by `-lpp1` beyond the layer purpose given by `-lpp2` be. If a value is given, it is used for the check instead of the `maxExtension` constraint.

`-max_extent2 [`*`f_userunit`*`]`

Checks the maximum extension rule of the layer purpose given by `-lpp2` beyond the layer purpose given by `-lpp1` be. If a value is given, it is used for the check instead of the `maxExtension` constraint.

`-min_bound_extent [`*`f_userunit`*`]`

Checks the minimum extension rule of the layer purpose given by `-lpp1` beyond the PRBoundary. If a value is given, it is used for the check instead of the `minBoundaryExtension` constraint.

`-min_extent1 [`*`f_userunit`*`]`

Checks the minimum extension rule of the layer purpose given by `-lpp1` beyond the layer purpose given by `-lpp2` be. If a value is given, it is used for the check instead of the `minExtension` constraint.

`-min_extent2 [`*`f_userunit`*`]`

Checks the minimum extension rule of the layer purpose given by `-lpp2` beyond the layer purpose given by `-lpp1` be. If a value is given, it is used for the check instead of the `minExtension` constraint.

`-output_lpp` *`s_lpp`*

Adds violating shapes to the specified output layer purpose.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-silent` Turns off detailed messages during checking.

`-subtotals [ true | false ]`

When set to `true`, outputs a Tcl list with violation counts reported separately (refer to “Processing Tcl Lists” on page 1018) in this order: `min_extent1`, `max_extent1`, `min_extent2`, and `max_extent2`. By default, the return value is the total number of violations found.

`-top_level_only` Specifies that only top-level shapes should be checked (against all levels). By default, all shapes are checked.

Note: Cell-to-cell placement errors will not be caught if this flag is set.

`-trim_corners` Specifies that corners be trimmed for checking.

`-verbose` Outputs a message for each error found during the check.

Value Returned

`i_count` Indicates the number of extension check failures found for the given criteria.

`i_min_extent1_errors`
`i_max_extent1_errors`
`i_min_extent2_errors`
`i_max_extent2_errors`

Tcl list with the number of `min_extent1`, `max_extent1`, `min_extent2`, and `max_extent2` violations found. This is the output format when `-subtotals` is given.

`-1` The command did not run due to a syntax error or a missing required argument.

Example

The following command reports occurrences of `Metal4` shapes extending less than 0.05 beyond `Metal3` shapes.

```
check_extensions -lpp1 Metal4 -lpp2 Metal3 -min_extent1 0.05
```

check_grid

```
check_grid
     -lpp s_layerlpp
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -no_sync ]
     [ -silent | -verbose ]
     [ -trim_corners ]
     [ -grid f_grid ]
     [ -error_limit i_count ]
     [ -check_mfg_grid [ true | false ] ]
     [ -check_routing_grid [ true | false ] ]
     [ -check_endpoints [ true | false ] ]
     [ -quantized_grids [ true | false ] ]
     [ -x_size f_userunit ]
     [ -y_size f_userunit ]
     [ -threads i_threads ]
     [ -top_level_only ]
```

Finds shapes that are off the manufacturing and/or the routing grid.

Arguments

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given layers and are listed by layer under *Grid Check—Manufacturing Grid* and *Routing Grid* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-check_endpoints [ true | false ]`

Checks the coordinates at each end of routing segments. Default is `false`.

`-check_mfg_grid [ true | false ]`

Checks for shapes off the manufacturing grid. Default is `true`.

`-check_routing_grid [ true | false ]`

Checks for shapes off the routing grid. Default is `true`.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-error_limit` *i_count*
Specifies the maximum number of errors to detect before stopping. By default, no error limit is set. A value of `-1` also specifies no limit.

`-grid` *f_grid*
Specifies an override value to use as the manufacturing grid. If this argument is not given, the manufacturing grid value from the technology file is used.

`-lpp` *s_layerlpp*
Checks shapes on this layer (optionally restricted to a specified purpose).

`-no_sync`
Prevents the OpenAccess database from being updated.

`-output_lpp` *s_lpp*
Adds violating shapes to the specified output layer purpose.

`-quantized_grids [ true | false ]`
When set `true` and the current routing mode is gridded (`set_route_on_grid -on_grid true`), the routing grid check will report all off-grid shapes. When set `false` and whenever the routing mode is gridless (`set_route_on_grid -on_grid false`), the routing grid check will not be performed. Default is `false`.

`-region` **{***f_xlo f_ylo f_xhi f_yhi***}**
Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-silent`
Turns off detailed messages during checking.

`-threads` *i_threads*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`
Checks only for top-level shapes to be on grid. By default, checks for shapes at all levels to be on grid.

`-trim_corners`
Specifies that corners be trimmed for checking.

`-verbose`
Outputs a message for each error found during the check.

`-x_size` *f_userunit*
Overrides the x-direction manufacturing grid for the check.

`-y_size` *f_userunit*
Overrides the y-direction manufacturing grid for the check.

Value Returned

`i_count`	Indicates the number of shapes found that are off-grid for the given criteria.
`-1`	The command did not run due to a syntax error or a missing required argument.

Related Information

Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

check_large_via_arrays

```
check_large_via_arrays
     -lpp s_layerlpp
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -top_level_only ]
     [ -annotate [ true | false ] ]
     [ -clear_annotations ]
     [ -annotation_limit i_count ]
     [ -error_limit i_count ]
     [ -check_mode [soft|hard] ]
     [ -no_sync ]
     [ -silent|-verbose ]
     [ -trim_corners ]
     [ -threads i_threads ]
```

Checks for existing `minLargeViaArraySpacing` constraint violations on the given layer or layer purpose pair.

Arguments

-annotate [true | false]

Chooses whether to mark violations with annotations. The annotations are added to the annotation:violation purpose of the given layers and are listed by layer under *Via Checks/Same Net Checks* in the Violations page of the Annotation Browser. By default, annotations are created.

-annotation_limit *i_count*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify -annotation_limit -1 to choose no limit.

-check_mode {soft|hard}

Chooses soft or hard constraint lookup. Default is hard.

-clear_annotations

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

-error_limit *i_count*

Specifies the maximum number of errors to detect before stopping. By default, no error limit is set. A value of -1 also specifies no limit.

-lpp *s_layerlpp*

Specifies the layer or layer purpose pair to be checked.

-no_sync

Prevents the output layer from being immediately synchronized to the OpenAccess database; it will be synchronized when the design is saved.

-output_lpp *s_lpp*

Specifies the optional output layer and purpose.

-region **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the boundary points for the area to be checked. If not specified, the entire design in the active artwork window is checked.

-silent

When specified, turns off all messages during the check.

-threads *i_threads*

Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`	Specifies that only vias associated with top-level geometry be checked (in other words, vias that are one level down from the top level). By default, all shapes are checked.
`-trim_corners`	Specifies that corners be trimmed for checking.
`-verbose`	Outputs a message for each error found during the check.

Value Returned

`i_count`	Is the number of large via array same net spacing violations found.
`-1`	The command did not run due to a syntax error or a missing required argument.

Example

The following commands set the required spacing for 2x2 cut arrays to 0.2 and for 4x4 cut arrays to 0.22 on Via1.

```
set_layer_constraint -layer Via1 -constraint minLargeViaArraySpacing \
 -hardness hard -row_name numCuts -OneDTblValue { 2 0.2 4 0.22 }
check_large_via_arrays -lpp Via1
```

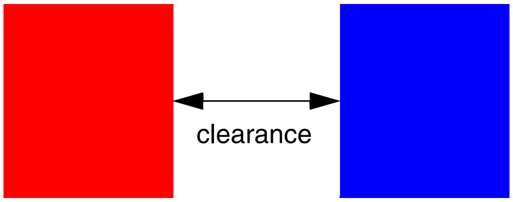
Related Information

Tcl Commands	check_space

check_layerpair_space

```
check_layerpair_space
     {[ -same_net ] [ -diff_net ]}
     -lpp1 s_layerlpp
     -lpp2 s_layerlpp
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -check_mode {hard | soft} ]
     [ -clear_annotations ]
     [ -allow_overlap [ true | false ] ]
     [ -allow_touching [ true | false ] ]
     [ -no_sync ]
     [ -process_rules_only [ true | false ] ]
     [ -silent|-verbose ]
     [ -space f_userunit ]
     [ -top_level_only ]
     [ -trim_corners ]
     [ -threads i_threads ]
```

Checks interlayer spacing of different shapes on different layers against the `minClearance` and/or the `minSameNetClearance` constraint for the two layers. Clearances are specified from the outside edge of a shape on the first layer to the outside edge of a shape on the second layer.



Arguments

`-allow_overlap [ true | false ]`

When set to `true`, spacing between shapes that overlap is not checked. This is useful to skip this check when the constraint is not set.

Default: `false`

`-allow_touching [ true | false ]`

When set to `true`, spacing between abutting shapes is not checked. This is useful to skip this check when the constraint is not set.

Default: `false`

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given `lpp1` layers and are listed by layer under *Layer to Layer Clearance Checks—Same Net Checks* and *Diff Net Checks* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-check_mode {soft|hard}`

Chooses soft or hard constraint lookup. Default is `hard`.

`-clear_annotations`
Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-diff_net`
Checks shapes on different nets and on the given layers against the `minClearance` constraint.

`-lpp1` *`s_layerlpp`*
Checks shapes on this layer (optionally restricted to a specified purpose) against shapes on the layer/purpose given by `-lpp2`.

`-lpp2` *`s_layerlpp`*
Checks shapes on this layer (optionally restricted to a specified purpose) against shapes on the layer/purpose given by `-lpp1`.

`-no_sync`
Prevents the OpenAccess database from being updated.

`-process_rules_only [ true | false ]`

When `true`, checking is performed against process rules constraints only.

Default: `false`

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-same_net`
Checks shapes on the same net and on the given layers against the `minSameNetClearance` constraint.

`-silent`
Turns off detailed messages during checking.

`-space` *`f_userunit`*
Specifies an override value to use for the clearance.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`
Specifies that only top-level shapes should be checked (against all levels). By default, all shapes are checked.

Note: Cell-to-cell placement errors will not be caught if this flag is set.

`-trim_corners`
Specifies that corners be trimmed for checking.

`-verbose`	Outputs a message for each error found during the check.

Value Returned

`i_count`	Indicates the number of shapes found that do not meet the `minClearance` or `minSameNetClearance` requirement.
`-1`	The command did not run due to a syntax error or a missing required argument.

Example

The following example sets the `minClearance` constraint for different net shapes on `poly` and `diff` layers, then checks for compliance using the `check_layerpair_space` command. Annotations are created to mark the violations.

```
set_layerpair_constraint -constraint minClearance -layer1 poly -layer2 diff
-Value .1 -group LEFDefaultRouteSpec -hardness hard
check_layerpair_space -diff_net -lpp1 poly -lpp2 diff
```

check_min_edge_length

```
check_min_edge_length
     -lpp {s_layerlpp…}
     [ -output_lpp s_layerlpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -error_limit i_count ]
     [ -no_sync ]
     [ -silent|-verbose ]
     [ -top_level_only ]
     [ -trim_corners ]
     [ -threads i_threads ]
```

Checks shapes for Optical Pattern Correction (OPC) suitability by determining whether each shape meets the following constraints: `minEdgeAdjacentDistance`, `minEdgeLength`, `minEdgeMaxCount`, and `oaMinEdgeAdjacentLength`. Only constraints that have been set will be checked.

Arguments

`-annotate [ true | false ]`

Chooses whether to mark violations with annotations. The annotations are added to the `annotation:violation` purpose of the given layers and are listed by layer under *Minimum Edge Length Check* in the Violations page of the Annotation Browser. By default, annotations are created.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-error_limit` *`i_count`*

Specifies the maximum number of errors to detect before stopping. By default, no error limit is set. A value of `-1` also specifies no limit.

`-lpp` **{***`s_layerlpp…`***}**

Specifies the list of layers and/or layer purpose pairs to check.

`-no_sync`

Prevents the OpenAccess database from being updated.

`-output_lpp` *`s_layerlpp`*

Specifies the layer purpose to add the checking results to. Shapes that fail to meet the minimum edge length requirements are added.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-silent`

When selected, turns off all messages during the check.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

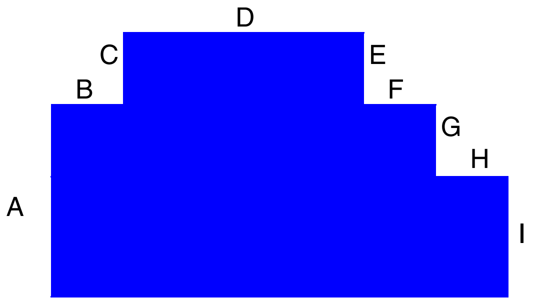
`-top_level_only`	Specifies that top-level shapes and shapes connected to top-level shapes be checked. By default, all shapes are checked.
`-trim_corners`	Specifies that corners be trimmed for checking.
`-verbose`	Outputs a message for each error found during the check.

Value Returned

`i_count`	Indicates the number of shapes that violate the rule for the given criteria.
`-1`	The command did not run due to a syntax error or a missing required argument.

Example

The following figure is used for the examples in this section.



Case 1—Using `minEdgeMaxCount`

```
set_constraint_parameter -name maxLength -Value 0.4
set_layer_constraint -layer M2 -constraint minEdgeMaxCount -IntValue 2
check_min_edge_length -lpp M2 -clear_annotations
```

Flags any sequence of more than two segments that are less than 0.4 in length. For example, if segments B, C, E, F, G and H are all less than 0.4 in length, then B and C are not violating since there are only two consecutive short segments. However, E-G and F-H are in violation because there are more than two consecutive short segments.

Case 2—Using `minEdgeAdjacentDistance`

```
set_constraint_parameter -name length -Value 0.4
set_constraint_parameter -name count -IntValue 2
set_layer_constraint -layer M2 -constraint minEdgeAdjacentDistance -Value 1.0
check_min_edge_length -lpp M2 -clear_annotations
```

Flags a violation if the length of segment D is less than 1.0 and at least segments B, C, E, and F are all less than 0.4 in length. For this example, the rule is triggered when there are two sets of edges, each with at least two (`count`) consecutive edges that are less than or equal to `length` in length.

Case 3—Using `minEdgeLength`

```
set_layer_constraint -layer M2 -constraint minEdgeLength -Value 0.4
check_min_edge_length -lpp M2 -clear_annotations
```

Flags an error if any edge is less than 0.4 in length.

Case 4—Using minEdgeLength with lengthsum parameter

```
set_constraint_parameter -name lengthSum -Value 1.0
set_layer_constraint -layer M2 -constraint minEdgeLength -Value 0.4
check_min_edge_length -lpp M2 -clear_annotations
```

Flags the same violations as case 3. However, if the sum of the consecutive segments that are less the 0.4 in length is less than 1.0, then there is no violation.

Case 5—Using `oaMinEdgeAdjacentLength`

```
set_constraint_parameter -name maxLength -Value 0.4
set_layer_constraint -layer M2 -constraint oaMinEdgeAdjacentLength -Value 1.0
check_min_edge_length -lpp M2 -clear_annotations
```

Flags a violation if any minimum length edge (length less than or equal to 0.4) is adjacent to an edge that is less than 1.0 in length. In the example, all shorter edges would be in violations.

Related Information

Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

check_minarea

```
check_minarea
     -lpp s_layerlpp
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -area f_userunit ]
     [ -check_mode [ true | false ] ]
     [ -clear_annotations ]
     [ -enclosed ]
     [ -error_limit i_count ]
     [ -exclude_pins [ none | all | toplevelonly | lowlevelonly ]
     [ -no_sync ]
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -silent|-verbose ]
     [ -threads i_threads ]
     [ -top_level_only ]
     [ -trim_corners ]
```

Finds all shapes that have an area that is less than the minimum area rule or finds non-metal spaces that are fully-enclosed by metal and are smaller than the minimum enclosed area requirement.

Arguments

`-annotate [ true | false ]`

Chooses whether to mark violations (shapes that do not meet the minimum area requirements) with annotations. The annotations are added to the `annotation:violation` purpose of the given layer and are listed by layer under *Minimum Area Check* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-area` *`f_userunit`*

Specifies an override value to use as the minimum area rule. If this argument is not given, the minimum area rule from the technology file is used.

`-check_mode {soft|hard}`

Chooses soft or hard constraint lookup. Default is `hard`.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-enclosed`

Searches for non-metal areas that are fully-enclosed by metal on the given layer purpose (`-lpp`) and are smaller than the minimum enclosed area given in the technology file. By default, if this argument is not given, or is set to `false`, this command searches for shapes in the given layer that are smaller than the minimum area rule.

`-error_limit` *`i_count`*

Specifies the maximum number of errors to detect before stopping. By default, no error limit is set. A value of `-1` also specifies no limit.

`-exclude_pins [none|all|toplevelonly|lowlevelonly]`

Indicates whether violations on pin shapes should be filtered.

Default: Pin shape violations are not filtered.

`-lpp` *`s_layerlpp`*	Specifies the layer (optionally restricted to a specific purpose) to use for the check.
`-no_sync`	Prevents the OpenAccess database from being updated.
`-output_lpp` *`s_lpp`*	Adds violating shapes to the specified output layer purpose.
`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**	
	Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.
`-silent`	Turns off detailed messages during checking.
`-threads` *`i_threads`*	Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.
`-top_level_only`	Restricts checks to shapes and areas associated with the top level geometry. By default, all levels are checked.
`-trim_corners`	Specifies that corners be trimmed for checking.
`-verbose`	Outputs a message for each error found during the check.

Value Returned

`i_count`	Indicates the number of shapes found that are smaller than the minimum area.
`-1`	The command did not run due to a syntax error or a missing required argument.

Example

The following example checks the entire design for `METAL1` shapes and adds rectangle annotations to the design for shapes that are smaller than the `METAL1` minimum area rule in the technology file.

```
check_minarea -lpp METAL1
```

The following example checks a region for areas that are fully-enclosed by `METAL1` shapes and are smaller than the `METAL1` minimum enclosed area rule in the technology file, and adds rectangle annotations only for those areas.

```
check_minarea -lpp METAL1 -enclosed -region { 1000.2 1120.4 1012.4 1131.3 }
-annotate rect
```

Related Information

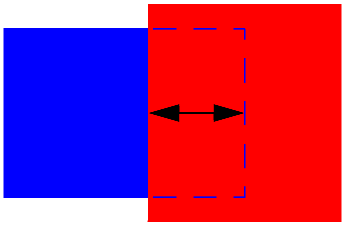
Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

check_overlap

```
check_overlap
     -lpp1 s_layerlpp
     -lpp2 s_layerlpp
     [ -min_overlap1 [f_userunit] ]
     [ -min_overlap2 [f_userunit] ]
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -top_level_only ]
     [ -error_limit i_count ]
     [ -silent | -verbose ]
     [ -trim_corners ]
     [ -subtotals [ true | false ] ]
```

Checks the spacing between an inside edge of a shape on one layer to an inside edge of a shape on another layer.

The `check_overlap` command can check the overlap of the two shapes in the following figure.



Arguments

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given `lpp1` layers and are listed by layer under *Minimum Overlap* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations` Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-error_limit` *`i_count`* Specifies the maximum number of errors of each type to detect before stopping. By default, no error limit is set. A value of `-1` also specifies no limit.

`-lpp1` *`s_layerlpp`* Checks shapes on this layer (optionally restricted to a specified purpose) against shapes on the layer/purpose given by `-lpp2`.

`-lpp2` *`s_layerlpp`* Checks shapes on this layer (optionally restricted to a specified purpose) against shapes on the layer/purpose given by `-lpp1`.

`-min_overlap1` *`f_userunit`*

Overrides the minimum overlap rule value for layer1 over layer2.

`-min_overlap2` *`f_userunit`*

Overrides the minimum overlap rule value for layer2 over layer1.

`-output_lpp` *`s_lpp`*
Adds violating shapes to the specified output layer purpose.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-silent`
Turns off detailed messages during checking.

`-subtotals`
Outputs a Tcl list with the number of `min_overlap1` violations and the number of `min_overlap2` violations reported separately (refer to “Processing Tcl Lists” on page 1018). By default, the return value is the total number of violations found.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`
Specifies that only top-level shapes should be checked (against all levels). By default, all shapes are checked.

Note: Cell-to-cell placement errors will not be caught if this flag is set.

`-trim_corners`
Specifies that corners be trimmed for checking.

`-verbose`
Outputs a message for each error found during the check.

Value Returned

i_count	Indicates the number of shapes found that do not meet the `minClearance` or `minSameNetClearance` requirement.
i_min_overlap1_errors *i_min_overlap2_errors*	Tcl list with the number of `min_overlap1` and `min_overlap2` violations found. This is the output format when `-subtotals` is given.
-1	The command did not run due to a syntax error or a missing required argument.

Example

The following command reports violations when the overlap of `Metal2` over `Metal3` shapes is less than `0.08.`

```
check_overlap -lpp1 Metal2 -lpp2 Metal3 -min_overlap1 0.08
```

check_routability

```
check_routability
     [ -net {s_netName…} ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -clear_annotations ]
     [ -error_limit i_count ]
     [ -no_blockage_check ]
     [ -no_coverobs_check false ]
     [ -no_direction_check ]
     [ -no_floating_shape_check ]
     [ -no_grid_check ]
     [ -no_min_space_check ]
     [ -no_min_width_check ]
     [ -no_out_of_bndry_check ]
     [ -no_pin_center_on_track_check ]
     [ -no_pin_on_grid_check ]
     [ -no_pin_on_track_check ]
     [ -no_prbndry_check ]
     [ -no_via_check ]
```

Checks the active design for known conditions that can cause potential routing problems. Design Rule Checks applied to pins and some routability checks are performed primarily to determine the accessibility of pins. Additional routability checks address special issues, such as the presence of differential pair nets. By default, missing cover obstruction checks are excluded and must be enabled using `-no_coverobs_check false`. All other checks are enabled by default and can be excluded by specifying its respective argument when this command is invoked. Annotations are automatically created for violations, except cover obstruction and PR boundary violations.

If violations are reported, use the Violations Browser (*Collaborate—Annotation Browser—Violations*) to locate them.

The following table describes the checks that are performed and how they are categorized in the Violations Browser.

Check	Severity	Violations Browser
Pin-to-pin spacing less than minimum spacing	Error	*Clearance Checks—Diff Net Checks*
Pin dimensions less than minimum width	Error	*Width Checks—Minimum Width Checks*

Check	Severity	Violations Browser
Pin origin not on manufacturing grid	Warning	*Grid Check—Manufacturing Grid*
Pins blocked by top-level blockage or blockage of another instance	Error	*Routability Checks—Blockage Checks*
Pins that cannot be escaped using available vias	Error	*Routability Checks—Via Checks*
Floating shapes on wire:detail, wire:pin on top of wire pin shapes	Warning	*Routability Checks—FloatShape Checks*
Diffusion shapes on drawing purpose	Warning	*Routability Checks—DiffShape Checks*
Diff pair nets if present in the design, as these will be skipped during the regular flow	Warning	*Routability Checks—DiffPairNets Checks*

A Routability Report is output to the Transcript area that summarizes the violations that are found. For example,

```
+-------------------------------------------------+
| Routability Report Summary:                     |
+-------------------------------------------------+
|          0 Min Spacing violation(s)             |
|          0 Min width violation(s)               |
|          0 Grid violation(s)                    |
|         10 Via violation(s)                     |
|          0 Blockage violation(s)                |
|          0 Floating shape violation(s)          |
|          0 Diffusion shape violation(s)         |
|          2 Diff pair nets violation(s)          |
+-------------------------------------------------+
|         56 pin(s) found                         |
|         56 pin(s) checked                       |
|          0 pin(s) already connected             |
|         12 pin(s) not used (skipped)            |
+-------------------------------------------------+
```

More severe issues, such as DRC violations, should be dealt with before attempting to use the data.

Diffusion shapes on the `drawing` purpose will not be seen by Space-based Router and Chip Optimizer. To be recognized and processed, they must be mapped to either `diffusion:blockage` or `diffusion:pin` shapes using map_purpose.

Differential pair nets are not routed in the regular routing flow and require special attention. For more information on routing these nets, refer to “Pair Routing Commands” on page 607.

Arguments

-clear_annotations
Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

-error_limit *i_count*
Specifies the maximum number of annotations to create for each type of check.

-exclude_net **{***s_netName…***}**

Excludes the given nets from the pin blockage checks.

-exclude_type {[power][ground][clock]}

Excludes nets of the given types from processing.

-net **{***s_netName…***}**
Limits processing to the nets in the list. By default, all nets are processed.

-no_blockage_check
Disables blockage checks.

-no_coverobs_check false

Enables checks for instances without a cover obstruction. By default, these checks are disabled.

-no_direction_check
Disables pin direction checks.

-no_floating_shape_check

Disables floating shape checks.

-no_grid_check
Disables grid checks.

-no_min_space_check
Disables minimum spacing checks.

-no_min_width_check
Disables minimum width checks.

-no_out_of_bndry_check

-
Disables pin out of boundary checks.

-no_pin_center_on_track_check

Disables checks for pin centers on tracks.

-no_pin_on_grid_check
Disables checks for pin centers on grid.

-no_pin_on_track_check

Disables checks for pins on tracks.

-no_prbndry_check
Disables checks for instances without a PR boundary.

`-no_via_check` Disables via checks.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

check_route_quality

```
check_route_quality
     {-all | -net s_netName | -set d_setObj}
     -checks {[wrongWay][crossing][hook][portShort][ruleChange ][closeVia]
        [routeSpecWidth][routeSpecLayer][routeSpecVia]
        [preferredLayer][softFence ][taperFuse ][robustPinConnection][softSpace]
     }
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -error_set d_setObj ]
     [ -ignore_nets {s_netName…} ]
     [ -ignore_power_ground [ true | false ] ]
     [ -ignore_route_shape_crossing_factor f_factor ]
     [ -ignore_segment_in_pin [ true | false ] ]
     [ -ignore_segment_in_via [ true | false ] ]
     [ -ignore_crossing_threshold f_userunit ]
     [ -ignore_route_shape_crossing_factor f_userunit ]
     [ -centerline [ true | false ] ]
     [ -taper_threshold f_userunit ]
     [ -taper_threshold_default f_userunit ]
     [ -use_term_taper [ true | false ] ]
     [ -wrong_way_distance_threshold f_userunit ]
     [ -ignore_route_segments [ true | false ] ]
     [ -hook_ratio_threshold f_userunit ]
     [ -via_proximity_threshold f_userunit ]
     [ -hard [ true | false ] ]
```

Performs one or more route quality checks on all nets, a given net, or nets in a set.

Arguments

`-all`

Operates on the entire design in the active artwork window.

`-annotate [ true | false ]`

Chooses whether to mark route quality problems with annotations. By default, annotations are added. Existing route quality annotations are automatically cleared for the selected checks when this command is run. New annotations are added to the `annotation:violation` purpose of the layer and are listed in the Violations page of the Annotation Browser, under the name of the check type, followed by the name of the net.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-centerline [ true | false ]`

(Applies only to port short and crossing checks) Determines whether the centerline (*stick*) representation or the physical representation is checked. By default, route segments are considered crossed if their centerlines cross.

If set `false`, the physical intersection of the shapes is checked. For example, a crossing at a corner can flag a violation, even when the centerlines do not cross.

`-checks` **{***`s_checkType`***…}**

Specifies the route quality checks (*`s_checkType`*) to perform:

`closeVia` Checks for two vias that are closer than the `via_proximity_threshold` distance.

`crossing` Checks for a crossing of two non-adjacent shapes on the same net. Shapes are adjacent if they are neighbors in the same route, if they intersect at a Steiner point, or if they intersect at the same port of a pin.

`hook` Checks the path distance between two shapes and flags a violation when the path distance greatly exceeds the physical distance between them. The path distance between two points is the sum of the lengths of the route segments connecting the two points.

`portShort` Is a special type of crossing involving a route segment or via shape and a port on a pin.

`preferredLayer` Occurs when a net does not adhere to its preferred layer rule. If no rule exists, the check is not performed. If a net has a preferred layer rule, then any wire that is not on a preferred layer is a violation. An exception is made if the violating wire is sufficiently close to a pin, and might be needed for pin access.

`robustPinConnection`

Checks for the length of the diagonal of the intersection rectangle between a wire or via and a pin. This must be greater than or equal to either:

- The width of the wire/via
- The longest edge of the pin

`routeSpecLayer` Occurs when a route segment on a layer is not in the route’s route spec (assigned route spec or inherited) as determined by the `validRoutingLayers` constraint.

`routeSpecVia` Checks for route vias that are not in the route's route spec (assigned route spec or inherited) as determined by the `validRoutingVias` constraint.

`routeSpecWidth` Checks for segments with a width that does not match the route's route spec (assigned route spec or inherited) as determined by the `minWidth` constraint.

`ruleChange` Occurs when the routing rule for a section of a net does not conform to the net's routing rule.

`softFence` Checks for nets with routes within a fence boundary without any terminals in it. If all terminals of a net are within a fence boundary, the entire net must be routed within the fence boundary.

`softSpace` Checks for soft spacing rule adherence by layer and by net and reports the percentage of adherence.

`taperFuse` Checks for terminals with more than one taper route, or steiner points with more than two taper routes. These are not necessarily fuses but they do have the potential to be due to tapering.

`wrongWay` Checks for wires whose orientation is perpendicular to the preferred routing direction of the layer. This is not illegal, but it can block routing resources on the layer.

`-error_set` *`d_setObj`* Specifies the set to add nets with route quality problems to.

`-hard [ true | false ]` Applies when the `routeSpecWidth` check is performed. When set to `true`, segment widths are compared with the *hard* width value, rather than the preferred (soft) rule.

`-hook_ratio_threshold` *`f_distance`*

(Applies only to hook checks) Ignores hooks with a path distance/physical distance less than this value. By default, a value of 3.0 user units is used.

`-ignore_crossing_threshold` *`f_distance`*

(Applies only to crossing checks) Ignores crossings with minimum intersection dimension less than the threshold value. The default is 0.0.

`-ignore_nets` **{***`s_netName`…***}**

Excludes the specified nets from processing. This argument is ignored if `-net` is given.

`-ignore_power_ground [ true | false ]`

Excludes power and ground nets from processing. This argument is ignored if `-net` is given. By default, power and ground nets are excluded.

`-ignore_route_segments [ true | false ]`

Ignores power/ground route segments when checking crossings or port shorts. Default is `false`.

`-ignore_route_shape_crossing` *`f_factor`*

Ignores crossings where both dimensions of the crossing intersection are less than the product of the factor and the minimum width of the crossing's layer. The default is 0.0.

`-ignore_segment_in_pin` (Applies only to crossing checks) Ignores segments entirely embedded within metal of a pin. The default is `false`.

`-ignore_segment_in_via` (Applies only to crossing checks) Ignores segments entirely embedded within metal of a via. The default is `false`.

`-net` *`s_netName`* Specifies the name of the net to check.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Limits the check to the area given by the boundary points. By default, the `-all`, `-set` or `-net` argument determines the nets to check, with no boundary limits.

`-set` *`d_setObj`* Operates only on the nets in the given set.

`-taper_threshold` *`f_userunit`*

(Applies only to rule change checks) If the threshold is set, then route segments that connect directly with a pin and have a length less than or equal to the given taper threshold are considered to be allowed *pin tapers* and will not be flagged by the rule change check. By default, the taper threshold is set to 0.0.

`-taper_threshold_default` *`f_userunit`*

Sets the default taper length around terminals for the preferredLayer check. Defaults to 0.

`-use_term_taper [ true | false ]`

Uses the terminal's maxTaperWindow constraint for the preferredLayer check. The default is `false`.

`-via_proximity_threshold` *`f_distance`*

Specifies the distance from a via to check for the presence of another via for a `closeVia` check. Defaults to 0.

`-window_based_taper_check`

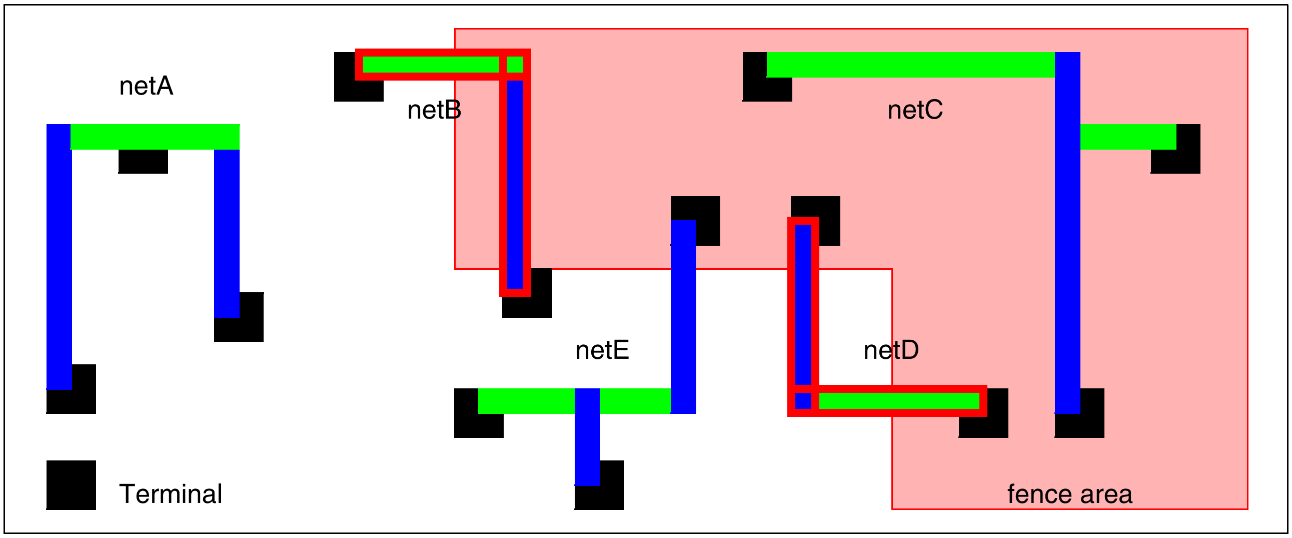
(Applies only to rule change checks) Routes connected to a pin within the taper window (`maxTaperWindow`) must have the same rulespec as the pin. Default is true.

`-wrong_way_distance_threshold` *`f_distance`*

Ignores wrong-way segments that are shorter in length than this value. Defaults to 0.

Example

The following are examples with soft fences. Of the five nets, nets netB and netD have soft fence violations. netA is wholly outside of the fence area. netC is wholly inside the fence area. netE crosses into the fence area to connect to a terminal. netB is in violation because both terminals are outside the fence area and routing goes into the fence area. To correct the violation, netB must be routed entirely outside the fence area. netD is in violation because both terminals are inside the fence area and routing goes outside the fence area. To correct the violation, netD must be routed entirely inside the fence area.



Value Returned

i_count Indicates the number of nets checked for the given criteria.

`-1` The command did not run due to a syntax error or a missing required argument.

check_space

```
check_space
     {[ -same_net ] [ -diff_net ]}
     -lpp1 s_layerlpp
     [ -lpp2 s_layerlpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -top_level_only ]
     [ -annotate [all | none | dim | rect] ]
     [ -clear_annotations ]
     [ -annotation_limit i_count ]
     [ -no_sync ]
     [ -check_mode {soft|hard} ]
     [ -crossing ]
     [ -process_rules_only ]
     [ -silent | -verbose ]
     [ -trim_corners ]
     [ -threads i_count ]
     [ -space f_userunit ]
     [ -check_trim [ true | false ] ]
```

Checks shapes for spacing design rule violations. You can specify the area to be checked and choose which shapes to include:

- An entire physical layer at a time
- A single layer purpose against an entire physical layer
- A single layer purpose against a single layer purpose (same or different nets)

By default, Space-based Router and Chip Optimizer creates annotations for violations found and outputs a summary report to the Transcript area. Two types of annotations can be created for each violation:

- The dimension annotation shows the measurement of the spacing violation.
- The rectangle annotation surrounds the violating spacing area.

Arguments

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations.

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given layer and are listed by layer under *Clearance Checks—Same Net Checks* or *Clearance Checks—Diff Net Checks* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-check_mode {soft|hard}`

Chooses soft or hard constraint lookup. Default is `hard`.

`-check_trim [ true | false ]`

By default and when set to `true`, checks trims. When set to `false`, no trim checking is done.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-crossing`

Enables same net crossing checks. By default, same net crossings checks are not performed.

`-diff_net`

Checks spacing between different nets.

`-lpp1` *`s_layerlpp`*

Checks shapes on this layer (optionally restricted to a specified purpose).

`-lpp2` *`s_layerlpp`*

Checks shapes on the layer specified by the `-lpp1` argument against routing shapes on this layer (optionally restricted to a specified purpose).

`-no_sync`

Prevents the OpenAccess database from being updated.

`-process_rules_only`

Uses only foundry constraints (process rules) for checking. By default, the appropriate route spec constraints are used.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-same_net`

Checks spacing between shapes on the same net.

`-silent`

Turns off detailed messages during checking.

`-space` *`f_userunit`*

Specifies an override value to use as the minimum spacing rule. If this argument is not given, the minimum spacing rule from the technology file is used.

`-threads` *`i_count`*

Specifies the number of threads, or processors, to use to process this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, a single processor is used.

`-top_level_only`

Specifies that only top-level shapes be checked against all levels. By default, all shapes are checked.

Important

> Cell-to-cell placement errors will not be detected if this flag is set.

`-trim_corners`

Specifies that corners be trimmed for checking.

`-verbose`

Outputs a message for each error found during the check.

Value Returned

i_count	Indicates the number of spacing rule violations found for the given criteria.
-1	The command did not run due to a syntax error or a missing required argument.

Related Information

Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

check_strong_weak

```
check_strong_weak
     {-all | -net s_netName | -set d_setObj}
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -error_set d_setObj ]
     [ -exclude_net {s_netName…} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
```

Checks for pin shapes that are weakly connected. The scope of the check can be the entire design, a given net, or a given set. The condition occurs when two pin shapes of an instance each terminate a route and the pin shapes are only connected through a high resistance material.

Arguments

`-all` Operates on the entire design in the active artwork window.

`-annotate [ true | false ]`

Chooses whether to mark weakly connected pin shapes with annotations. The annotations are added to the `annotation:violation` purpose of the pin layer and are listed by pin under *Check Connectivity—Strong/weak violation* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-error_set` *`d_setObj`* Specifies the set to add nets with weak connects to.

`-exclude_net` **{***`s_netName`*…**}**

Excludes the specified nets from processing. This argument is ignored if `-net` is given.

`-exclude_set` *`d_setObj`* Excludes nets in the given set from processing.

`-exclude_type` **{**`[power][ground][clock]`**}**

Excludes nets of the specified types from processing.

`-net` *`s_netName`* Checks strong/weak connectivity on the specified net.

`-set` *`d_setObj`* Checks strong/weak connectivity on the selected structures.

Value Returned

`i_count` Indicates the number of weakly connected pins found.

`-1` The command did not run due to a syntax error or a missing required argument.

Example

The following example runs the strong/weak analysis on the entire design.

```
check_strong_weak -all
```

Related Information

Menu Command	*Verify—Connectivity*

check_vertex_spacing

```
check_vertex_spacing
     -type [inner | outer]
     -lpp1 s_layerlpp
     -lpp2 s_layerlpp
     [ -overlapLpp s_layerlpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [ true | false ] ]
     [ -clear_annotations ]
     [ -annotation_limit i_count ]
     [ -top_level_only [ true | false ] ]
     [ -verbose [ true | false ] ]
     [ -silent [ true | false ] ]
     [ -trim_corners [ true | false ] ]
```

Checks for `minInnerVertexSpacing` and `minOuterVertexSpacing` violations.

Arguments

`-annotate [all|none ]`

Chooses whether to mark violations with annotations. The annotations are added to the `annotation:violation` purpose of the given lpp1 and are listed by layer under *Minimum inner vertex spacing* and *Minimum outer vertex spacing* in the Violations page of the Annotation Browser. By default, all annotations are created.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations` Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-lpp1` *`s_layerlpp`* Specifies the layer or layer purpose pair for the first layer.

`-lpp2` *`s_layerlpp`* Specifies the layer or layer purpose pair for the second layer.

`-overlapLpp` *`s_layerlpp`*

(Used only when `-type inner`) Specifies the layer or layer purpose pair overlapping with the other two lpps.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Specifies the boundary points for the area to be checked. If not specified, the entire design in the active artwork window is checked.

`-silent [ true | false ]`

When `true`, turns off all messages during the check. Default: `false`

`-top_level_only` Specifies that only vias associated with top-level geometry be checked (in other words, vias that are one level down from the top level). By default, all shapes are checked.

`-trim_corners` Specifies that corners be trimmed for checking. Default: `false`

`-type {inner|outer}`	Specifies whether to check inner (three-layer) or outer (two-layer) vertex spacing.
`-verbose [ true | false ]`	
	When `true`, outputs a message for each error found during the check. Default: `false`

Value Returned

`i_count`	Is the number of inner or outer vertex spacing violations found.
`-1`	The command did not run due to a syntax error or a missing required argument.

Example

The following example sets `minInnerVertexSpacing` and `minOuterVertexSpacing` constraints and checks for violations.

```
set_layerarray_constraint -constraint minInnerVertexSpacing \
  -layer_array {Metal1 Metal2 Metal3} -Value 2.0
check_vertex_spacing -type inner -lpp1 Metal1 -lpp2 Metal2 -overlapLpp Metal3

set_layerpair_constraint -constraint minOuterVertexSpacing -layer1 Metal1 \
  -layer2 Metal2 -Value 0.7 -symmetric false -create
check_vertex_spacing -type outer -lpp1 Metal1 -lpp2 Metal2 -annotation 10
```

check_vias

```
check_vias
     -lpp {s_layerlpp…}
     [ -stack_check ]
     [ -numcuts_check ]
     [ -extension_check ]
     [ -stacklimit_check ]
     [ -inter_layer_check ]
     [ -include_pins [ true | false ] ]
     [ -output_lpp s_lpp ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -top_level_only ]
     [ -annotate [ true | false ] ]
     [ -clear_annotations ]
     [ -annotation_limit i_count ]
     [ -error_limit i_count ]
     [ -no_sync ]
     [ -silent | -verbose ]
     [ -threads i_threads ]
     [ -check_mode [soft|hard] ]
     [ -trim_corners ]
```

Checks for via violations in stacking, numcuts and extensions, in the entire active design or a specific region. Violating shapes can be output and/or annotated.

Arguments

`-annotate [ true | false ]`

Specifies whether to mark violations. The annotations are added to the `annotation:violation` purpose of the given via layer and are listed by the check name and the layer under *Via Checks* in the Violations page of the Annotation Browse. By default, annotations are added.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-check_mode {soft|hard}`

Chooses soft or hard constraint lookup. Default is `hard`.

`-clear_annotations`
Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-error_limit` *`i_count`*
Specifies the maximum number of errors to detect before stopping. By default, there is no limit. A value of -1 chooses no limit.

`-extension_check`
Checks for via extension (via enclosure) violations (`minExtension`, `minDualExtension`, `minRedundantViaSetback`, `minCenterLineExtension`). Default is `false`.

`-include_pins [ true | false ]`

Specifies whether via pin shapes are checked. Default is `true`.

`-inter_layer_check`
Invokes via to via interlayer checks. Default is `false`.

`-lpp` **{***`s_layerlpp…`***}**
Specifies the via layers or layer purposes to check.

`-no_sync`
Prevents the OpenAccess database from being updated with the output shapes.

`-numcuts_check`
Causes via objects and via instances to be checked for the minimum number of cuts required when connecting between two wide wire shapes or when connecting a wide shape to a pin. Default is `false`.

`-output_lpp` *`s_lpp`*
Adds violating shapes to the specified output layer purpose.

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**
Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

`-silent`
Turns off detailed messages during checking.

`-stack_check`
Causes via objects to be checked for stackability with other via objects. Works in conjunction with set_via_stackability. Default is `false`.

`-stacklimit_check`
Checks cut layers against the via stack limit given by the `viaStackLimit` constraint. If the via stack limit is *`k`*, at most *`k`* consecutive cut layers may be aligned, otherwise a violation results for every intersection of *`k`*+1 cuts that are aligned. Default is `false`.

For an example, refer to “Via Stack Limit Check” on page 1012.

`-threads` *`i_threads`*
Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`
Restricts checks to shapes and areas associated with the top level geometry. By default, all levels are checked.

`-trim_corners`
Specifies that corners be trimmed for checking.

`-verbose`
Outputs a message for each error found during the check.

Value Returned

i_count Indicates the number of violating vias found.

`-1` The command did not run due to a syntax error or a missing required argument.

Example

The following example checks for numcut, via stack, and extension violations for vias on layers V2, V3 and V1 in the area currently being viewed in the active window and shows the results.

```
check_vias -numcuts_check -stack_check -extension_check -region [get_window_area]
-lpp {V2 V3 V1}
Running check_vias
V2:ALL -- 0 Stacked via violations found
V2:ALL -- 0 Numcut Via violations found
V2:ALL -- 16 Via Extension violations found
Running check_vias
V3:ALL -- 0 Stacked via violations found
V3:ALL -- 0 Numcut Via violations found
V3:ALL -- 0 Via Extension violations found
Running check_vias
V1:ALL -- 6 Stacked via violations found
V1:ALL -- 1 Numcut Via violations found
V1:ALL -- 0 Via Extension violations found
check_vias completed in 0.1s (elapsed), using 0.0s/0.0s0.1s
(user/kernel/total CPU).
23
```

Via Stack Limit Check

The following commands set the maximum stack to two consecutive vias for cut layers `VIA1`, `VIA2`, `VIA3`, and `VIA4`.

```
set_constraint_parameter -name lowerLayer -LayerValue VIA1
set_constraint_parameter -name upperLayer -LayerValue VIA4
set_constraont -constraint viaStackLimit -IntValue 2
```

If the optional parameters are not specified, the stack limit applies to the lowest cut layer containing shapes up to the top-most cut layer. If the `lowerLayer` parameter is specified as a non-cut layer, it will be snapped up to the closest cut layer containing shapes. If the `upperLayer` parameter is specified as a non-cut layer, it will be snapped down to the closest cut layer. If there are a total of less than *k*+1 layers where the stack limit is *k*, then no stack limit is applied. The following command will flag a violation if vias are stacked on more than two consecutive cut layers (for example, `VIA1`, `VIA2,` and `VIA3`, or `VIA2`, `VIA3`, and `VIA4)`.

```
check_vias -stacklimit_check
```

Related Information

Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

check_width

```
check_width
     -lpp s_layerlpp
     [ -output_lpp s_layerlpp ]
     [ -width f_userunit ]
     [ -max_width [f_userunit] ]
     [ -object_width [ true | false ] ]
     [ -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -annotate [all | none | dim | rect] ]
     [ -annotation_limit i_count ]
     [ -clear_annotations ]
     [ -diagonal_aware [ true | false ] ]
     [ -discrete_width [ true | false ] ]
     [ -lpp_width [ true | false ]
     [ -no_sync ]
     [ -silent | -verbose ]
     [ -top_level_only ]
     [ -trim_corners ]
     [ -threads i_threads ]
     [ -subtotals ]
```

Checks shapes for minimum width, maximum width, and discrete width violations. By default, all width checks are run if the respective constraints are set, violation annotations are created, and a summary report is output to the Transcript area.

Two types of annotations can be created for each violation found:

- The dimension annotation shows the measurement of the shape width violation.
- The rectangle annotation outlines the area of the violating shape.

Arguments

`-annotate [all | none | dim | rect]`

Specifies the annotations to use to mark violations (sections that do not meet the density requirements).

`all`	(Default) Creates rectangle and dimension annotations where violations occur.
`none`	Prevents annotations from being created.
`dim`	Creates only dimension annotations to show measurements where violations occur.
`rect`	Creates only rectangle annotations where violations occur.

The annotations are added to the `annotation:violation` purpose of the given layers and are listed by check type (*Minimum Width Checks*, *Maximum Width Checks*, or *Discrete Width Checks*) and layer, under *Width Checks* in the Violations page of the Annotation Browser.

`-annotation_limit` *`i_count`*

Specifies the maximum number of violations to create annotations for. By default, annotations are created for up to 1000 violations. Specify `-annotation_limit -1` to choose no limit.

`-clear_annotations`

Removes existing violation annotations of the same check type and input layers before performing this check. If not specified, existing violations are not cleared.

`-diagonal_aware [ true | false ]`

Specifies whether width checks are orthogonal only (`false`), or support arbitrary shapes with orthogonal and 45 degree sides (`true`/default).

`-discrete_width [ true | false ]`

Disables discrete width checking. By default, discrete width checking is performed if the `allowedWidthRange` constraint is set. Results are reported for the number of violations within ranges defined by the constraint values.

-lpp *s_layerlpp*

Specifies the list of layer purpose pairs to check.

-lpp_width [true | false]

Specifies whether widths should be checked based on the layer constraint (`false`/default) or the layer purpose pair constraint (`true`).

-max_width [*f_userunit*]

Checks for shapes that are wider than the specified width. If the given value is `0`, the maximum width (`maxWidth`) constraint from the technology file is used for the check. If this argument is not given, maximum width checking is not performed.

-no_sync

Prevents the OpenAccess database from being updated.

-object_width

Specifies that the width is checked at the object level. Default is `false`.

-output_lpp *s_layerlpp*

Specifies the layer purpose to add the checking results to. Shapes that fail to meet the width rule requirements are added.

-region **{***f_xlo f_ylo f_xhi f_yhi***}**

Specifies the boundary points for the area to check. If not specified, the entire design in the active artwork window is checked.

-silent

When selected, turns off detailed messages during the check.

-subtotals

Outputs a Tcl list with the number of minimum width, maximum width, and discrete width violations reported separately (refer to "Processing Tcl Lists" on page 1018). By default, the return value is the total number of violations found.

`-threads` *`i_threads`*

Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

`-top_level_only`

Specifies that top-level shapes and shapes connected to top-level shapes be checked. By default, all shapes are checked.

> *Important*
>
> Cell-to-cell placement errors will not be detected if this argument is true.

`-trim_corners`

Specifies that corners be trimmed for checking.

`-verbose`

Outputs a message for each error found during the check.

`-width` *`f_userunit`*

Overrides the minimum width (`minWidth`) constraint value. A value of zero prevents the minimum width check from being run.

Value Returned

`i_count`

Indicates the total number of shapes that violate the width rules for the given criteria.

`i_minviolcount`
`i_maxviolcount`
`i_discreteviolcount`

Tcl list with the number of minimum width violations, maximum width, and discrete width violations found. This is the output format when `-subtotals` is given.

`-1`

The command did not run due to a syntax error or a missing required argument.

Example

The following command checks shapes in `metal1:wire:detail` for minimum width rule violations using the width rules in the technology file, and outputs the shapes violating the rule to the `metal1:annotation:violation` LPP.

```
check_width -lpp {"metal1 wire:detail"} -output_lpp
{"metal1:annotation:violation"}
```

Processing Tcl Lists

For commands that return a Tcl list, use the following example to process the list. In this example, a Tcl list is output that includes the number of `minWidth`, `maxWidth`, and `discreteWidth` errors found.

```
set results [check_width -max_width 0.8 -subtotals]
set list_count [llength $results]
set minwidth [lindex $results 0]
set maxwidth [lindex $results 1]
set_discretewidth [lindex $results 2]
puts $minwidth; puts $maxwidth; puts $discretewidth
```

Related Information

Tcl Command	enable_multithreading
Menu Command	*Verify—Shapes*

get_shape_connectivity

```
get_shape_connectivity
     [ -shape d_setObj ]
```

Creates a set of shapes that comprise the net connected to the shape in the specified set.

Caution

This command determines connectivity for a single shape. If more than one shape is contained in the specified set, an error message is output to the Transcript area.

Arguments

`-shape` *`d_setObj`*	Specifies the set identifier that contains the given shape. If this argument is not specified, the shape in the selected set is used.

Value Returned

`d_setObj`	Specifies the identifier for shapes that comprise the net connected to the selected shape.
`-1`	The command did not run due to a syntax error or because more than one shape is selected.

Example

The following command highlights the shapes connected to the shape in the selected set.

```
add_highlight -color cyan -name conn_shapes -set [get_shape_connectivity]
```

Related Information

Tcl Commands	update_net_connectivity

update_net_connectivity

```
update_net_connectivity
     -all | -net s_netName | -set d_setObj
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -silent [ true | false ] ]
```

Checks for opens and shorts in the design by tracing through the shapes. This physical connectivity is compared against the connectivity provided by the network in the design, such as the original Design Exchange Format (DEF) file. Space-based Router and Chip Optimizer will display guides in the artwork window connecting the nearest points for any opens that are found. You can choose to evaluate the entire design, specific nets, or sets.

Arguments

`-all` Checks the connectivity of all nets in the active window.

`-exclude_net` **{***`s_netName`* …**}**

Excludes the named nets from processing.

`-exclude_set` *`d_setObj`* Excludes nets in the given set from processing.

`-exclude_type` **{**`[power][ground][clock]`**}**

Excludes nets of the specified types from processing.

`-net` *`s_netName`* Specifies the name of the net to operate on.

`-set` *`d_setObj`* Checks the connectivity on nets in the set.

`-silent [ true | false ]`

When `true`, disables printing of information messages.

Default: `false`

Value Returned

`0` Indicates the command completed. Results are reported in the Transcript area.

`-1` The command did not run due to a syntax error or when more than one shape is selected.

Example

The following example checks the connectivity for the entire cellview, excluding the `VDD` and `GND` nets.

```
update_net_connectivity -all -exclude_net { VDD GND }
```

In this example, the following is output to the Transcript area:

```
#Net GND was ignored.
#Net VDD was ignored.
#Connectivity updated on 8458 net(s), 3 guide(s) created!
#Checked 0 shapes, 17348 instTerms, 198 terms and 85 routes.
#Total CPU time: 0.3 seconds (0.3 user, 0.0 kernel)
0
```

Related Information

Tcl Commands	add_custom_via_def change_layer tech_info get_shape_connectivity verify_connectivity
Menu Commands	*Verify—Connectivity*

verify_connectivity

```
verify_connectivity
     -all | -set d_setObj | -net {s_netName…}
     [ -exclude_net {s_netName …} ]
     [ -exclude_set d_setObj
     [ -exclude_type {[power][ground][clock]}
     [ -ignore_opens_between_shape_terms [ true | false ] ]
     [ -annotate [ true | false ] ]
     [ -annotation_limit i_count ]
     [ -composite [ true | false ] ]
     [ -report s_fileName [ -level [1|2]
     [ -report_in_user_unit [ true | false ] ]
     [ -threads i_count ]
```

Checks for opens and shorts in the design by tracing through the shapes. This physical connectivity is compared against the connectivity provided by the network in the design, such as the original Design Exchange Format (DEF) file. Annotations can be added for the opens and shorts. You can choose to evaluate the entire design, specific nets, or nets in a set.

Arguments

`-all` Checks the connectivity of all nets in the active window.

`-annotate [ true | false ]`

Adds annotations for opens and shorts that are found.

For shorts, red rectangle annotations outline the shorted metal. The annotations are added to the `annotation:violation` purpose of the layer and are listed by net under *Connectivity Checks—Shorts Checks* in the Violations page of the Annotation Browser.

For opens, red line annotations are added to the `annotations:extern viols` objects and are listed by net under *Connectivity Checks—Nets With Opens* in the Violations page of the Annotation Browser.

By default, annotations are added.

`-annotation_limit i_count`

Specifies the maximum number of annotations that can be created for this operation. The default value is 1000.

`-composite [ true | false ]`

When set to `true`, only checks the connectivity of composite nets used in specialty routing such as pair routing.

Default: (`false`) Checks specified nets given by `-all`, `-set` or `-net`.

-exclude_net **{***s_netName…***}**

Excludes nets in the given list.

-exclude_set *d_setObj* Excludes nets in the given set.

-exclude_type **{**[power][ground][clock]**}**

Excludes nets of the specified types.

-ignore_opens_between_shape_terms [true | false]

Ignores open violations between shape terms. The default is `false`.

-level [1|2] (Valid only when `-report` is given) Specifies the level of detail to include in the report. Level 1 outputs the summary of nets with opens and shorts and is the default. Level 2 lists details for shorts and opens by net.

-net **{***s_netName…***}** Specifies the names of the nets to operate on.

-report *s_fileName* Outputs errors to the given file.

-report_in_user_unit When set to `true`, reports connectivity metrics in user unit. This option is valid only when the `-report` and `-level` option values are set to `2`.

-set *d_setObj* Operates on nets in the given set.

-threads *i_threads* Specifies the number of threads or processors to use in parallel to run this command. By default, if multi-threading has been enabled, the session threads are used, otherwise, one processor is used.

Value Returned

`i_count` Indicates the total number of opens and shorts found.

`-1` The command did not run due to a syntax error or a missing required argument.

Example

The following example verifies the connectivity for the entire cellview, excluding the `VDD` and `VSS` nets.

```
verify_connectivity -all -exclude_net {VDD VSS}
```

In this example, the following is output to the Transcript area:

```
Net VSS was ignored.
Net VDD was ignored.
Connectivity verified on 8458 net(s). Found 3 open(s) and 2 short(s).
Checked 603570 shapes, 27755 instTerms, 55 terms.
Total CPU time: 10.6 seconds (0.1 kernel, 10.5 user)
5
```

The following example verifies the connectivity on a single net and adds annotations for opens and shorts that are found.

```
verify_connectivity -net_name "clk" -annotate
```

Related Information

Tcl Commands	add_custom_via_def change_layer tech_info get_shape_connectivity update_net_connectivity
Menu Commands	*Verify—Connectivity*

17

Window Commands

This chapter describes the Window commands. The commands are presented in alphabetical order.

add_button

```
add_button
     { -name s_name -command s_cmdText | -separator }
     [ -toolbar s_name ]
     [ -index i_position ]
     [ -icon_label s_label [ -font s_fontName ] [ -size i_size ][ -icon
     s_index ] ]
     | -icon_file s_fileName ]
     [ -force_width i_pixel -force_height i_pixel ]
```

Adds a button or a vertical separator to a toolbar.

Arguments

`-command` *`s_cmdText`*	Specifies the Tcl commands to invoke when you click the button. The string must be enclosed in quotes. Multiple commands can be included, separated by the semicolon (;) character.
`-font` *`s_fontName`*	Specifies the font type to use for the label.
`-force_height` *`i_height`*	Sizes the height, in pixels, for an icon that is given by `-icon_file`. If this argument is not given, the icon is sized to standard dimensions.
`-force_width` *`i_width`*	Sizes the width, in pixels, for an icon that is given by `-icon_file`. If this argument is not given, the icon is sized to standard dimensions.
`-icon` *`i_index`*	Specifies the index for the icon to use or, if -`icon_label` is given, the color scheme to use for the icon.
`-icon_file` *`s_fileName`*	Specifies the file that contains the icon to display for the button.
`-icon_label` *`s_label`*	Draws the icon as a rectangular button containing the label. For multi-line labels, use "`\n`". For example, "`line1\nline2`".
`-index` *`i_position`*	Specifies the position in the toolbar to place the button or separator. By default, the button or separator is added after the last button/separator. A value of 0 is the first position in the toolbar.

`-name` *`s_name`*	Specifies the name to assign to the button. This name is displayed when you move the cursor over the button in the toolbar.
`-separator`	Adds a vertical line separator to the toolbar. By default, the separator is added after the last button/separator but can be placed using `-index`.
`-size` *`i_size`*	Specifies the point size for a label given by `-icon_label`.
`-toolbar` *`s_name`*	Specifies the name of the toolbar to add the button to. The Space-based Router and Chip Optimizer built-in toolbars are named: `Standard`, `View`, `Edit`, `Panels`, `Advanced Editing`, `View Context Quick Navigator`, `Annotate`, and `User-defined`. If you specify a toolbar other than one of the built-ins, the toolbar will be created. If you do not include this argument, the button is added to the User-defined toolbar.

Example

The following command adds an icon button labeled `TDSP` to the User-defined toolbar. When you click the TDSP button, the tdsp design will be opened.

```
add_button -name load_tdsp -command "read_db -lib tdsp_library -cell tdsp -view
layout" -icon_label TDSP
```

The following command adds a blue icon button to the `Bold` toolbar. When the button is clicked, highlight sets are created to use with `remaster_via`.

```
add_button -name add_highlights -command "add_highlight -color red -name
NEW_VIAS_PUSHED; add_highlight -color blue -name NEW_VIAS; add_highlight -color
purple -name UNFIXED_VIAS" -icon 2 -toolbar Bold
```

arrange_window

```
arrange_window
     [ -cascade | -tabify | -tile ]
```

Selects the format for arranging artwork windows in the workspace.

Arguments

`-cascade` Displays windows in overlapping cascade format.

`-tabify` Displays one window at a time in the workspace. Windows are tabbed.

`-tile` Displays multiple non-overlapping windows in the workspace.

Example

```
arrange_window -cascade
```

Arranges artwork windows in overlapping cascade format.

Related Information

Menu Commands *Window—Arrange—Tabify*
Window—Arrange—Tile
Window—Arrange—Cascade

busy_button

```
busy_button
     {-hide | -show}
```

Hides or shows the Status/Interrupt (busy) button in the Graphical User Interface. By default, the busy button is visible and updated periodically to show the status of the current operation. If you are running interactively over a slow network, it can be helpful to temporarily hide the busy button for faster processing.

Arguments

`-hide`	Hides the Status/Interrupt button.
`-show`	Shows the Status/Interrupt button.

Example

The following command hides the busy button.

```
busy_button -hide
```

hide_gui

```
hide_gui
     [ -aerial_view ]
     [ -color_map_legend ]
     [ -command_line ]
     [ -lods ]
     [ -net_manager ]
     [ -sequencer ]
     [ -transcript ]
     [ -view_contexts ]
```

Hides the specified auxiliary windows in the GUI.

Arguments

`-aerial_view`	Hides the aerial view.
`-color_map_legend`	Hides the Color Map Legend.
`-command_line`	Hides the command entry area.
`-lods`	Hides the layer object display panel.
`-net_manager`	Hides the Net Manager docking window.
`-sequencer`	Hides the Sequencer docking window.
`-transcript`	Hides the Transcript area.
`-view_contexts`	Hides the view contexts browser.

Example

This command hides the aerial view.

```
hide_gui -aerial_view
```

Related Information

Tcl Commands	show_gui

Menu Commands

Window—Aerial View
Window—Command Line
Window—Layer Object Panel
Window—Transcript Area
Window—View Contexts Browser

new_window

```
new_window
     [ -window_id i_windowID ]
```

Creates a new artwork window with the cellview in the active window or the given window.

Arguments

`-window_id` *`i_windowID`*	Uses the cellview in the given window for the new window.

Value Returned

`i_windowID`	Is the identifier for the new window.
`-1`	A syntax error occurred or the cellview cannot be determined because the given or active window does not exist.

Example

The following command creates a new artwork window with the cellview in the active window.

```
new_window
```

Related Information

Menu Commands	*Window—New Window*

remove_button

```
remove_button
     -toolbar s_name
     -index i_position
```

Removes a button from a toolbar.

Arguments

`-index i_position`	Specifies the toolbar index of the button to remove. An index of 0 is the first button on the toolbar.
`-toolbar s_name`	Specifies the name of the toolbar to remove the button from. The Space-based Router and Chip Optimizer built-in toolbars are named: `Standard`, `View`, `Edit`, `Panels`, `Advanced Editing`, `View Context Quick Navigator`, `Annotate`, and `User-defined`.

Example

The following command removes the second button from the `User-defined` toolbar.

```
remove_button -toolbar User-defined -index 1
```

Related Information

Tcl Commands	add_button

remove_tool_bar

```
remove_tool_bar
     -name s_name | -empty_bars
```

Removes the specified toolbar or all toolbars that have no buttons.

Arguments

`-empty_bars`	Removes toolbars that have no buttons.
`-name s_name`	Specifies the name of the toolbar to remove.

Related Information

Tcl Commands	add_button remove_button

restore_default_configuration

```
restore_default_configuration
```

Resets the size and placement of the main Space-based Router and Chip Optimizer window and its dock windows, including the toolbars, to default values.

Arguments

None

Related Information

Menu Commands	*Window—Restore Default Configuration*

show_gui

```
show_gui
     [ -aerial_view ]
     [ -color_map_legend ]
     [ -command_line ]
     [ -lods ]
     [ -net_manager ]
     [ -sequencer ]
     [ -transcript ]
     [ -view_contexts ]
```

Displays the auxiliary areas in the GUI.

Arguments

`-aerial_view`	Displays the aerial view.
`-color_map_legend`	Displays the Color Map Legend.
`-command_line`	Displays the command entry area.
`-lods`	Displays the layer object display panel.
`-net_manager`	Displays the Net Manager docking window.
`-sequencer`	Displays the Sequencer docking window.
`-transcript`	Displays the Transcript area.
`-view_contexts`	Displays the view contexts browser.

Example

This command displays the layer/object display panel and the aerial view.

```
show_gui -lods -aerial_view
```

Related Information

Tcl Commands	hide_gui

Menu Commands

Window—Aerial View
Window—Command Line
Window—Transcript Area
Window—Layer Object Panel
Window—View Contexts Browser

window_graphics_type

```
window_graphics_type
     [ -window_id i_windowID ]
```

Returns the graphics type of the given window. This is useful if you have changed the `gui.graphics` environment variable directly or in the Session Options form (*Edit—Preferences—Session*) mid-session and want to determine with certainty the type of a window. By default, the type of the active window type is returned.

Arguments

`-window` *`i_windowID`*	Gets the graphics type of the specified window. If this argument is not given, the active window is used.

Value Returned

`opengl | qt | nograph`	Is the graphics type of the window. If you started up Space-based Router and Chip Optimizer with the `-noGraph` argument, this will return `nograph`.

18

Report Commands

Report commands let you generate reports.

Color maps are used to graphically display computational results from congestion analysis, CMP analysis, and yield analysis.

The commands are presented in alphabetic order:

- ❑ unload_color_map on page 1082

report_design_stats

```
report_design_stats
     [ -file s_fileName ]
```

Outputs a summary of layer, component, and connectivity statistics for the design in the active window to the Transcript area, and optionally to a file.

Arguments

`-file s_fileName`	Outputs the design statistics to the given file. By default, no file is created.

Example

The following example outputs the design statistics for active window to the Transcript area and file `small8000.txt`.

```
report_design_stats -file small8000.txt
```

The following is an example of design statistics reported:

```
# Design Statistics Summary: small8000/small8000_routed/layout
#  Layer Summary
#   layers: 14, routing layers: 5
#  Component Summary
#   total components: 8306, unplaced:    0, placed: 8306, fixed:    0
#          core comp: 8102, unplaced:    0, placed: 8102, fixed:    0
#           pad comp:  197, unplaced:    0, placed:  197, fixed:    0
#    block/ring comp:    3, unplaced:    0, placed:    3, fixed:    0
#         other comp:    4, unplaced:    0, placed:    4, fixed:    0
#  Connectivity Summary
#   pins: 614
#   total nets: 8458, unrouted:   55, routed: 8403, single pin or undetermined:0
#       signal: 8456, unrouted:   55, routed: 8401, single pin or undetermined:0
#        power:    1, unrouted:    0, routed:    1, single pin or undetermined:0
#       ground:    1, unrouted:    0, routed:    1, single pin or undetermined:0
#        clock:    0, unrouted:    0, routed:    0, single pin or undetermined:0
#        other:    0, unrouted:    0, routed:    0, single pin or undetermined:0
#   terminals: 45090
# End of Design Statistics Summary
```

The `single pin or undetermined` category represents nets that contain at most one *real* terminal. For example, a net with two pins, one real and one *logical* (no geometries associated with it) would be included in this category.

Related Information

Menu Commands	*Report—Design Statistics*

report_fill

```
report_fill
     [ -layer {all | {s_layer…}} ]
     [ -all | -set d_setObj | -region {f_xlo f_ylo f_xhi f_yhi} ]
     [ -print_number [ true | false ] ]
     [ -print_area [ true | false ] ]
     [ -tcl_list [ true | false ] ]
```

Reports the number of fill shapes and, optionally, the area, in micron^2, occupied by the fill shapes of each type:

- Floating fill (`fill` purpose)
- Notch fill (`gapFill` purpose)
- Power/Ground connected fill (`fill` purpose)

  For power and ground nets, each net is reported separately.

You can optionally limit the report to a given region, or to nets in a set, otherwise fill shapes in the entire design are reported for all layers or the given layers only.

Arguments

`-all` (Default) Reports on the entire design.

`-layer {all | {`*`s_layerName…`*`}}`

Reports on fill shapes in the entire design (`all`) or on the given layers. By default, fill shapes in the entire design are reported.

`-print_area [ true | false ]`

(Default/true) Reports the total shape area for each type of shape in micron^2.

`-print_number [ true | false ]`

(Default/true) Reports the number of shapes for each type of fill.

`-region {`*`f_xlo f_ylo f_xhi f_yhi`*`}`

Limits reporting to fill shapes in the specified area of the cellview, given by the lower left and upper right coordinates. By default, fill shapes in the entire design are reported.

`-set d_setObj` Limits the report to nets in the given set.

`-tcl_list [ true | false ]`

Outputs the results as a Tcl list.

Example

The following is an example of the values reported for this command:

```
report_fill -region [get_window_area] -layer Metal2
```

```
+------+-------------+-------+-----------+-------+-------+----------------+
| Layer|   Floating  |GapFill|  VDD_PLL  |  VDD  |  VSS  |     Total      |
+------+-------------+-------+-----------+-------+-------+----------------+
|      |Num      Area|NumArea|    NumArea|NumArea|NumArea|  Num       Area|
+------+-------------+-------+-----------+-------+-------+----------------+
|Metal2| 64       453|  0   0|      0   0|  0   0|  0   0|   64        453|
+------+-------------+-------+-----------+-------+-------+----------------+
| TOTAL| 64       453|  0   0|      0   0|  0   0|  0   0|   64        453|
+------+-------------+-------+-----------+-------+-------+----------------+
```

Related Information

Tcl Commands

connect_fill
create_fill
create_net_fill
create_pg_fill

report_grids

```
report_grids
     [ -file s_fileName ]
```

Outputs the x- and y-step values, the x- and y-offset values and the manufacturing grid for each metal and via layer in the active window to the Transcript area, and optionally to a file.

Arguments

`-file` *`s_fileName`*

Outputs the grid information to the given file. By default, no file is created.

Example

The following is an example of the values reported for this command:

```
--------------------------------------------------------------------------------
Layer         X Step      Y Step      X Offset      Y Offset      Mfg Grid
--------------------------------------------------------------------------------
PC               0.4         0.4             0             0             0
CA               0.4         0.4             0             0             0
Metal1           0.4         0.4             0             0             0
V1               0.4         0.4             0             0             0
Metal2           0.4         0.4             0             0             0
```

Related Information

Tcl Commands

get_routing_grid
get_via_grid

report_net_stats

```
report_net_stats
     { -set d_setObj | -net {s_netName…} | -summary }
     [ -file s_fileName ]
     [ -omit_transcript ]
     [ -include_extents ]
```

Generates a Net Statistics report on the nets in the given list or set, or a Net Term Count report for all nets (`-summary`).

The Net Statistics report is output to the Transcript area and, optionally, to a file, and reports either the net count (`-omit_transcript`), or the following information:

- Number of terms (logical points on nets that can be connected from outside)
- Number of inst terms (logical points on instances that can be connected to a net)
- Number of guides (opens in nets)
- Number of steiners (pins that are not associated with a terminal or instance and are used to implement a virtual pin for routing control)
- Total, horizontal, and vertical wire length for each layer
- Number of vias for each layer
- Number of route segments for each layer

The Net Term Count report gives the number of nets grouped by the number of terms per net.

Arguments

`-file` *`s_fileName`*	Specifies a file to output the net statistics to. By default, no file is created.
`-include_extents`	Includes extents in the reported wire lengths. By default, extents are not included.
`-net` **{***`s_netName…`***}**	Reports on nets in the list.
`-omit_transcript`	Prevents net statistics from being output to the Transcript area. By default, the report is output to the Transcript area.
`-set` *`d_setObj`*	Reports on nets in the given set.

`-summary` Reports the number of nets with the same number of terms per net. For example, `2:6  3:1  4:2` reports that six nets have 2 terms, one net has 3 terms and two nets have 4 terms.

Example

The following example outputs the net statistics for the nets in the `HL1` highlight set to the Transcript area and the `HL1nets.txt` file.

```
report_net_stats -file HL1nets.txt -set [get_highlight -name HL1]
```

The following is an example of net statistics reported for one of the nets in the set:

```
Net Statistics:

Net: TDSP_CORE_INST_dmov_inc       Cellview: small8000/small8000_routed/layout

# Terms  # Inst Terms  # Guides  # Steiners
0            29         0           0

Layer     Wirelength  HWirelength  VWirelength  # Vias (Down)  # Route Segments
Total         1804.4       1100.1        704.3             99                93
met1            88.5         88.5            0              0                12
met2           476.2           12        464.2             45                42
met3           633.1        631.2          1.9             34                28
met4           238.2            0        238.2             12                 7
met5           368.4        368.4            0              8                 4
```

The following is an example of a Net Term Count report.

```
Net Term Count Report (12 nets):
    1:   4      2:   2      3:   1      4:   3      5:   2
12
```

Related Information

Menu Commands *Report—Net Statistics*

report_routing_stats

```
report_routing_stats
     [ -set d_setObj | -net {s_netName…} ]
     [ -file s_fileName ]
     [ -summary ]
     [ -subtotals [ -silent] ]
```

Generates a report on the nets in the given list or set, or the entire design. The report is output to the console, Transcript area and, optionally, to a file. The following information is included:

- Routing Summary

  Includes the number of nets, routes and guides found and the percentage of routing that has been completed.

- Routing History
- Layer Details

  Reports by layer down-via counts, the lengths for guides, and detail and global routes by direction.

You can optionally create an environment variable package containing the statistics (`-subtotals`).

Arguments

`-file` *`s_fileName`*	Specifies a file to output the routing statistics to. By default, no file is created.
`-net` **{***`s_netName…`***}**	Limits reporting to the nets in the given list. The Routing Summary and Layer Details for the nets in the list are output. By default, all nets are reported.
`-set` *`d_setObj`*	Limits reporting to the nets in the given set. The Routing Summary and Layer Details for the nets in the set are output. By default, all nets are reported.
`-silent`	Suppresses all console and log file output. This is useful to streamline processing when running scripts.
`-subtotals`	Creates an environment variable package containing the statistics for the entire design. For information about the environment variable package, refer to “Reading Statistics from an Environment Variable Package” on page 1051.

`-summary`	Outputs only the Routing Summary. If no arguments are given, the Routing Summary, Routing History, and Layer Details for the entire design are all output.

Reading Statistics from an Environment Variable Package

When you use the `-subtotals` argument, an environment variable package is created that contains the statistics for the entire design. The package contains two child packages, `count` and `stats`.

Child Environment Variables	Type	Description
`count.nets`	IntegerEnv	Number of nets
`count.routes`	IntegerEnv	Number of routes
`count.guides`	IntegerEnv	Number of guides
`count.completion_pct`	RealEnv	% Routing Completed
`stats.wire_length`	RealEnv	Total wire length (by layer)
`stats.detail_horiz_length`	RealEnv	Total horizontal detail length (by layer)
`stats.global_horiz_length`	RealEnv	Total horizontal global length (by layer)
`stats.detail_vert_length`	RealEnv	Total vertical detail length (by layer)
`stats.global_vert_length`	RealEnv	Total vertical global length (by layer)
`stats.guide_count`	IntegerEnv	Number of guides (by layer)
`stats.down_vias`	IntegerEnv	Number of down vias (by layer)
`stats.guide_down_vias`	IntegerEnv	Number of guide down vias (by layer)

For example, to access the data given layers `metal1`, `metal2` and `metal3`, use the following:

```
set net_count [getvar -cmd report_routing_stats count.nets]
set route_count [getvar -cmd report_routing_stats count.routes]
set guide_count [getvar -cmd report_routing_stats count.guide_count]
set completed_pct [getvar -cmd report_routing_stats count.completion_pct]
set metal1_length [getvar -cmd report_routing_stats stats.metal1.wire_length]
set metal2_down_vias [getvar -cmd report_routing_stats stats.metal2.down_vias]
```

Example

The following example requests routing statistics and shows the output.

```
report_routing_stats

Nets       =    8458   Routes =    29222   Guides =        0
Completion = 100.00%
| ROUTING HISTORY ==============
|       Pass         |         |
| Name          | No. | Max Mem |
|------------------------------|
| GRoute        |   1 |    72.3 |
| GRoute        |   2 |    72.3 |
| GRoute        |   3 |    72.3 |
| GRoute        |   4 |    72.3 |
|------------------------------|

= LAYER DETAILS
=========================================================================================
|        |   |      Total |     Horizontal   |     Vertical     | Guide |    Down Vias  |
|  Layer |Dir|     Length |    Detail |Global|     Detail|Global| Length|  Total | Guide|
|---------------------------------------------------------------------------------------|
|  met1  | H |   45223.77 |  44390.36 | 0.00 |    833.41 | 0.00 |  0.00 |      0 |    0 |
|  met2  | V |  193789.39 |   7112.48 | 0.00 | 186676.91 | 0.00 |  0.00 |  27803 |    0 |
|  met3  | H |  325697.86 | 318464.60 | 0.00 |   7233.26 | 0.00 |  0.00 |  23392 |    0 |
|  met4  | V |  306787.93 |   1987.13 | 0.00 | 304800.80 | 0.00 |  0.00 |   8265 |    0 |
|  met5  | H |  254937.34 | 254694.66 | 0.00 |    242.68 | 0.00 |  0.00 |   3205 |    0 |
|---------------------------------------------------------------------------------------|
| Totals |   | 1126436.29 | 626649.23 | 0.00 | 499787.06 | 0.00 |  0.00 |  62665 |    0 |
| Percent|   |     100.00 |     55.63 | 0.00 |     44.37 | 0.00 |  0.00 |        |      |
=========================================================================================
```

Related Information

Menu Commands	*Report—Routing Statistics*

report_rs

```
report_rs
     [ -all
     | -name s_rsName
     | -net s_netName
     | -set d_setObj ]
     [ -filename s_fileName ]
     [ -match [ true | false ] ]
     [ -routes ]
```

Outputs to the Transcript area route spec information (width, spacing, pitch, routability of metal layers and via layers) by layer, and/or route spec names. If no arguments are given, the layer information is given for the global net default route spec.

Arguments

`-all`	Outputs the names of all of the used route specs on the design.
`-filename` *`s_fileName`*	Used with the `-name` argument, prints detailed route spec information to the given file.
`-match [ true | false ]`	
	Used with the `-name` argument, when set to `true`, outputs the names of the nets with the named route spec. By default, the names of the nets that are on different route specs are output with their route spec names.
`-name` *`s_rsName`*	Outputs the information for the given route spec. If no name is given, the name of the global net default route spec is given.
`-net` *`s_netName`*	Outputs the route spec information for the given net.
`-routes`	Outputs the route spec name for every net, and the route spec name for every route of the net that differs from the net's route spec.
`-set` *`d_setObj`*	For each net in the specified set, outputs the name of the route spec and the route spec information.

Example

The following example requests the names of all of the used route specs in the active cellview and shows the output.

```
report_rs -all
"LEFDefaultRouteSpec" "wideRouteSpec"
```

Related Information

Tcl Commands	dump_ctu_constraints

report_via_stackability

```
report_via_stackability
     [ -via s_viaName [ -via2 s_viaName
     [ -rot1 {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ]
     [ -rot2 {R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90} ] ] ]
```

Reports the stackability of vias in the active cellview or for specific vias. When no argument is given, each master via instance is listed with the following attributes:

- topOfStack
- bottomOfStack

Arguments

`-rot1` *`s_orientName`*

Specifies the orientation for the first via. Valid values are: `R0`, `R90`, `R180`, `R270`, `MY`, `MYR90`, `MX`, and `MXR90`. For a description of orientation values, refer to “Orientation Key” on page 208.

Default value: `R0`

`-rot2` *`s_orientName`*

Specifies the orientation for the first via. Valid values are: `R0`, `R90`, `R180`, `R270`, `MY`, `MYR90`, `MX`, and `MXR90`. For a description of orientation values, refer to “Orientation Key” on page 208.

Default value: `R0`

`-via` *`s_viaName`*

Names a specific via to report on. If this argument is not given, all vias are reported.

`-via2` *`s_viaName`*

Specifies a second via. Use with `-via` to specify that the report indicate whether the two vias are stackable.

Example

The following is an example of partial data output by Space-based Router and Chip Optimizer after a `report_via_stackability` command is issued:

```
Via56_stack_west topOfStack true bottomOfStack false
Via34_stack_west topOfStack true bottomOfStack false
Via56_stack_east topOfStack true bottomOfStack false
```

report_via_stats

```
report_via_stats
     [ -window_id i_windowID ]
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj ]
     [ -do_via_layers {s_layerName …} ]
     [ -exclude_set d_setObj ]
     [ -exclude_net {s_netName …} ]
     [ -exclude_type {[power] [ground] [clock]} ]
     [ -exclude_via_pin [ true | false ] ]
     [ -output_pins_only [ true | false ] ]
     [ -report_level {summary | basic | detail} ]
     [ -file s_fileName ]
     [ -show_all_columns [ true | false ] ]
```

Outputs statistics for vias in a set or the selected set, within a given region or in the entire design to the Transcript area, and optionally to a file. If the `cutClass` constraint is defined, the report includes a cut class name column for each cut count group.

Note: To properly report larger enclosure single-cut via counts, the `minDualExtension` constraint must be set.

Arguments

`-all` Sets the target area to the entire cellview.

`-do_via_layers` **{***s_layerName …***}**

Limits the output to the via layers in the given list. If this argument is not specified, all via layers are reported.

`-exclude_net` **{***s_netName …***}**

Specifies the names of nets to exclude from reporting.

`-exclude_set` *d_setObj* Excludes from reporting vias, routes and nets in the given set.

`-exclude_type` **{**`[power][ground][clock]`**}**

Specifies one or more net types to exclude from reporting.

`-exclude_via_pin [ true | false ]`

Excludes via pins (top level only) from the report. By default, both via detail and via pins are included.

`-file` *s_fileName* Outputs the via information to the given file. By default, no file is created.

-output_pins_only [true | false]]

Limits processing to vias on output pins.

Default: false

-region {*f_xlo f_ylo f_xhi f_yhi*}

Limits reporting to vias only in the specified area of the cellview, given by the lower left and upper right coordinates.

-report_level {summary | basic | detail}

Controls the amount of detail to include in the summary:

summary	Reports the number of vias for each layer, grouped by the number of cuts and whether they are minimum enclosure or larger enclosure. This is the default.
basic	Reports summary information and subgroups vias on each layer by enclosure dimensions.
detail	Reports basic information and subgroups vias on each layer by via master.

-set *d_setObj* Reports only vias in the specified set. If no target area or set is specified, the command operates on the vias in the selected set.

-show_all_columns [true | false]

If true, each cut count is reported separately. By default or if false, results are broken down as 1-cut, 2-cut, 3-cut, 4-cut and greater than 4-cut (vias with greater than 4 cuts are consolidated into one column).

-window_id *i_windowID* Specifies the window to process. If this argument is not specified, the active artwork window is used.

Example

The following is an example of a summary via report.

```
# Design:  tdsp_library/tdsp/layout
 Date:  Fri Feb 03 2006    Time:  13:05:44
 Elapsed Time:  0.1 seconds     Memory Usage:  0.0 MB
remaster_via completed in = 0.1s (elapsed), using 0.1s/0.0s/0.1s (user/kernel/
total CPU).
+----------------------------------------------------------+
|Layer|Count|         1-cut         |         2-cut         |
|     |     |       Enclosure       |       Enclosure       |
|     |     |     Min    | Larger   |    Min     |  Larger  |
+----------------------------------------------------------+
|Via1 |   69| 22  (31.9%)|0   (0.0%)|47  (68.1%)|0   (0.0%)|
|Via2 |   81| 81 (100.0%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|----------------------------------------------------------|
|Total|  150|103  (68.7%)|0   (0.0%)|47  (31.3%)|0   (0.0%)|
+----------------------------------------------------------+
```

The following is an example of a `basic` via report.

```
# Design:  tdsp_library/tdsp/layout
 Date:  Fri Feb 03 2006    Time:  13:06:05
 Elapsed Time:  0.0 seconds     Memory Usage:  0.0 MB
report_via_stats completed in = 0.0s (elapsed), using 0.0s/0.0s/0.0s (user/
kernel/total CPU).
+--------------------------------------------------------------------------+
|Layer|   Enclosure   |Count|         1-cut          |         2-cut        |
|     | (Lower Upper) |     |      Enclosure         |      Enclosure       |
|     |               |     |     Min    |  Larger   |     Min    |  Larger |
+--------------------------------------------------------------------------+
|Via1 |               |   69| 22  (31.9%)|0   (0.0%)|47  (68.1%)|0   (0.0%)|
|     |0.4x0.4 0.4x0.4|   22| 22 (100.0%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|     |0.4x0.9 0.4x0.9|   24|  0   (0.0%)|0   (0.0%)|13  (54.2%)|0   (0.0%)|
|     |0.9x0.4 0.9x0.4|   23|  0   (0.0%)|0   (0.0%)|13  (56.5%)|0   (0.0%)|
|     |0.4x0.9 0.4x0.9|   24|  0   (0.0%)|0   (0.0%)|11  (45.8%)|0   (0.0%)|
|     |0.9x0.4 0.9x0.4|   23|  0   (0.0%)|0   (0.0%)|10  (43.5%)|0   (0.0%)|
|Via2 |               |   81| 81 (100.0%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|     |0.4x0.5 0.4x0.4|   21|  9  (42.9%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|     |0.4x0.4 0.4x0.4|   60| 60 (100.0%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|     |0.4x0.5 0.4x0.4|   21| 12  (57.1%)|0   (0.0%)| 0   (0.0%)|0   (0.0%)|
|--------------------------------------------------------------------------|
|Total|               |  150|103  (68.7%)|0   (0.0%)|47  (31.3%)|0   (0.0%)|
+--------------------------------------------------------------------------+
```

The following is an example of a `detail` via report.

```
# Design:  tdsp_library/tdsp/layout
 Date:  Fri Feb 03 2006    Time:  13:06:12
 Elapsed Time:  0.0 seconds     Memory Usage:  0.0 MB
report_via_stats completed in = 0.0s (elapsed), using 0.0s/0.0s/0.0s (user/
kernel/total CPU).
+-------------------------------------------------------------------------------+
|Layer|Via Master|   Enclosure   |Cnt|         1-cut         |       2-cut        |
|     |          | (Lower Upper) |   |       Enclosure       |     Enclosure      |
|     |          |               |   |   Min    |  Larger   |    Min   | Larger  |
+-------------------------------------------------------------------------------+
|Via1 |          |               | 69|22 (31.9%)|0   (0.0%)|47 (68.1%)|0  (0.0%)|
|     |M2M1      |0.4x0.4 0.4x0.4| 22|        22|         0|         0|         |
|     |M2M1_north|0.4x0.9 0.4x0.9| 13|         0|         0|        13|        0|
|     |M2M1_west |0.9x0.4 0.9x0.4| 13|         0|         0|        13|        0|
|     |M2M1_south|0.4x0.9 0.4x0.9| 11|         0|         0|        11|        0|
|     |M2M1_east |0.9x0.4 0.9x0.4| 10|         0|         0|        10|        0|
|Via2 |          |               | 81|81(100.0%)|0   (0.0%)| 0  (0.0%)|0  (0.0%)|
|     |Via23south|0.4x0.5 0.4x0.4|  9|         9|         0|         0|        0|
|     |M3M2      |0.4x0.4 0.4x0.4| 60|        60|         0|         0|        0|
|     |Via23north|0.4x0.5 0.4x0.4| 12|        12|         0|         0|        0|
|-------------------------------------------------------------------------------|
|Total|          |               |150|103(68.7%)|0   (0.0%)|47 (31.3%)|0  (0.0%)|
+-------------------------------------------------------------------------------+
```

Related Information

Menu Commands	*Report—Via Statistics*

report_wire_spacing_stats

```
report_wire_spacing_stats
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -file s_fileName ]
     [ -ignore_blockages [ true | false ] ]
     [ -ignore_boundary [ true | false ] ]
     [ -layers {s_layerName…} ]
     [ -length_threshold f_userunit ]
     [ -reported_space {f_spacing…} ]
     [ -top_level_only ]
```

Reports statistics on spacing between nets in the entire design, in a given set, or in a given region. This is useful for quantifying spread results.

The statistics include the following:

- For each layer,
  - Edge lengths for each reported space grouping
  - The percentage of all edge lengths for each reported space grouping
  - The total edge length
- The totals for each column

By default, the space columns report spacings for facing edges that are `minSpacing`, `minSpacing`*2, `minSpacing`*3, and greater than `minSpacing`*3. You can change the reported spacing groups using -`reported_space`.

Only edge lengths that are greater than the `minWidth` constraint for the layer will be reported. You can optionally change the length threshold using the -`length_threshold` argument.

Arguments

`-all` Reports on all nets. This is the default.

`-exclude_type {[power][ground][clock]}`

Excludes the given net types from the report. By default, all net types are included.

`-file s_fileName` Outputs the report to the given file.

`-ignore_blockages [ true | false ]`

Specifies whether to ignore spacing between wires and blockages.

Default: `false`

`-ignore_boundary [ true | false ]`

Specifies whether to ignore wire spacing to the area boundary.

Default: (`true`) Wire spacing to the area boundary is ignored.

`-layers` **{***`s_layerName`*…**}** Reports only on layers in the list. By default, all valid routing layers are reported.

`-length_threshold` *`f_userunit`*

Restricts processing to edge lengths that are greater than the given length. The minimum value is `minWidth` for the layer.

Default: `minWidth` for the layer

`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Restricts reporting to the area given by the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates.

`-reported_space` **{***`f_spacing`*…**}**

Groups reported spacings according to the values in the given list. Each value must be a multiple of the manufacturing step. The default reported spacings are minSpacing, minSpacing*2, and minSpacing*3.

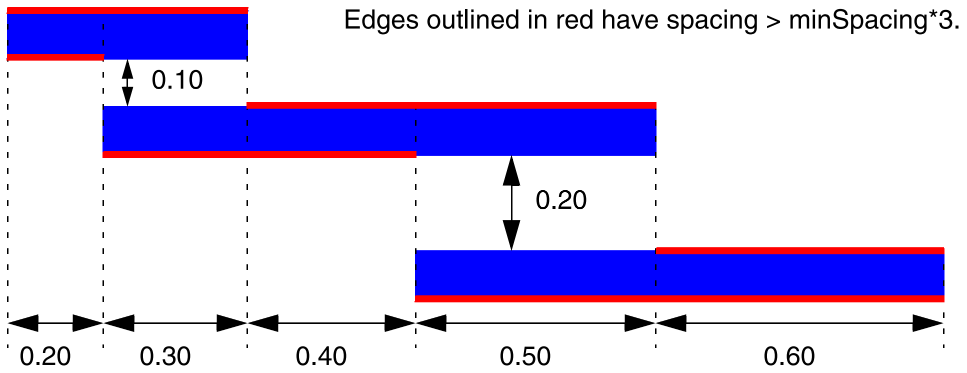
`-set` *`d_setObj`* Reports on nets in the given set.

`-top_level_only` When set to `true`, reports only wires on the top level.

Default: Wire spacing is reported for top-level wires only.

Example

The following example shows how spacings are reported.



Edges outlined in red have spacing > minSpacing*3.

For this example, the following is the default report (assuming a `minSpacing` of 0.10) for the M1 layer:

```
+-------+-------------+-------------+-------------+------------+------------+
| Layer |space <=0.10 |space <=0.20 |space <=0.30 |space >0.30 |    Total   |
+-------+-------------+-------------+-------------+------------+------------+
|M1     |  0.60 (11%) |  1.00 (18%) |  0      (0%) | 4.00 (71%) | 5.6 (100%) |
+-------+-------------+-------------+-------------+------------+------------+
| TOTAL |  0.60 (11%) |  1.00 (18%) |  0      (0%) | 4.00 (71%) | 5.6 (100%) |
+-------+-------------+-------------+-------------+------------+------------+
```

The space columns report spacings for facing edges that are `minSpacing`, `minSpacing`*2, `minSpacing`*3, and greater than `minSpacing`*3.

report_wire_width_stats

```
report_wire_width_stats
     [ -all | -region {f_xlo f_ylo f_xhi f_yhi} | -set d_setObj ]
     [ -exclude_type {[power][ground][clock]} ]
     [ -file s_fileName ]
     [ -layers {s_layerName…} ]
```

Reports statistics on wire widths for route segments in the entire design, in the given set, or wholly within the given region. This is useful for quantifying `widen_wire` results. Each `gapFill` shape (used for wire widening) can only belong to one route segment.

The statistics include the following:

- The number of route segments (NRS) for each width and layer
- The length of route segments (LRS) for each width and layer
- The total number of route segments by layer
- The total length of route segments by layer
- The average width of the reported route segments

  This gives a quick measure for comparison when widening wires.

Arguments

`-all` Reports on all nets. This is the default.

`-exclude_type {[power][ground][clock]}`

Excludes the given net types from the report. By default, all net types are included.

`-file` *`s_fileName`* Outputs the report to the given file.

`-layers` **{***`s_layerName…`***}** Reports only on layers in the list. By default, all valid routing layers are reported.

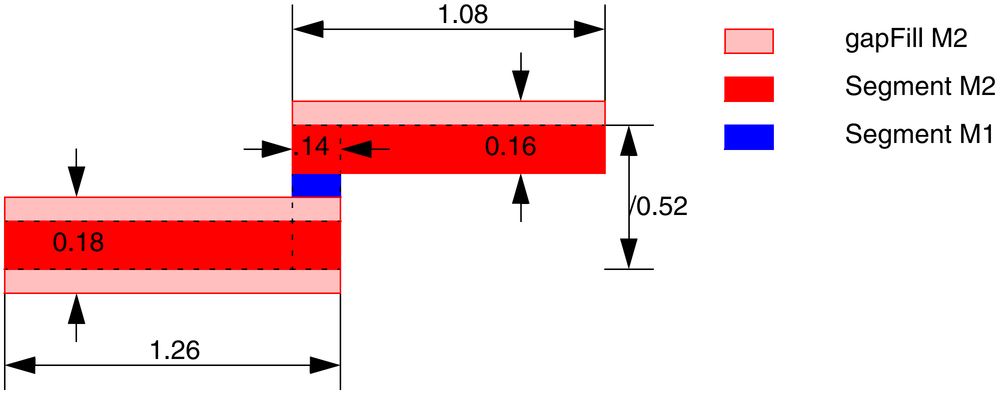
`-region` **{***`f_xlo f_ylo f_xhi f_yhi`***}**

Restricts reporting to the area given by the lower-left (*`f_xlo`*, *`f_ylo`*) and the upper-right (*`f_xhi`*, *`f_yhi`*) bounding box coordinates.

`-set` *`d_setObj`* Reports on nets in the given set.

Example

The following example shows how wire widths are reported.



For this example, the following would be reported for the three route segments:

```
report_wire_width_stats -set [get_selection_set]
NRS = Number of Route Segments
LRS = Length of Route Segments
+-------+-----------+-----------+-----------+-----------+
| Layer |      0.14 |      0.16 |      0.18 |    Total  |
+-------+-----------+-----------+-----------+-----------+
|       |NRS    LRS |NRS    LRS |NRS    LRS |NRS    LRS |
+-------+-----------+-----------+-----------+-----------+
|M1     |  1     0.52|  0        0|  0        0|  1     0.52|
|M2     |  0        0|  1     1.08|  1     1.28|  2     2.36|
+-------+-----------+-----------+-----------+-----------+
| TOTAL |  1     0.52|  1     1.08|  1     1.28|  3     2.88|
+-------+-----------+-----------+-----------+-----------+
The average width is .165
```

create_map_coloring

```
create_map_coloring
     -name s_coloringName
     -type {HSV
               [[ -startHue i_0to360 ][ -stopHue i_0to360 ]
                [ -startSaturation i_0to255 ][ -stopSaturation i_0to255 ]
                [ -startLuminance i_0to255 ][ -stopLuminance i_0to255 ]
                [ -startValue f_value ][ -stopValue f_value ] ]
          | threshold
                [ -ranges {{f_startValue f_stopValue i_red i_green i_blue}…} ]}
```

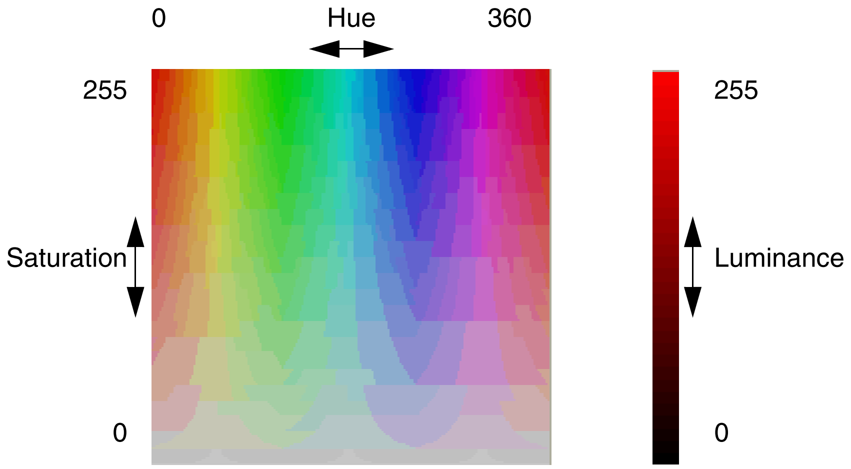
Creates map colorings to use when displaying color maps.

Two types of color mapping are available: `HSV` and `threshold`.

- HSV (Hue Saturation Value)

  HSV color mapping represents color as a continuous change of hue, saturation and luminance (or brightness) as the data value increases. You specify the range for each of these color components. By default, the computed range of data values is mapped to each range of given components to determine the color for each map tile. You can optionally limit the range of data values to color.

  - Hue specifies the color from red (0), through yellows, greens, blues, magenta and finally back to red (360), completing the color wheel.
  - Saturation specifies the color intensity from gray (0) to vivid (255). For a consistent saturation level, specify the same values for start and stop saturation.
  - Luminance chooses the color brightness from light (0) to dark (255). For consistent brightness, specify the same values for start and stop luminance .



- Threshold

Threshold color mapping assigns a color, given by red, green and blue (RGB) components, for a range of data values. For more color detail, a greater number of ranges must be specified. The following is a sampling of colors with their RGB representations:

Color	Red	Green	Blue
Red	255	0	0
Magenta	255	0	255
Purple	125	0	255
Blue	0	0	255
Cyan	0	255	255
Green	0	255	0
Yellow	255	255	0
Orange	255	125	0
White	255	255	255

For example, three ranges are given in the following:

```
-ranges {50 60 255 0 0 60 70 0 255 0 70 80 0 0 255}
```

Data in the first range (50-60) are colored red (255 0 0), in the second range (60-70) green (0 255 0), and in the last range (70-80) blue (0 0 255). Data outside of the given ranges (0-50, 80 and above) will not be colored.

Tip

Use the Select Color form to assist you in choosing HSV and RGB settings. To access this form, right-click a layer or object name in the Layer Object Display Panel, then click the Red-Green-Blue color bars on the right hand side of the pop-up. The Select Color form appears. Change the HSV or RGB values on the right side of the form to see the color that is represented by the values chosen.

Arguments

`-name` *`s_coloringName`* Specifies the name for the map coloring.

`-ranges` **{**`{`*`f_startValue f_stopValue i_red i_green i_blue`*`}`**…}**

(For `-type threshold`) Specifies the coloring to use for a range of values. You must specify all five values for each range. One or more groupings of ranges can be given with the following five values in order:

`f_startValue` Specifies the start value for the range.

`f_stopValue` Specifies the stop value for the range.

`i_red` Specifies the red component for the color. This value must be in the range of 0 to 255.

`i_green` Specifies the green component for the color. This value must be in the range of 0 to 255.

`i_blue` Specifies the blue component for the color. This value must be in the range of 0 to 255.

`-startHue` *`i_0to360`*

(`HSV`) Used with `stopHue` to specify the range of colors for the data values. This value must be in the range of 0 to 360 and represents a color of the color wheel that begins with red (0), migrates through yellows, greens, blues, magenta and back to red (360). The default value is 240 (blue).

`-startLuminance` *`i_0to255`*

(`HSV`) Used with `stopLuminance` to specify the range of brightness for the range of data values. Legal values are 0 to 255. The default value is 125.

`-startSaturation` *`i_0to255`*

(`HSV`) Used with `stopSaturation` to specify the range of color intensity for the range of data values. Legal values are 0 to 255. The default value is 255.

`-startValue` *`f_value`*

(`HSV`) Used with `stopValue` to specify the range of data values to color. If this value is not given, the computed minimum value for the data is used.

`-stopHue` *`i_0to360`*

(`HSV`) Used with `startHue` to specify the range of colors for the range of data values. Legal values are 0 to 360. The default value is 0.

`-stopLuminance` *`i_0to255`*

(HSV type) Used with `startLuminance` to specify the range of brightness for the range of data values. Legal values are 0 to 255. The default value is 125. The default value is 125.

`-stopSaturation` *`i_0to255`*

(HSV type) Used with `startSaturation` to specify the range of saturation for the range of data values. Legal values are 0 to 255. The default value is 255.

`-stopValue` *`f_value`*

(HSV) Used with `startValue` to specify the range of data values to color. If this value is not given, the computed maximum value for the data is used.

`-type {HSV|threshold}`

Specifies the type of color mapping to use.

`HSV`	Represents coloring as a continuous change of hue, saturation and luminance (or brightness) as the data value increases.
`threshold`	Assigns a color (given by red, green and blue components) for a range of data values. For more color detail, a greater number of ranges must be specified.

Example

Two color maps, thermalMap and thermalMapBands, are predefined. The following examples illustrate how these would be set using `create_map_coloring`.

- `thermalMap`

```
create_map_coloring -name thermalMap -type HSV -startHue 180 -stopHue 0
-startSaturation 125 -stopSaturation 255 -startLuminance 125 -stopLuminance
225
```

This HSV mapping colors tiles by linearly mapping the minimum and maximum values for tiles to the color map scale starting with blue (180) at the minimum value, through green, and to red (0) at the maximum value. Saturation (opacity) and luminance are scaled in a similar manner.

- `thermalMapBands`

```
create_map_coloring -name thermalMapBands -type threshold -ranges {60 80 0 0
255 80 95 0 255 0 95 100 255 0 0}
```

This threshold mapping colors tiles with values of 60-80 in blue, values of 80-95 in green, and values of 95-100 in red. Tiles with values under 60 are not colored.

Related Information

Tcl Commands	display_color_map

create_map_tile

```
create_map_tile
     -pattern s_patternName
     [ -line
        {horizontal [ -alignment {center | top | bottom} ]
      | {vertical   [ -alignment {center | left | right} ]}
        [ -lineStyle {solidline | dashedline | dottedline | dashdottedline
                   | dashdotdottedline} ]
        [ -lineWidth i_pixel ]
     | [ -rect
           [ -fillStyle {solid | nofill | crosshatch | diagcrosshatch
                      | negdiaghatch | posdiaghatch | horizhatch | verthatch} ]
           [ -scaleFactor f_multiplier ] ]
     [ -opacity i_0to255 ]
```

Creates and sets parameters for a map tile which determines whether a line or a rectangle will be used to represent the status of cells. If neither `-line` nor `-rect` is given, a solid rectangle of the cell size is used.

Arguments

`-fillStyle` *`s_style`*

Used with `-rect` and specifies the fill style for the rectangle tiles as one of the following:

`solid`	■
`nofill`	□
`crosshatch`	▦
`diagcrosshatch`	▩
`negdiaghatch`	▧
`posdiaghatch`	▨

`horizhatch`	▤
`verthatch`	▥

`-line {horizontal | vertical} [ -alignment` *`s_alignment`*`]`

Chooses a line as the pattern type. The line direction must be `horizontal` or `vertical` and determines the valid choices for the alignment of the line with respect to the cell. By default, the line is drawn through the center of the cell.

Line Direction	Valid choices:
`horizontal`	`center`, `bottom`, `top`
`vertical`	`center, left, right`

`-lineStyle` *`s_style`* Chooses the style for the line tile from the following:

`solidline`	━━━━━━
`dashedline`	▬ ▬ ▬ ▬ ▬ ▪
`dottedline`	▪▪▪▪▪▪▪▪▪▪▪▪▪
`dashdottedline`	▬ ▪ ▬ ▪ ▬ ▪ ▬
`dashdotdottedline`	▬ ▪ ▪ ▬ ▪ ▪ ▬

By default, a solid line is used.

`-lineWidth` *`i_pixel`* Specifies the width for a line tile.

`-opacity` *`i_0to255`* Chooses the opacity for a tile where 0 is transparent and 255 is fully opaque.

`-pattern` *`s_patternName`*

Specifies the name to assign to this color map tile pattern. To use this tile pattern in a color map, specify this name for the `-tile` argument in `display_color_map`.

`-rect` Chooses a rectangle as the tile type. Must be used with the `-fillStyle` argument.

`-scaleFactor` *`f_multiplier`*

Sizes the rectangle tile using the multiplier value. If the value is less than 1, the tile area is scaled down from the size of the cell. If the value is greater than 1, the tile area is scaled up from the cell size. By default this value is 1.

Example

The following command creates a map tile named `tilevl` that is a vertical dashed line on the left side of the cell.

```
create_map_tile -pattern tilevl -line vertical -alignment left -lineStyle
dashedline
```

Related Information

Tcl Commands	display_color_map

display_color_map

```
display_color_map
     { {-name s_computationName | -names {s_computationName…} }
          [ -tile s_patternName ]
          [ -coloring s_coloringName ]
          [ -add ]
          [ -composer Compare ]}
     | -off
```

Displays a color map or turns off all color maps. You must specify the computation to color map from either a recent analysis or loaded using load_color_map. In addition, you specify the display features such as the tile and coloring to use, and whether the new color map should replace the existing color maps or be added to the existing color maps.

Using the Layer Object Display Panel visibility settings, you can see a composite map for multiple layers, the color map for a single layer, or you can hide the color mapping on all layers.

Arguments

`-add`

Layers the new color map over existing color maps. If this argument is not given, existing color maps are removed.

`-coloring` *`s_coloringName`*

Specifies the name of the map coloring to use. There are two predefined map colorings: `thermalMap` (an HSV mapping that shows data as a continuous change of color from blue to red as the data value increases) and `thermalMapBands` (a threshold mapping that assigns specific colors to specific ranges of values). You can customize your own coloring using create_map_coloring. By default, `thermalMap` is used.

`-composer Compare`

Displays the difference between the given computation results, instead of the computation results. For example, you can use this to visually compare the congestion results before and after re-routing.

`-name` *`s_computationName`* `| -names` **{***`s_computationName…`***}**

	Specifies the names of an active computation or a loaded computation to color map. Active computations are results for analyses that have already been run in the current session for the active design. Refer to Table 18-1 on page 1074 for the accepted names and the command that must be run for each computation.
	A loaded computation is the name of a previously saved computation that you loaded using load_color_map. These computation names are of the format:
	`viewName_computationName` or *`viewName_computationName_i`*
	For example, `layout_ViaCongestion_2`
`-off`	Removes all color maps from the artwork.
`-tile` *`s_patternName`*	Specifies the name of the map tile (lines or rectangles) to use. By default, a filled rectangle is used. You can customize tiles using create_map_tile.

Table 18-1 lists computation names and the analysis that must be run on the current design to display the respective results with `display_color_map`.

Table 18-1 Color Map Computation Types

Generated by	*`s_computationName`*
`cmp_analysis`	`CustomHotspots`
	`Density`
	`Hotspots`
	`Thickness`
`congestion_analysis`	`HorizontalCellCongestion`
	`VerticalCellCongestion`
	`LeftEdgeCongestion`
	`BottomEdgeCongestion`
	`ViaCongestion`
	`ChipAssembly`

Generated by	*s_computationName*
report_yield	YldCell
	YldOpen
	YldRouting
	YldShort
	YldVia

Example

The following commands show composite color maps for `HorizontalCellCongestion` and `ViaCongestion`.

```
display_color_map -name HorizontalCellCongestion
display_color_map -name ViaCongestion -add
```

The following commands display a color map representing `LeftEdgeCongestion` using the default `thermalMapBands` color map and left-aligned line tiles.

```
create_map_tile -pattern linevl -line vertical -alignment left
display_color_map -name LeftEdgeCongestion -tile linevl -coloring thermalMapBands
```

Related Information

Tcl Commands	create_map_coloring create_map_tile

display_color_map_value

```
display_color_map_value
     [ -font s_font ]
     [ -size i_fontSize ]
     [ -color s_color ]
     [ -rect [ true | false ] ]
     [ -name
          { HorizontalCellCongestion | VerticalCellCongestion | LeftEdgeCongestion
          | BottomEdgeCongestion | ViaCongestion | ChipAssembly
          | Density | Thickness | Hotspots
          | YldCell | YldVia | YldShort | YldOpen | YldRouting
          | s_computationName }]
     [ -regex_style [ true | false ] ]
```

Displays the value of the color map under the cursor until this action is canceled by the `ESC` key. You can optionally choose display characteristics for the value, including font, font size, color and the computation name.

Arguments

`-color` *`s_color`*

Chooses the color for the value. The default is `orange`.

`-font` *`s_fontName`*

Specifies the name of the font to use for displaying the value. The default is `Arial`.

`-name` *`s_computationName`*

Specifies the name of the computation results to display. The name can be an expression with wildcard characters, letting you restrict the search of multiple computation results. The file is scanned for the first computation name that matches the expression. Refer to Table 18-1 on page 1074 for a list of accepted names and "Pattern Matching" on page 134 for guidelines on special characters that can be used. If this argument is not given, the first computation that was added to the color map will be applied.

`-rect [ true | false ]`

If set to `true`, the boundary of the tile is also highlighted in the same color as the color map value and the displayed value is centered in the tile. The default is `true`.

`-regex_style [ true | false ]`

Applies only when `-name` is given. By default and when set `true`, the computation name can be a regular extended expression with wildcards. When set to `false`, glob expressions can be used for the computation name.

`-size` *`i_fontSize`* Specifies the size of the value font.

load_color_map

```
load_color_map
     -file s_fileName
     [ -name { s_computationName } ]
     [ -regex_style [ true | false ] ]
```

Loads a file containing computation results, allowing you to color map results without re-running the analysis.

After loading data with this command, you must use display_color_map to display the color map for the computation results.

Arguments

`-file` *`s_fileName`*

Specifies the name of the file containing the computation results.

`-name` *`s_computationName`*

Specifies the name of the computation results to load. The name can be an expression with wildcard characters, letting you load multiple computations from a single file. The file is scanned for computation names that match the expression. Refer to Table 18-1 on page 1074 for a list of accepted names and "Pattern Matching" on page 134 for guidelines on special characters that can be used. If this argument is not given, all computations in the file are loaded.

To avoid conflict with a current analysis result, a computation loaded by this command is assigned a unique name in the format: *`viewName_computationName`* or *`viewName_computationName_i`* as needed for uniqueness. For example, `layout_ViaCongestion`, or if that already exists, `layout_ViaCongestion_2`, and so on.

`-regex_style [ true | false ]`

By default and when set `true`, the computation name can be a regular extended expression with wildcards. When set to `false`, glob expressions can be used for the computation name.

Example

The following command loads all computations from the file `mycolormaps` and a Space-based Router and Chip Optimizer response is shown.

```
load_color_map -file mycolormaps
The computation 'layout_ViaCongestion' was loaded
The computation 'layout_VerticalCellCongestion' was loaded
```

In this case, the file contained two computations, `ViaCongestion` and `VerticalCellCongestion`. To display these computations using default settings, you would use the following commands:

```
display_color_map -name layout_ViaCongestion
display_color_map -name layout_VerticalCellCongestion
```

Related Information

Tcl Commands	save_color_map unload_color_map

save_color_map

```
save_color_map
     -file s_fileName
     [ -name { HorizontalCellCongestion | VerticalCellCongestion
                  | LeftEdgeCongestion | BottomEdgeCongestion | ViaCongestion
                  | ChipAssembly | Density | Thickness | Hotspots
                  | YldCell | YldVia | YldShort | YldOpen | YldRouting } ]
     [ -regex_style [ true | false ] ]
```

Saves computations for currently displayed color maps to a file, allowing you to color map results at a later time without running the analyses. Only computations that are currently displayed by display_color_map can be saved.

Arguments

`-file` *`s_fileName`*

Specifies the name of the file to save the computations to.

`-name` *`s_computationName`*

Specifies the name of the computation results to save. The name can be an expression with wildcard characters, letting you save multiple computations to a single file. The currently displayed color maps are scanned for computation names that match the given string. Refer to Table 18-1 on page 1074 for a list of possible base names and "Pattern Matching" on page 134 for guidelines on special characters that can be used. If this argument is not given, all currently displayed computations are saved.

`-regex_style [ true | false ]`

By default and when set `true`, the computation name can be a regular extended expression with wildcards. When set to `false`, glob expressions can be used for the computation name.

Return Value

The following command saves all of the currently displayed computations to file `colormap` and the Space-based Router and Chip Optimizer response is shown.

```
save_color_map -file colormap
The computation 'HorizontalCellCongestion' was saved.
The computation 'VerticalCellCongestion' was saved.
The computation 'ViaCongestion' was saved.
```

In this case, three computations were saved.

Related Information

Tcl Commands	load_color_map unload_color_map

unload_color_map

```
unload_color_map
     -name s_computationName
     [ -regex_style [ true | false ] ]
```

Unloads one or more computations from memory. Only computations that had been loaded using load_color_map can be unloaded. Unloaded color maps will be removed from the display.

Arguments

`-name s_computationName`

Specifies the name of the computation results to unload. The name can be an expression with wildcard characters, letting you unload multiple computations to a single file. The currently loaded color maps are scanned for computation names that match the given string. Refer to Table 18-1 on page 1074 for a list of possible base names and "Pattern Matching" on page 134 for guidelines on special characters that can be used. If this argument is not given, all currently loaded computation results are unloaded.

`-regex_style [ true | false ]`

By default and when set `true`, the computation name can be a regular extended expression with wildcards. When set to `false`, glob expressions can be used for the computation name.

Example

The following command unloads all currently loaded computations with names ending in `Congestion` and a Space-based Router and Chip Optimizer response is shown.

```
unload_color_map -name *Congestion
The computation 'layout_VerticalCongestion' was unloaded.
The computation 'layout_HorizontalCongestion' was unloaded.
```

Related Information

Tcl Commands	load_color_map save_color_map

19

Technology Commands

This chapter describes the Technology commands. The commands are presented in alphabetical order.

add_custom_via_def

```
add_custom_via_def
     -lib s_libName
     -cell s_cellName
     -view s_view_Name
     -layer1 s_layerName
     -layer2 s_layerName
     -tech_lib s_techName
     -via s_viaName
     [ -no_save ]
```

Marks via cells in the library. Via cells are defined to contain three shapes—the top metal layer, the bottom metal layer and the cut layer that shorts the two metals together. Defining via cells for the extraction process is an optional procedure. If you define via cells, your memory utilization of the loaded GDS-based cellview will be reduced by approximately 20%.

Caution

Use of this command can change the data in your technology database file, `tech.db`***, and should be used with caution because others in your group might also be using the file.***

You must define the layer stack (change_layer) before you can use this command. Once the technology information is properly set, you need not repeat the setup for subsequent sessions. Review and verify the technology information using the tech_info command before you load the design.

Arguments

`-cell` *`s_cellName`*	Specifies the cell name of the via cellview.
`-layer1` *`s_layerName`*	Specifies the bottom layer name.
`-layer2` *`s_layerName`*	Specifies the top layer name.
`-lib` *`s_libName`*	Specifies the library that contains the via cell.
`-no_save`	Prevents the information in this command from being saved to the `tech.db` file. Changes are only made to the library in memory.
`-tech_lib` *`s_techName`*	Specifies the name of the technology library to update with the information in this command.

`-via` *`s_viaName`*	Specifies the name that will be assigned to the via in the technology file. Cadence recommends that you use the same name as the cell name.
`-view` *`s_viewName`*	Specifies the view name of the via cellview.

Example

The following command defines a custom via, `CVIA12`, and adds the definition to the technology file for `newTiny`.

```
add_custom_via_def -tech_lib "newTiny" -via "CVIA12" -layer2 "MET2" -layer1 "MET1"
-lib "newTiny" -cell "CVIA12" -view "layout"
```

Related Information

Tcl Command	change_layer check_custom_via_defs recognize_vias tech_info

add_physical_layer

```
add_physical_layer
     -name s_layerName
     -number i_layerNum
     -material {nWell | pWell | nDiff | nlmplant | plmplant | poly |
     contactlessMetal | metal | cut}
     -mask_number i_maskNum
     [ -no_save ]
     [ -tech_lib s_techName ]
```

Adds a physical layer to the design in the active window. The given layer and number cannot exist in the technology file.

Caution

Use of this command can change the data in your technology database file, `tech.db`***, and should be used with caution because others in your group might also be using the file.***

Arguments

`-mask_number` *`i_maskNum`*

Specifies the mask number. The order of the layer stack is set by the mask number sequence.

`-material`

Specifies the material type for the layer. Valid choices are `nWell`, `pWell`, `nDiff`, `nlmplant`, `plmplant`, `poly`, `contactlessMetal`, `metal` and `cut`.

`-name` *`s_layerName`*

Specifies the name to assign to the layer.

`-no_save`

Prevents the information in this command from being saved to the `tech.db` file. Changes are only made to the library in memory.

`-number` *`i_layerNum`*

Specifies the layer number.

`-tech_lib` *`s_techName`*

Adds the layer to the specified OpenAccess technology library.

Note: Use this argument before opening the design.

change_layer

```
change_layer
     -mask_number i_maskNum
     -name s_layerName
     -tech_lib s_techName
     -material {nWell | pWell | nDiff | pDiff | nImplant | pImplant | poly |
     contactlessMetal | metal | cut}
     [ -x_routing_grid f_grid ]
     [ -y_routing_grid f_grid ]
     [ -thickness f_micron ]
     [ -height f_micron ]
     [ -manufacturing_grid f_micron ]
     [ -new_name s_layerName ]
     [ -no_save ]
     [ -preferred_direction {horizontal | vertical | leftDiag | rightDiag ]
     [ -routing_grid_offset f_micron ]
     [ -routing_grid_pitch f_micron ]
     [ -enabled ]
     [ -routable ]
     [ -max_routing_distance f_micron ]
     [ -max_pickup_distance f_micron ]
     [ -boundary_clearance f_micron ]
```

Renames layers and provides additional information about layers, such as material types and mask numbers. Use this command when you read GDSII stream data that does not contain enough information for physical design tools.

Caution

Use of this command can change the data in your technology database file, `tech.db`***, and should be used with caution because others in your group might also be using the file.***

Arguments

`-boundary_clearance` *`f_micron`*

Specifies the minimum clearance required between the edge of an object and the boundary.

`-enabled` Determines whether the layer will be loaded.

`-height` *`f_micron`* Sets the layer height in microns.

`-manufacturing_grid` *`f_micron`*

Specifies the course for this layer. No geometries for this layer can exist off of this grid.

`-mask_number` *`i_maskNum`* Specifies the mask number. The order of the layer stack is set by the mask number sequence.

`-material` Specifies the material type for the layer. Valid choices are `nWell`, `pWell`, `nDiff`, `pDiff`, `nImplant`, `pImplant`, `poly`, `contactlessMetal`, `metal` and `cut`.

`-max_pickup_distance` *`f_micron`*

Specifies the maximum distance for a single layer route to a pin.

`-max_routing_distance` *`f_micron`*

Specifies the maximum distance for a single layer route.

`-name` *`s_layerName`* Specifies the default layer name assigned by Space-based Router and Chip Optimizer for the GDS-based library. When a GDS file is streamed in, it only contains mask numbers. Space-based Router and Chip Optimizer assigns layer names using the default layer name, followed by the layer number (for example, `L1`).

`-new_name` *`s_layerName`* Specifies a new name to assign to the layer.

`-no_save` Prevents the information in this command from being saved to the `tech.db` file. Changes are only made to the library in memory.

`-preferred_direction` Specifies the preferred direction the router will use. Valid values are `horizontal`, `vertical`, `leftDiag`, `rightDiag`.

`-routable` Indicates whether routes can be added to this layer.

`-routing_grid_offset` *`f_micron`*

Specifies the offset from 0 for the router to use for this layer.

`-routing_grid_pitch` *`f_micron`*

Specifies the grid for the router to use when placing centerlines for this layer.

`-tech_lib` *`s_techName`* Specifies the technology library to use.

`-thickness` *`f_micron`* Sets the layer thickness.

`-x_routing_grid` *`f_micron`*

Specifies where route segments can begin and end, and where vias can be placed on the x-axis.

`-y_routing_grid` *`f_micron`*

Specifies where route segments can begin and end, and where vias can be placed on the y-axis.

Example

The following example renames the third mask with default name `L3` layer to `MET1` of material type `metal.`

```
change_layer -tech_lib "newTiny" -name "L3" -new_name "MET1" -material "metal"
-mask_number 3
```

Related Information

Tcl Command	add_custom_via_def tech_info

check_custom_via_defs

```
check_custom_via_defs
     -tech_lib s_techName
```

Evaluates all custom vias that are defined in the given OpenAccess technology library. Space-based Router and Chip Optimizer checks for the existence of a cellview for each custom via definition in the library. If the cellview exists but is not the appropriate cell type, Space-based Router and Chip Optimizer corrects the cell type in the library.

Arguments

`-tech_lib` *`s_techName`*	Specifies the technology library to check for custom vias.

clean_layer

```
clean_layer
     -lpp {s_lpp …}
     [ -clean_oa ]
     [ -deleteLPP ]
     [ -hierarchical ]
```

Removes all shapes from the occurrence model of the given layer purpose and, optionally, from the corresponding OpenAccess (OA) database and/or all levels of the hierarchy. In addition, the specified layer purpose can be removed from the design.

Arguments

`-clean_oa`	Removes data from the OA database for the given layer purpose.
`-deleteLPP`	Removes the specified layer purpose from the design and reclaims memory used by the layer.
`-hierarchical`	Cleans all levels of the hierarchy that contain the specified layer purpose. By default, only top-level shapes are removed.
`-lpp` **{***s_lpp* **…}**	Specifies the layer purpose to remove shapes/data from.

Example

The following example removes all shapes from the `annotation:viaOpt` purpose of `Metal6`.

```
clean_layer -lpp { Metal6:annotation:viaOpt }
```

Related Information

Tcl Command	add_physical_layer

delete_purpose_map

```
delete_purpose_map
```

Removes all user purpose mappings.

To permanently remove the user purpose map from a design, load the design, invoke delete_purpose_map, then save the design.

Arguments

None

Related Information

Tcl Command	map_purpose report_purpose_map

destroy_derived_cgs

```
destroy_derived_cgs
```

Destroys the transient constraint groups, rule specifications, and derived tracks created for speciality nets in shielding and differential pair routing while working with WSP or track-based flows.

Arguments

None

get_dbu_per_user_unit

```
get_dbu_per_user_unit [ -window_id i_windowID ]
```

Returns the number of DBUs per user unit for the design in the given window or the active window.

Arguments

`-window_id` *`i_windowID`*	Specifies the identifier for the window to report on. If this argument is not given, the active window is used.

Value Returned

`i_count`	Is the number of DBUs per user unit.
`-1`	Indicates that there was an error in the command syntax or that the specified window is not active.

Example

The following example returns the number of DBUs per user unit for the design in window 2.

```
get_dbu_per_user_unit -window_id 2
2000
```

Related Information

Tcl Command	tech_info

get_use_process_rules_only

```
get_use_process_rules_only
```

Returns the rules processing mode.

Arguments

None

Value Returned

`true | false`	If true, the system will consider rules on base layers only, and ignore route specs. If false, the system always obeys rules.

Related Information

Tcl Command	set_use_process_rules_only

map_purpose

```
map_purpose
     {-from_lpp s_lpp | -user_purpose s_purposeName}
     -to_purpose s_purposeName
```

Maps a user purpose or a layer purpose in the design to a predefined Space-based Router and Chip Optimizer purpose.

When a design is loaded, all user purposes map to `drawing`. If the user purpose name is `bob`, it is mapped to `bob:drawing`, and, by default, is ignored. Use `map_purpose` to make an explicit mapping to a predefined Space-based Router and Chip Optimizer purpose, or to map one layer purpose to another purpose.

The mapping must be set prior to loading the design and can be saved with the design as a `CatenaUserPurposeMap` property. Once saved, the mapping will automatically be loaded when the design is loaded, and can be removed using delete_purpose_map.

Mappings set using the `map_purpose` command apply to all designs that are subsequently loaded. If you load a design that has the `CatenaUserPurposeMap` property set, the mappings from that property will apply only to that design and will take precedence over mappings set using the `map_purpose` command. To prevent saved mappings from being loaded with a design, issue the following before loading the design:

```
setvar db.map_user_purposes false
```

For a listing of the predefined Space-based Router and Chip Optimizer purposes, use

```
help map_purpose
```

Arguments

`-from_lpp` *`s_lpp`*

Maps a layer purpose to a predefined Space-based Router and Chip Optimizer purpose. The layer purpose must be in the following format: *`layer`*:*`purpose`*. For example, `M2:blockage.`

`-to_purpose` *`s_purposeName`*

Specifies the predefined Space-based Router and Chip Optimizer purpose to map the design’s user purpose to.

`-user_purpose` *`s_purposeName`*

Specifies the user purpose in the design to be mapped.

Example

The following example will cause objects on the `Unknown` user purpose of the design to appear as objects on the `fill` purpose in the workspace and as `Unknown:fill` in the Layer Object Display Panel.

```
map_purpose -user_purpose Unknown -to_purpose fill
```

Related Information

Tcl Command	delete_purpose_map report_purpose_map

recognize_vias

```
recognize_vias
     -lib s_libName
     [ -append {true | false} ]
     [ -cmd_file s_fileName ]
     [ -force ]
     [ -ignore_case ]
     [ -ignore_layers {s_layerName …} ]
     [ -no_save ]
     [ -name_pattern s_viaExpr ]
     [ -remove_existing ]
     [ -view s_viewName ]
```

Performs automatic via recognition by evaluating every cellview in the given library. A custom via definition is added to the technology file for each cellview that meets the via conditions listed below. If the cellview type is not a via cellview, the cell type is changed in the library.

Caution

Use of this command can change the data in your technology database file, `tech.db`***, and should be used with caution because others in your group might also be using the file.***

The `recognize_vias` command requires that layer definitions (change_layer) exist in the OpenAccess technology file to mark the material type and the mask number appropriately. In addition, the technology file may not already contain custom via definitions. Review and verify the technology information using the tech_info command before you load the design.

When you use this command, every cellview in the technology library is evaluated for the following conditions:

- The cellview contains three layer purpose pair headers.
- There is one cut layer.
- The other two layers are the above and below layers (based on their mask numbers) to the cut layer.
- The above layer is of material type "wire".
- The below layer is of material type "wire" or "poly".
- The above layer has one shape.
- The below layer has one shape.

The command has the following limitations:

- Does not check that the above and below shapes enclose the cut layer shapes.
- Does not define vias to P or N diffusion.

Arguments

`-append {true|false}`
If true, appends to a list of existing via definitions. If false, the command will check for existing via definitions in the technology library and will not run if via definitions already exist.

`-cmd_file` *`s_fileName`*
Writes a Tcl script of add_custom_via_def commands for each cellview that meets the via criteria instead of updating the technology file. This allows you to evaluate the vias found and remove any that have been improperly identified. You can source the script to add the via definitions.

`-force`
Forces vias to be recognized even when the cut layer is not completely within the two metal layers. By default, the via is only recognized when the cut layer is completely within the two metal layers.

`-ignore_case`
If name_pattern is set, then case is ignored when matching cellnames.

`-ignore_layers` **{***`s_layerName`* **…}**
Ignores the given layers during evaluation.

`-lib` *`s_libName`*
Specifies the library to scan.

`-name_pattern` *`s_viaExpr`*
Automatically recognizes cellviews whose names match the given expression as via cells.

`-no_save`
Prevents custom via definitions and cellview changes from being saved to the library and the technology file. When you exit Space-based Router and Chip Optimizer, a dialog will appear that lets you save these changes.

`-remove_existing`
Removes all via definitions before starting.

`-view` *`s_viewName`*
Specifies the view name in the given library to scan for via cells. By default, the `layout` view is scanned.

Example

The following command creates a file, `via_def_cmds.tcl`, of `add_custom_via_def` commands that can be sourced to create custom via definitions for the `newTiny` library.

```
recognize_vias -lib "newTiny" -cmd_file via_def_cmds.tcl
```

Related Information

Tcl Command	add_custom_via_def tech_info

report_purpose_map

```
report_purpose_map
     [ -tcl [ true | false ] ]
```

Reports all user purpose map overrides. The default mapping is not reported. Can optionally output the map_purpose commands for the current user purpose map overrides.

Arguments

`-tcl [true	false]`	When set to `true`, outputs the mappings as map_purpose commands. When set to `false` and by default, the mappings are reported as *`designPurpose`*->*`mapPurpose`*.

Example

The following example shows the command that is issued and the output.

```
report_purpose_map
cover->fill
0
report_purpose_map -tcl
map_purpose -user_purpose cover -to_purpose fill
0
```

Related Information

Tcl Command	delete_purpose_map map_purpose

save_tech

```
save_tech
```

Saves the technology database.

Arguments

None

set_net_implementation_purpose

```
set_net_implementation_purpose
     -net s_netName | -set d_setObj
     -purpose s_purposeName
```

(Virtuoso® Routing IDE only) Specifies the implementation purpose for the given net or nets in the given set. All new shapes for the given nets that are created by the router will be saved on the specified implementation purpose. Vias will be created using cdsVias, so these must be defined in the technology file.

> **Note:** This command is needed only if you will load the saved design in Virtuoso Layout Suite L (VLS L). If you do not set the implementation purpose before routing using Space-based Router and Chip Optimizer, important purpose-based information can be lost when you load in VLS L.

Example

In this example, `hv` shapes have a voltage swing between 0.0 and 3.3, as specified in the `techPurposes` section of the technology file.

```
controls (
   techVersion ("1.0")
)
techPurposes(
;(PurposeName Purpose# Abbreviation)
;(--------------------------------)
;User-defined Purposes:
(hv 13 hv 'sigType "digital" 'parent "drawing" voltageRange (0.0 3.3))
)
```

To ensure that new routing that is created for `netA` is saved on the `hv` purpose, use

```
set_net_implementation_purpose -net netA -purpose hv
```

set_tech_references

```
set_tech_references
     -tech_lib s_techName
     {-clear | -ref_tech_libs {s_techName…}}
```

Establishes or resets references from one technology library to other technology databases.

OpenAccess allows for libraries to refer to technology information that exists in one or more technology libraries. For example, a base technology library could define a process with seven interconnect layers and another library could refer to that base technology information and add three more interconnect layers for a total of ten layers. This is known as an incremental technology library.

Use `-ref_tech_libs` to establish references for an incremental technology database. The technology databases must be defined in the `lib.defs` file.

Use `-clear` to reset an incremental database by removing all references from the base technology library.

Use tech_info to verify the current technology information.

Note: Establishing technology references may generate warnings or errors if duplicate information exists (for example, when a metal layer is defined in more than one library).

Arguments

`-clear` Removes all references from the base technology database.

`-ref_tech_libs` **{***s_techName…***}**

Establishes references from the base technology library to the technology databases in the list.

`-tech_lib` *s_techName* Specifies the name of the base technology library.

Example

The following commands remove all references from the `techLib` database, then establishes a reference for the `sq1` technology database to `techLib`.

```
set_tech_references -tech_lib techLib -clear
set_tech_references -tech_lib techLib -ref_tech_libs { sq1 }
```

Related Information

Tcl Commands	tech_info

set_use_process_rules_only

```
set_use_process_rules_only
     -process_only [ true | false ]
```

Sets the mode for processing rules.

Arguments

`-process_only [ true | false ]`	
	If true, the system will consider rules on base layers only, and ignore route specs. If false, the system always obeys rules. This is the default.

Related Information

Tcl Command	get_use_process_rules_only

set_via_stackability

```
set_via_stackability
     -via s_viaName
     -up {true|false}
     -down {true|false}
     [ -tech_lib s_techName ]
     [ -bottom_of_stack [ true | false ] ]
     [ -top_of_stack [ true | false ] ]
```

Sets the direction that the specified via can be stacked. By default, vias can be stacked both upward and downward.

Arguments

`-bottom_of_stack [ true | false ]`

Sets the bottomOfStack attribute on the via.

`-down {true|false}` Specifies whether the via can be stacked downward.

`-tech_lib` *`s_techName`* Specifies the technology library for the via.

`-top_of_stack [ true | false ]`

Sets the topOfStack attribute on the via.

`-up {true|false}` Specifies whether the via can be stacked upward.

`-via` *`s_viaName`* Specifies the via.

Example

The following example enables only upward stackability for via `M6_M5`.

```
set_via_stackability -via M6_M5 -up true -down false
```

Related Information

Tcl Command	report_via_stackability tech_info

sync_layer

```
sync_layer
     -lpp s_lpp
     -output_lpp s_lpp
     [ -dont_clean_shapes ]
     [ -no_save ]
```

Copies all shapes from one layer purpose to another layer purpose and to the technology library. This command is typically used to sync the OA database and the technology library with the occurrence model.

Arguments

`-dont_clean_shapes`	Specifies that shapes already on the output layer purpose be retained. If this argument is not specified, all shapes are removed from the output layer purpose before the copy is performed. Use of this argument is only recommended for boolean layers.
`-lpp` *`s_lpp`*	Specifies the layer purpose to copy.
`-no_save`	Prevents the data from being written to the technology library.
`-output_lpp` *`s_lpp`*	Specifies the output layer purpose.

Related Information

Tcl Command	add_physical_layer

tech_info

```
tech_info
     [ -file s_fileName ]
     [ -tech_lib s_techName ]
     [ -route_specs ]
     [ -default_route_spec ]
     [ -via_defs ]
     [ -redundant_via_defs ]
```

Outputs a summary of the technology information for the layers of a library or the active cellview to the Space-based Router and Chip Optimizer Transcript area.

Arguments

`-default_route_spec`	Outputs the technology information for the global net default route spec.
`-route_specs`	Outputs the technology information for all of the route specs.
`-file` *`s_fileName`*	Outputs the requested technology information to the specified file instead of the Transcript area.
`-redundant_via_defs`	Outputs redundant via masters.
`-tech_lib` *`s_techName`*	Specifies the name of the OpenAccess library to retrieve the technology information from. If this argument is not specified, the technology information for the active cellview is output.
`-via_defs`	Outputs via definitions.

Example

The following command outputs the `newTiny` library information to the Transcript area.

```
tech_info -tech_lib "newTiny"
```

The following command outputs the library information for the active cellview to the Transcript area.

```
tech_info
```

Related Information

Tcl Command	add_custom_via_def recognize_vias

20

System Commands

This chapter describes the System commands. The commands are presented in alphabetical order.

cancel_command

```
cancel_command
```

Cancels the current interactive mode and sets the mouse command mode to `select`. This command is invoked whenever you press the `Esc` key or choose `Cancel` while using the interactive wire editor.

Arguments

None

define_key_binding

```
define_key_binding
     -command s_cmdText
     -key s_key
```

Defines bindkeys to give you quick access to the functions you use most often.

Arguments

`-command` *`s_cmdText`*	Specifies the Tcl command string that is invoked when the bindkey is pressed.
`-key` *`s_key`*	Specifies the bindkey. You can use alphanumeric keys, function keys (`F1` through `F10`) and special keys (`ALT` and `CTRL`) in combinations by using the plus (+) sign.

Example

The following command provides a quick way to display the design statistics by creating a bindkey. After you invoke this command, each time you press the `S` key, the design statistics are output to the Transcript area.

```
define_key_binding -command report_design_stats -key s
```

The following command sets a bindkey to display the statistics for the net in the selected set. After this command is invoked, each time the CTRL key is pressed with the `S`key, the statistics are output to the Transcript area.

```
define_key_binding -command "report_net_stats -set [get_selection_set]" -key
"CTRL+s"
```

Related Information

Tcl Command	undefine_key_binding
Menu Command	*File—Read Application Files—Bind Keys...*

disable_multithreading

```
disable_multithreading
```

Disables multi-threading and releases licenses that were reserved for that purpose.

Arguments

None

Related Information

Tcl Command	enable_multithreading

enable_multithreading

```
enable_multithreading
     [ -threads i_count ]
```

Reserves licenses for multi-threading which allows Space-based Router and Chip Optimizer to use multiple processors in one workstation to accelerate throughput. You must have additional licenses for the additional processors.

Licenses reserved by issuing this command are globally available to all Space-based Router and Chip Optimizer commands during the session. Multi-threading commands can also override the number of threads available globally in the session. If a command requests more threads than the number of available session threads, Space-based Router and Chip Optimizer will attempt to check out additional licenses, as needed, and when the command finishes, the additional licenses will be released.

Number of Threads	Number of Licenses Required
2-4	One additional (for a total of 2)
5-8	Two additional (for a total of 3)
9 and up	One additional for each group of 4 processors

Arguments

`-threads` *`i_count`* Specifies the number of threads, or processors, to use for multi-threading. If this argument is not specified, the default is the maximum number of processors in the workstation.

Related Information

Tcl Command enable_multithreading

get_entry_lpp

```
get_entry_lpp
     [ -window_id i_windowID ]
```

Returns the layer purpose pair (LPP) that is the current entry layer for the given window or the active window.

Arguments

`-window_id` *`i_windowID`*	Specifies the window to use. If not specified, the entry layer for the active window is returned.

Value Returned

`s_lpp`	Indicates the LPP for the given window.

Example

The following example requests the entry layer for the current window and shows the response from Space-based Router and Chip Optimizer.

```
get_entry_lpp
"met5:wire:detail"
```

Related Information

Tcl Command	set_entry_lpp

get_license_info

```
get_license_info
     [ -popup ]
```

Outputs the product licensing information.

Arguments

`-popup`	Displays the License Info pop-up with the licensing status information. By default, the information is displayed in the Transcript area.

Example

```
get_license_info
```

This example returns the licensing information in the Transcript area. For example,

```
"Your system is using a FlexLM license."
```

has_clients

```
has_clients
```

Queries the system for clients.

Arguments

None

Value Returned

`0`	No clients exist.
`1`	Clients exist.

Related Information

Tcl Command	write_client

hide_application

```
hide_application
```

Hides the Space-based Router and Chip Optimizer Graphical User Interface.

Arguments

None

Related Information

Tcl Command	raise_application

in_RIDE_mode

```
in_RIDE_mode
```

Used in Tcl scripts to determine whether the script is being processed by Virtuoso Routing IDE. This is useful for scripts that are shared by Virtuoso Routing IDE and Cadence Space-based Router to prevent Space-based Router from trying to run Virtuoso Routing IDE-specific commands, such as checkpoint, ge_to_rde_selection, and hsm_to_rde_selection.

Arguments

None

Value Returned

`true`	Script is being processed by Virtuoso Routing IDE.
`false`	Script is not being processed by Virtuoso Routing IDE.

install_path

```
install_path
```

Returns the path to the Space-based Router and Chip Optimizer installation. This is useful when coding Tcl scripts.

Arguments

None

Value Returned

`s_installation_dir`	Is the installation path.

Example

The following command creates a variable, `install_dir`, that contains the installation path.

```
set install_dir [install_path]
puts $install_dir
"/net/mysun/home/mybuild"
```

is_advanced_node_version

```
is_advanced_node
```

Specifies whether this version of Space-based Router and Chip Optimizer supports advanced node constraints.

For more information on advanced node constraints and parameters, refer to *Advanced Node (ICADV12.1 only): Constraints*.

Arguments

None

Value Returned

`true`	This version supports advanced node constraints
`false`	This version does not support advanced node constraints.

Example

The following command determines whether advanced node constraints are supported by this release.

```
is_advanced_node_version
```

is_graphical

```
is_graphical
```

Specifies whether Space-based Router and Chip Optimizer is running in graphics mode. By default, Space-based Router and Chip Optimizer runs in graphics mode except when started with the `-noGraph` option. In non-graphics mode, graphics do not appear on the screen and messages are transcripted to the console window in which you started. You can perform all normal operations, except graphics output commands, such as `export_jpeg`, but can only view transcripted messages.

Arguments

None

Value Returned

0	Running in non-graphics mode. Graphics output commands cannot be run when in this mode.
1	Running in graphics mode.

is_interrupted

```
is_interrupted
     {-check | -start | -stop}
```

Controls the interrupt checker. You use the interrupt checker in Tcl scripts to break out of sequences and loops.

Arguments

`-check`	Checks for a user interrupt. If the `ESC` key has been pressed, the command returns `true`.
`-start`	Starts the interrupt checker.
`-stop`	Stops the interrupt checker.

Value Returned

`true | false`	For `is_interrupted -check,` specifies whether the `ESC` key has been pressed when the interrupt checker is active.

Example

The following example illustrates how to start, check, and stop the interrupt checker in a script.

```
# start the interrupt checker
is_interrupt -start

while {<condition is true>} {
   # has the ESC key been pressed?
   if {[is_interrupted -check]} {
      # user interrupted loop. Stop the interrupt checker and break out.
      is_interrupted -stop
      return;
   }
   # No interrupt, continue with normal loop processing here
}

# stop the interrupt checker
is_interrupted -stop
```

Related Information

Tcl Commands	pause

message_box

```
message_box
     { -critical | -info | -warning | -question }
     -msg s_text
     -title s_text
     [ -button1 s_name ]
     [ -button2 s_name ]
     [ -button3 s_name ]
```

Outputs a user-defined dialog box to the screen, then returns a string to indicate which button was selected in the dialog. This is useful when operating remotely.

Arguments

`-button1` *`s_name`*	Specifies the label for the leftmost button in the dialog. *`buttonStr1`* can be one of the following: `NoButton`, `Ok, Cancel`, `Yes`, `No`, `Abort`, `Retry`, `Ignore`, `YesAll`, or `NoAll`. The default for this button is `OK`.
`-button2` *`s_name`*	Specifies the label for the center button in the dialog. *`buttonStr2`* can be one of the following: `NoButton`, `Ok, Cancel`, `Yes`, `No`, `Abort`, `Retry`, `Ignore`, `YesAll`, or `NoAll`. The default for this button is `NoButton`, causing this button not to be drawn for this position in the dialog.
`-button3` *`s_name`*	Specifies the label for the rightmost button in the dialog. *`buttonStr3`* can be one of the following: `NoButton`,`Ok`, `Cancel`, `Yes`, `No`, `Abort`, `Retry`, `Ignore`, `YesAll`, or `NoAll`. The default for this button is `NoButton`, causing this button not to be drawn for this position in the dialog.
`-critical | -info | -warning | -question`	
	Indicates the type of dialog box to output.
`-msg` *`s_text`*	Specifies the message to output in the dialog box.
`-title` *`s_text`*	Specifies the title to use for the dialog box.

Value Returned

`s_name`	Indicates the label of the button that was selected.

num_processors

```
num_processors
     [ -avail ]
```

Returns the number of processors in the system or the number of available processors. The total number of processors is also output when you use the show_system_info command. You can use `num_processors` when issuing commands, such as check_space, that can be multi-threaded.

Arguments

`-avail`	Returns the number of processors that are not currently in use. When this argument is not given, the total number of processors is returned.

Value Returned

`i_count`	Indicates the number of processors.

Related Information

Tcl Commands	check_space show_system_info

pause

```
pause
     [ -delay i_delay ]
```

Pauses the system for a specific interval or until the *Continue* button is clicked.

Arguments

`-delay` *`i_delay`*	Pauses the system for an interval of time. A value of 1000 is approximately one second. If this argument is not given, you must click the *Continue* button to resume operation.

raise_application

```
raise_application
```

Deiconifies or brings to the front the Space-based Router and Chip Optimizer application.

Arguments

None

set_entry_lpp

```
set_entry_lpp
     -lpp s_lpp
     [ -window_id i_windowID ]
```

Sets the current entry layer for a specified window or the active window.

Arguments

`-lpp` *`s_lpp`*	Specifies the layer purpose pair (LPP) to set the entry layer to.
`-window_id` *`i_windowID`*	Specifies the window to use. If not specified, the active window is used.

Value Returned

`0`	The entry lpp has been set to the desired lpp.
`-1`	The entry lpp was not set. Check the format and syntax of the lpp argument.

Example

The following command sets the entry layer for the current window to `met1:wire:detail`.

```
set_entry_lpp -lpp met1:wire:detail
```

Related Information

Tcl Command	get_entry_lpp

setvar

```
setvar
     [-journal_changes {true | false}]
     s_variable_name
     s_value
```

Sets the value of a Space-based Router and Chip Optimizer environment variable. Optionally specifies whether the command is echoed back to the transcipt window whenever the environment variable is set. By default, the command is journaled.

Arguments

s_value	Specifies the value for the variable.
s_variable_name	Specifies the environment variable.
`-journal_changes {true	false}`	
	Specifies whether the command is echoed back to the transcipt window whenever the environment variable is set. By default, the command is journaled.

Example

```
setvar gui.scale 2
```

Sets the minimum shape size for display to 2.

```
setvar -journal_changes true gui.scale 7
```

Sets the minimum shape size for display to 7 and causes the servar command to be journaled whenever the `gui.scale` setting is changed.

```
setvar -journal_changes false gui.scale
```

Prevents `gui.scale` environment variable settings from being journaled.

Related Information

Documentation	“Space-based Router Environment Variables” on page 1142

show_system_info

```
show_system_info
```

Outputs the following information to the Transcript area:

- Machine name
- OS Version
- Architecture
- Number of processors

Arguments

None

Example

The following is an example of output information.

```
# Machine name:         Freddy
# OS Version:           Windows XP 5.1.2600
# Architecture          Intel Pentium 0x00000207
# Number of Processors: 1
```

show_version

```
show_version
```

Outputs the software version number.

Arguments

None

Example

The following is an example of output information.

```
"7.2r287"
```

toggle_alt_key

```
toggle_alt_key
```

Simulates pressing and releasing the `ALT` key. Each time this command is invoked, the state of the `ALT` key is toggled between the `up` and `down` position. Use this command if your system setup does not recognize the `ALT` key+mouse combination used for dynamic panning (as with applications using VNC). For ease of use, assign a bindkey to the command and toggle the `ALT` key state by pressing the bindkey when the workspace is active.

Arguments

None

Example

The following example assigns the `T` key to the `toggle_alt_key` command using the `define_key_binding` command.

```
define_key_binding -command toggle_alt_key -key t
```

You can also assign the bindkey in the bindkey file that is read on system startup.

Related Information

Tcl Command	define_key_binding

undefine_key_binding

```
undefine_key_binding
     -key s_key
```

Removes the given key from the bindkey list.

Arguments

`-key` *`s_key`*	Specifies the bindkey to remove.

Example

This example removes the "`CTRL+f`" key from the bindkey list.

```
undefine_key_binding -key "CTRL+f"
```

Related Information

Tcl Command	define_key_binding

write_client

```
write_client
     -txt s_text
     -all | -client i_socket
```

Writes a string to all client sockets or a specific client socket.

Arguments

`-all`	Writes the string to all clients sockets.
`-client` *`i_socket`*	Writes the string to a specific client socket.
`-txt` *`s_text`*	Specifies the string to write.

Related Information

Tcl Command	has_clients

21

Environment Variables

Environment variables control various aspects of the environment including the GUI style and display options.

This chapter describes the following:

Space-based Router Environment Variables

Space-based Router environment variables are described in these sections:

- Database Environment Variables
- Graphical User Interface Environment
- Command Environment
- Interactive Environment
- Router Environment
- Power Router Environment
- Verify Environment

Database Environment Variables

You should set these variables prior to reading a design to control what is loaded.

Variable Name	Description	Type	Default
`db.abstract_view_name`	Use with `db.map_layout_to_abstract_view true` to specify the name of the view to search for when building instances.	String	`abstract`
`db.allow_pin_to_pin_space_checking`	If `true`, makes the checker flag violations between pin shapes.	Boolean	`false`

Variable Name	Description	Type	Default
`db.check_constraints_basics`	If true, performs basic constraint checks to prevent fundamental routing issues, including checks for `minWidth`, `minSpacing`, and the existence of at least one via between routing layers. Messages will be printed on load or when routing and the command will exit. This is useful when incomplete custom data is used.	Boolean	`false`
`db.clearance_based_zoning`	If `true`, automatically estimates a worst-case maximum clearance of 3x `minSpacing` for spacing requirements. When `false`, the spacing constraint settings are used.	Boolean	true

Variable Name	Description	Type	Default
`db.connect_array_insts`	Provides connectivity and routability support for array instances. ■ `routable`: Only routable array instances (with master having at least one terminal) are flattened and connected. ■ `all`: All array instances are flattened and routable instances are connected as well. ■ `none`: array instances are not flattened and connected by default. An appropriate warning message appears in this case. ■ `never`: this value is similar to `none` but does not display a warning message.	Cyclic	`none`
`db.connect_IO_pin_shapes`	When `true`, the router will connect all pin shapes of the same top-level IO terminal (pin). When `false`, the router will connect only one pin shape for each top-level IO terminal.	Boolean	`false`

Variable Name	Description	Type	Default
`db.connect_mustjoin_pins`	When `true`, the router forces connection to must join pins even though there might be shapes on lower layers that connect them.	Boolean	`false`
`db.create_inst_terms_in_subdesign`	When `true`, create_instance will create instances of subdesigns with all the instance terminals. By default, the instance terminals for subdesigns are not created and results in faster processing.	Boolean	`false`
`db.default_rule_spec_name`	Specifies the name of the default constraint group and default taper. This is required if there is no LEFDefaultRouteSpec. Virtuoso typically sets this to virtuosoDefaultSetup.	String	`""`
`db.design_type`	Specifies the design style that is used in routing, which is implicitly set in the routing form or when loading the design (see also: read_db). You do not need to set the design style directly. It is one of three values: `devicelevel`, `asic`, or `chipassembly`.	String	`""`

Variable Name	Description	Type	Default
`db.ead_mode`	When `false`, CSR and VSR ignore any non-poly, non-metal, and non-via shapes in a via cell. For example, consider a via that contains a diffusion layer shape. In this case, the diffusion shape is ignored for the purpose of design rule checking and not displayed in the CSR window. However, when `true`, the diffusion shapes are not ignored. **Note:** The variable is enabled automatically when you load an EAD setup inside Virtuoso.	Boolean	`false`
`db.enable_blockage_constraints`	Determines whether blockage constraints are loaded with your design. You must set this variable before loading the design. Designs that have already been loaded will not be affected by a change to this variable.	Boolean	`true`
`db.enable_create_derived_vias`	Controls whether standard vias are automatically created. This should be used only by Virtuoso users before assisted routing. When `false`, standard vias will not be automatically generated and the vias in `validRoutingVias` will be used.	Boolean	`true`

Variable Name	Description	Type	Default
`db.enable_effective_width_on_blockages`	Specifies a list of cell types for a cell or a design. When set, it applies `effectiveWidth` on `oaLayerBlockages` on cells or designs of the specified cell types provided the blockages do not have the `effectiveWidth` or spacing attribute already set. The valid values of `cellType` are: `nil`, `none`, `block`, `blockRing`, `cover`, `coverBump`, `pad`, `padSpacer`, `core`, `coreSpacer`, `coreAntenna`, `corner`, `softMacro`, `via`, `blockBlackBox`, `padAreaIO`, and `coreWellTap`. **Note:** The environment variable setting is effective only if specified before the design is loaded. Updating the environment variable setting after loading the design does not affect the already loaded design.	Stringlist	`""`

Variable Name	Description	Type	Default
`db.enable_min_spacing_on_blockages`	Specifies a list of cell types for a design. When set, it applies minimum spacing on `oaLayerBlockages` on cells or designs of the specified cell types provided the blockages do not have the `effectiveWidth` or minimum spacing already set. The valid values of `cellType` are: `nil`, `none`, `block`, `blockRing`, `cover`, `coverBump`, `pad`, `padSpacer`, `core`, `coreSpacer`, `coreAntenna`, `corner`, `softMacro`, `via`, `blockBlackBox`, `padAreaIO`, and `coreWellTap`. **Note:** If the same `cellType` is provided for `enable_effective_width_on_blockages` and `enable_min_Spacing_on_blockages`, then `enable_effective_width_on_blockages` takes precedence. Thus, only `effectiveWidth` is applied on `oaLayerBlockage` and `minSpacing` is ignored. The environment variable setting is only effective when specified before the design is loaded. Updating the environment variable setting after loading the design does not affect the already loaded designs.	Stringlist	`""`

Variable Name	Description	Type	Default
`db.ignore_routing_grid_check_for_via_metal_layers`	Ignores the routing grid check for metal layers with via shapes, if the corresponding cut shape is enclosed by non-via metal. For example, if the specified metal list is `M2 M3 M4 M5`, then the `M1-V1-M2` via is not ignored. The ignored vias are `M2-V2-M3`, `M3-V3-M4` and `M4-V4-M5`. If you want to also ignore `M1-V1-M2`, then you need to specify `M1` in the list.	Stringlist (a list of metal layers)	`""`
`db.load_as_detailed_abstract`	Assigns custom spacing to blockages created at higher hierarchy levels for routes and nets at the specified level when the design is loaded.	Integer	0
`db.load_blockages`	Determines whether to load standard cell blockages that originated in LEF.	Boolean	`true`
`db.load_blockages_start_level`	Specifies the level to start loading blockages from.	Integer	`0`
`db.load_blockages_stop_level`	Specifies the level to stop loading blockages from.	Integer	`ctuMAX32`
`db.load_cdsDefTech`	If your design does not rely on the layer definitions in the cdsDefTechLib, set this to `false` to avoid "Technology database conflict" messages during file load.	Boolean	`true`

Variable Name	Description	Type	Default
`db.load_core_blockages`	Determines whether to load core cell blockages (usually top-level) that originated in the blockage section in DEF. If your design has blockage polygons at the top level that cause many false spacing violations to metal routes, set this environment variable `false` before loading the design.	Boolean	`true`
`db.load_lower_level_wire_shapes_as_blockages`			
	Loads level-2 shapes and below as blockages. Also, these shapes are not used for top-level connections. In case, the pin is small and the shapes on lower layers are required to be considered as routing targets, then set the variable to false. In this case, the extractor is used to extract the connectivity for the shapes.	Boolean	`true`
`db.load_prboundary_shape_as_boundary`			

Variable Name	Description	Type	Default
	Identifies the layer-purpose pair to be used for creating prBoundary in lower-level (non-top) designs, provided they do not have the `oaPRBoundary` object. **Note:** Coordinates of one of the shapes belonging to user-defined layer-purpose pair is used to create prBoundary. You can specify layer and purpose name for prBoundary shape using the following two environment variables ■ `db.prboundary_layer_name` ■ `db.prboundary_purpose_name` **Note:** These variable are only effective when the design is loaded after appropriately setting them.	Boolean	`false`
`db.load_pushed_down_blockages_start_level`			

Variable Name	Description	Type	Default
	Controls the loading of pushed down blockages at the time the cellview is opened. This environment variable specifies the level from which to start loading pushed down blockages. oaBlockages with the `isPushedDown()` attribute set to `true` will use this start level.	Integer	0
`db.load_pushed_down_blockages_stop_level`			
	Controls the loading of pushed down blockages at the time the cellview is opened. This environment variable specifies the level from which to stop loading pushed down blockages. oaBlockages with the `isPushedDown()` attribute set to `true` will use this stop level.	Integer	`MAX_INT`
`db.make_prboundary_blockage`	When set `true`, prevents the router from routing outside the prBoundary.	Boolean	`false`
`db.map_layout_to_abstract_view`	When `true` and building instances, if the viewName of the master is “layout”, will attempt to find an “abstract” cellview and use it instead.	Boolean	`false`
`db.one_shape_term_per_net`	Determines whether pre-existing power/ground nets are treated as a single terminal or as multiple shapes.	Boolean	`true`

Variable Name	Description	Type	Default
db.paths_as_routes	Converts paths to routeSegs.	Boolean	false
db.poly_layer_name	If this environment variable is set when the cellview is loaded, the named layer will automatically be recognized as the poly layer and will be a valid routing layer.	String	poly
db.prboundary_layer_name	Specifies the layer name for the prBoundary shape.	String	prBoundary
db.prboundary_purpose_name	Specifies the purpose name for the prBoundary shape.	String	boundary
db.proute_tap_to_depth	Specifies a value to insert vias on the power ground grid mesh shapes that are not at the top-level.	Integer	0
db.preserve_routing_via_order	Specifies whether the router will select vias in the order specified by the list of valid routing vias (validRoutingVias or extendedValidRoutingVias) or selects vias from the list based on calculated costs (false).	Boolean	false
db.recursive_pg_label	When true, prevents verify_connectivity from tagging a short where power cover cells overlay (abstract) block pins and one pine is not assigned to a net.	Boolean	false
db.save_original_via_master_property			

Variable Name	Description	Type	Default
	Determines whether original via properties should be saved/loaded for each remastered via. Set this variable before saving the design and before re-loading the design to permit uncloning of remastered vias.	Boolean	`false`
`db.suppress_warn_on_same_mask_num_layers`	Controls the display of a warning message when a technology library has multiple layers with the same mask number. The warning message is suppressed when the environment variable is set to `true`.	Boolean	`false`
`db.use_separate_pref_ext_dir`	Determines whether the router considers settings for `preferredExtensionDirection`, `inlineViaPreferred`, and `preferredViaOrigin` when selecting vias for routing.	Boolean	`false`

Variable Name	Description	Type	Default
db.use_valid_lpps	Determines whether the router should consider validRoutingLPPs. When the environment variable is specified, the extractor extracts only those routing shapes that are on validRoutingLPPs. However, validRoutingLayers should exist with the valid list of layers for the router to use. **Note:** You can add the environment variable to the vsrPreLoad.tcl file after adding the constraint.	Boolean	true
db.use_via_rules_from_via_def	Specifies one or more via definitions for proute_via_insertion to use instead of the default behavior of choosing the best via based on the design constraints.	Stringlist	
db.user_namespace	Sets up the system to understand the syntax used for terminal and net names. For example, the bus notation in CDB is <> but in Verilog it is []. openaccess is the same as native. Choices are: native, verilog, def, spef, spf, vhdl, openaccess, cdba	String	def
db.use_worst_cut_blockage_spacing			

Variable Name	Description	Type	Default
	Applies only to shapes whose size is larger than cutClass or minWidth of the layer (blob) and shapes with at least one dimension that is smaller than cutClass or minWidth of the layer (sliver) when the intended measurement style is center-to-center, or the cutClassProfile parameter (for minCutClassSpacing and minCutClassClearance only) is specified. When `true`, the worst case edge-to-edge spacing or clearance is used as the required spacing and there is no center-to-center measurement. If the cutClassProfile parameter is set, the profile value with the maximum required spacing or clearance is used.	Boolean	`true`

Variable Name	Description	Type	Default
	When `false`, the smaller cut center-to-cut center spacing or clearance (for the largest square cutClass) is used by projecting the Vx into the blob. There is no space checking for slivers, because they cannot be projected into a cutClass. No profile value is used for the required spacing or clearance computation. Applies to oaMinViaSpacing, oaMinViaClearance, minCutClassSpacing, and minCutClassClearance.		
`db.zone_unplaced_instances`	Prevents unplaced devices in the design from being discarded when the design is loaded. Setting to `false` can improve load time on designs with a large number of unplaced instances.	Boolean	`true`

Graphical User Interface Environment

The variables in the following table control GUI settings.

Variable Name	Description	Type	Default Value
`gui.active_window`	Gets the current active window.	Integer	not applicable

Variable Name	Description	Type	Default Value
`gui.delay_read_pixels`	Enables/disables graphics caching.	`true` `false`	`false`
`gui.disable_redraw_on_source`	If `true`, whenever the Tcl source() or eval () commands are called, the `gui.enable_redraw` environment variable is set `false`, the file is sourced, then `gui.enable_redraw` is set to `true` (enabled)	Boolean	`false`
`gui.dimension.auto_complete`	When `true`, the dimension measurement is completed after the second endpoint is set. When `false`, the extension lines and dimension readout can be positioned after the endpoints are set.	Boolean	`false`
`gui.dimension.axis_lock`	Controls the pointer movement after the first endpoint is set, when in dimension mode with shape snapping disabled. `"*"` any direction `"+"` horizontal or vertical `"|"` vertical `"_"` horizontal `"X"` or `"x"` diagonal `"/"` positive diagonal `"\"` negative diagonal	Character	"*"

Variable Name	Description	Type	Default Value
`gui.dimension.shape_snapping`	When `true` with dimension mode enabled, edges and vertices under the pointer are highlighted as you move the pointer in the workspace. If an edge is selected as the first endpoint, only orthogonal measurements can be made. If a vertex or point is selected first, then non-orthogonal measurements can be made. When `false`, the dimension point is placed at the pointer position.	Boolean	`true`
`gui.enable_redraw`	Specifies whether redraws are permitted.	Boolean	`true`
`gui.fit_next_previous_changes_view_area_only`	Controls the behavior of `fit -next` and `fit -previous`. If this variable is true, the current Layer Object Display Panel parameters are retained when scrolling through next and previous views, and the coordinates and magnification that were saved for the views are used. If this variable is false, the visibility and active state for the saved view are also restored.	Boolean	`false`

Variable Name	Description	Type	Default Value
`gui.graphics`	Selects the graphics rendering system. When you change the value, the next cellview window that you open will use the new setting; existing windows are not changed. ■ `OpenGL` draws using OpenGL ■ `QT` draws using QT. ■ `X11` (Unix only) draws using Xlib calls.	String	`OpenGL`
`gui.hide_lcp`	Hides boundaries of all the shapes with layer and color combination specified as `layername:colorName`. For example, `M1:Red`. Example: `setvar gui.hide_lcp { M1:red }` This hides boundaries of all the shapes, present on `M1` layer, having Red color outlines.	String	`""`
`gui.ignore_colors_file`	Prevents the default `colors.txt` file from loading at startup.	Boolean	`false`

Variable Name	Description	Type	Default Value
`gui.inst_scale`	Specifies the minimum instance size (in pixels) for display. Instances that are smaller than this value in width and height are not drawn.	Integer	1
`gui.max_track_size_in_pixels`	Used with `gui.zoom_out_factor` to limit the zoom factor. If the zoom factor by itself would cause a track size to be larger than the given number of pixels, then the zoom is limited to that point. The track size is the sum of the wire width and clearance in the default rule for the top occurrence for the first metal layer in the design.	Real	40
`gui.precision`	Specifies the precision of numeric values (0-4) in the status bar.	Integer	3
`gui.preview_shape_limit`	Sets the maximum number of connected shapes to extract.	Integer	`10000`

Variable Name	Description	Type	Default Value
`gui.property_inspector.visibility.<object>.<property>`	Specifies the properties to be hidden while inspecting the objects within a design in the Properties browser. Example: `setvar gui.property_inspector.visibility.Guide.toPoint false` This hides the `toPoint` property in a `Guide`.	Boolean	`true`
`gui.rulers`	Specifies whether to show horizontal and vertical rulers in the artwork.	Boolean	`false`
`gui.scale`	Specifies the minimum shape size for display. Shapes that are smaller than this value in width and height are not drawn.	Integer pixel size	5
`gui.show_shape_overlaps`	If `true`, the areas where two or more shapes overlap on the same layer are drawn brighter.	Boolean	`false`
`gui.style`	Sets the window look as `motif` or `windows`.	String	current platform
`gui.truncate_history_when_saving_view`			

Variable Name	Description	Type	Default Value
	Controls the previous and next view stack. If `true` and you are in the middle of the stack when you create a new view, the top of the stack will be truncated and replaced with the new view. If `false`, the stack will be expanded to include the traversed views and the new view.	Boolean	`true`
`gui.use_vertex_arrays`	Determines whether Space-based Router uses OpenGL® vertex arrays. Set this to `false` if you are running Space-based Router through a remote X client (for example, X-Win32 or XFree86™) and are receiving GLX™ error messages. **Note:** Setting this variable to `false` will decrease the rendering performance.	Boolean	`true`
`gui.zoom_out_factor`	Controls the zoom out for a fit. A value of 1 fits the entire selected object or region in the current artwork window. A value greater than 1 zooms out by the factor, allowing you to view the surrounding area for context.	Real	1.2

Variable Name	Description	Type	Default Value
`GUISettings.output_window_ font ...`	Sets font for the Transcript Area manually.	String	`""`
`GUISettings.status_area_font ...`	Sets font for the Status Bar manually.		

`GUISettings.userdefined_ button_font ...`	Sets font for the user-defined buttons manually.
	Here, `...` is a Qt font specification of the following form:
	`<`*`fontFamilyName`*`>`
	or
	`<`*`fontFamilyName`*`>`, `<`*`pointSize`*`>`
	or
	`<`*`fontFamilyName`*`>`, `<`*`pointSize`*`>`, `<`*`pixelSize`*`>`
	Examples:
	`envSetVal CatenaFinale.GUISettings.output_window_font ""`
	`envSetVal CatenaFinale.GUISettings.status_area_font "Lucidatypewriter"`
	`envSetVal CatenaFinale.GUISettings.userdefined_button_font "Lucidatypewriter,8.5"`		

Command Environment

Table 21-1 lists variables for command mode settings.

Table 21-1 Command Environment Variables

Variable Name	Description	Type	Default
`cmd.repeating`	Determines whether interactive modes are repeating (`repeat`) or run `once`.	String	`repeat`
`cmd.`*`mode`*`.repeating`	Overrides the global repeating setting for the given interactive mode. Available settings are `default` (which uses the global `cmd.repeating` setting), `repeat`, and `once`. The *`mode`* can be one of the following: `arrow`, `create_instance`, `create_label_int`, `create_rect`, `digitize_area`, `dimension`, `edit_wire`, `highlight`, `icut`, `imove`, `rectangle`, `select`, `text`.	String	`default`
`cmd.stop_button`	When `true`, the stop button has been clicked. Set this flag to `false` using `setvar` to clear it.	Boolean	`false`

Interactive Environment

Variables in the following table control interactive mode settings.

Variable Name	Description	Type	Default
`interactive.allow_pin_movement`	If true, allows top-level pins to be moved in move or slide mode.	Boolean	`false`
`interactive.save_new_wiring_as_special`	If true, saves wires created by the wire editor as special nets in DEF.	Boolean	`false`
`interactive.via_list_length`	Controls the maximum number of lowest cost vias per number of cuts to choose from when adding vias interactively.	Integer	10
`interactive.turbo_edit`	Determines whether turbo mode is enabled. Use this for improved graphics performance when working remotely.	Boolean	`true`

Router Environment

Variables in the following table control router mode settings.

Variable Name	Description	Type	Default
`droute.adjustvias_in_localroute`	If `true`, local route fixes specific `minSpacing` errors using the `adjust_vias` command.	Boolean	`false`

Variable Name	Description	Type	Default
`droute.align_vias`	If `true`, the detail router will use single-cut vias with aligned extensions in the direction of routing. By default, the detail router will use vias with extensions in the preferred routing direction for the respective metal layers.	Boolean	`false`
`droute.check_pins_on_wsp`	If `true`, the routability check verifies whether pins and instance pins are WSP compliant. The pins and instance pins that fail the routability check are excluded from automatic routing.	Boolean	`true`
`droute.offset_transition_vias`	If `true`, double-cut transition vias for tapering will be on-wire. By default, the via origin for double-cut transition vias is centered between the cuts.	Boolean	`false`
`droute.offset_vias`	If `true`, the detail router will use on-wire vias whenever possible.	Boolean	`false`
`droute.pattern_route`	Determines how the detail router treats bus routes. 0=No bus routing 1=Bus route, then detail route 2=Bus route only	Integer	`0`

Variable Name	Description	Type	Default
`droute.shields_in_violation`	If `true`, shields will be added even when they cause design rule violations. By default, shields that cause DRC violations are not added.	Boolean	`false`
`droute.snap_to_pin_center`	If `true`, the `p2p_route` and `detail_route` commands snap wires and vias to the center of pin shapes, including simple rectangle pin shapes on multiple layers.	Boolean	`false`
`droute.vias_must_be_fully_enclosed`	If `true`, the router only adds vias at pins if the via can be fully enclosed in the pin shape.	Boolean	`false`

`droute.vias_must_be_fully_enclosed_on_layer`	Specifies the layers on which the router only adds vias at pins if the via can be fully enclosed in the pin shape. Applies only when `droute.vias_must_be_fully_enclosed` is `true`.	String	`"{ }"`
			for all layers

Power Router Environment

Variables in the following table control power router mode settings.

Variable Name	Description	Type	Default
`proute.ignore_pin_blockage`	Determines whether proute_via_insertion can legally insert a via from a power pin down to another power pin that is embedded in metal blockage.	Boolean	`false`
`proute.max_via_def`	When `true` and `db.use_via_rules_from_via_def` is set, via checking is done after via insertion is complete. Otherwise, vias can be checked both before and after the insertion process.	Boolean	`false`
`proute.undoable`	Determines whether power router (`proute`*) commands can be undone. When `true`, consecutive power router commands can be individually reversed using the `undo` command, *Edit—Undo*, or the Undo toolbar icon. This variable can be overridden for a single power router command which includes the -`undoable` argument.	Boolean	`false`

Verify Environment

Variables in the following table control verify mode settings.

Variable Name	Description	Type	Default
`checking.disable_check_trim_in_check_space`	When set to `true`, changes the default value of the `-check_trim` argument of the `check_space` command to `false`.	Boolean	`false`
`checking.check_trim_spacing_oneshape_insideprboundary`	When set to `true`, trims that are completely inside the prBoundary are checked against all other trims both inside and outside the prBoundary. All trims that are completely outside the prBoundary are not checked against all trims outside the prBoundary.	Boolean	`false`

Setting an Environment Variable

Variables can be set from the Command line or within a Tcl script. The environment variables are persistent and can be changed by a user or by Space-based Router during runtime. Many of these variables are set by built-in Space-based Router commands and are journaled so that they can be replayed.

To set an environment variable,

1. Type the following on the Command line:

   ```
   setvar variable_name value
   ```

   For example, the following command sets the minimum shape size for display to 2.

   ```
   setvar gui.scale 2
   ```

Getting the Value of an Environment Variable

To get the current value of an environment variable,

1. Type the following on the Command line:

```
getvar variable_name
```

22

Using Tcl

Tcl is used to run Virtuoso® Routing Integrated Development Environment (Routing IDE) and Cadence® Space-based Router and Chip Optimizer. All commands performed by the tool, from either the GUI or from command line input, are logged as Tcl commands in the Transcript window. The logged commands can be replayed to duplicate the session. Tcl Core commands are supported and you can also write Tcl scripts for use with Virtuoso Routing IDE or Space-based Router and Chip Optimizer.

This chapter describes the following:

Loading and Running Tcl Scripts

Important

To run a Tcl script, you must first load it.

Loading Tcl Scripts

➤ To load a Tcl script, you must source it from the command window.

```
source script_file_name
```

Examples

The following example loads a Tcl script from the directory where you started the tool.

```
source myscript.Tcl
```

The following example loads a Tcl script from your home directory.

```
source ~/myscript.Tcl
```

The following example loads a Tcl script from your PC’s F: drive.

```
source f:/projectdir/myscript.Tcl
```

Running Tcl Scripts

You can run Tcl scripts from the command line or from a bindkey.

➤ To run a Tcl script from the command line, type the script name followed by required arguments.

```
script_file_name [arguments]
```

Example

```
search_design NAND2 50
```

Examples

A convenient methodology for creating Tcl scripts is to copy the desired set of commands from the transcript into a file and modify the file as needed, adding arguments and a procedural interface.

The following examples use a combination of Tcl Core commands and Space-based Router and Chip Optimizer commands.

Example 1 - Finding and Highlighting a Net Using a Single Command

```
add_highlight -color "orange" -set [find_net -name A*]
```

This example passes the output of the `find_net` to the `-set` parameter of the `add_highlight` command.

Example 2 - Finding and Highlighting an Instance Using a Stored Variable

```
set inst_I7 [find_instance -name I7]
add_highlight -color "red" -set $inst_I7
```

This example stores the result of the `find_instance` command in a variable `inst_I7` and then passes the variable contents as the value of the `-set` parameter in the `add_highlight` command.

Example 3 - Finding and Highlighting an Instance (Master) Using a Stored Variable

```
set master_IOPAD_1 [find_instance_of -lib myproject -cell IOPAD_1 -view abstract]
add_highlight -color "blue" -set $master_IOPAD_1
```

This example find instances whose masters match the expression. The results are stored and are then passed to the `add_highlight` command.

Example 4 - Creating a Procedure to Manipulate Display Levels

```
# This sets the start and the stop display levels for the active window.
# By default it repaints the window after being set.
proc changeDisplayLevel {p_start p_stop {p_repaint 1}} {
# Get the active window
    set active_window [getvar gui.active_window]
    # Build up the string for the active window
    set active_window_string [format "%s%d" "window" $active_window]
    # Build up the string to set the start and end display levels
    set end_string [format "gui.%s.end_level" $active_window_string]
    set start_string [format "gui.%s.start_level" $active_window_string]
    # Set the start and end display levels
    setvar $start_string $p_start
    setvar $end_string $p_stop
    # Repaint if desired
    if {$p_repaint == 1} {
        refresh -window_id $active_window
    }
}
```

This procedure sets the start and stop display levels for the active window. It can be bound to a bindkey or invoked from the command line after loading. The third argument is optional and specifies whether a screen refresh should be performed. The following example sets the start and end display levels to 1 and 5, respectively, and does not refresh the screen.

```
changeDisplayLevel 1 5
```

Example 5 - Finding and Highlighting Instances Matching a Pattern Name Using a Stored Variable

```
set inst_rp [find_instance -name rp*]
add_highlight -color "green" -set $inst_rp
```

This example looks for all instances with names starting with `rp`. The found set is saved in the variable `inst_rp` and then is highlighted in green.

Example 6 - Finding and Highlighting a Subset of a Set Using Stored Variables

```
set inst_rp_regB [find_instance -set $inst_rp -name rp/rebB*]
add_highlight -color "orange" -set $inst_rp_regB
```

This example uses the set `inst_rp`, found in the previous example, and searches within that set for instances that match the given expression. The found set is saved in the variable `inst_rp_regB` and then is highlighted in orange.

23

Inspecting Properties

This chapter describes how to use Tcl commands to get, set, and add object properties, and provides Tcl procedure examples that use these commands.

The following sections are included:

Using Properties Tcl Commands

The Space-based Router and Chip Optimizer properties Tcl commands are described in detail in Chapter 3, “Edit Commands.”

The following table provides a quick reference for the commands.

Table 23-1 Properties Commands

Command	Function
hp	Specifies whether a property exists for an object.
inspect_config	Customizes the information displayed in the Properties Browser for a specific type of object.
inspect_getprop	Gets a property for an object.
inspect_prop	Another method for getting a property for an object. Shorthand name is `ip`.
inspect_setprop	Sets a property for an object.
make_list	Constructs a Tcl list of object identifiers from the given set of objects. Shorthand name is `ml`.

Get a Property Value for an Object

Two commands, `inspect_getprop` and `ip` (shorthand for `inspect_prop`) can be used to get the property value for an object.

```
inspect_getprop -prop_name s_propName -object d_ctuObj
```

and

```
ip s_propName d_ctubobj
```

are equivalent in function.

For example, both of the following commands return the name of the net given by the `$net` variable:

- `inspect_getprop -prop_name name -object $net`
- `ip name $net`

Get a Property Value for Items in a Set

To get a property value for an item in a set, use one of the following:

- `inspect_getprop -prop_name` *`s_propName`* `-item` *`i_itemNbr`* `-set` *`d_setObj`*
- `ip` *`s_propName`* `[lindex [ml` *`d_setObj`*`] [`*`i_itemNbr`*`-1]]`

The following examples return the `type` property value for the first object in the set given by the `iSet` variable.

- `inspect_getprop -prop_name type -item 1 -set $iSet`
- `ip type [lindex [ml $iSet] 0]`

By default, `inspect_getprop` operates on the selected set, if not empty, and `-item` must be given if there is more than one object in the set.

Tcl Examples for Inspecting Properties

The procedures in this section are included in the `examples/tcl/inspectProcs.tcl` file in your installation directory.

To use these procedures, do the following:

1. On the Space-based Router and Chip Optimizer command line, type:

   ```
   source installPath/tools/dfII/samples/rde/examples/tcl/inspectProcs.tcl
   ```

   where *`installPath`* is the path to your installation directory.

Procedure or Tcl command	Description
`findItemsInSet`	Find Items in a Set
`findElementsOfItems`	Find Elements of Items in a Set
`createSetOfNetElementsType`	Create a Set of Net Elements of a Specific Type
`createListOfNetElementsType`	Create a List of Net Elements of a Specific Type
`findViasInNets`	Finding All Vias on Nets in a Set
`findViaMaster`	Finding All Vias with a Given Master on Nets in a Set
`getNetName`	Get the Names of Nets in a Set
`getGuideNetNames`	Get the Names of Nets with Guides
`selectNetsFromRouteSegments`	Select Nets from Route Segments
`getNetsFromRouteSegments`	Create a Set of Nets Containing Route Segments from Another Set
`selectRoutesFromNets`	Select Routes from Nets
`getRoutesFromNets`	Create a Set of Routes Containing Nets from Another Set
`getGuideNetSet`	Create a Set Containing Nets with Guides
`findNetConnections`	Create a Set Containing Instances Attached to Nets in a Set
`findInstancesConnectedToNets`	Find Instance Connections for Nets Matching Net Name Expression
`set2ann`	Create Annotations for Items in a Set

Find Items in a Set

This procedure prints the `type` property for all elements in the given set.

```
proc findItemsInSet {inputSet} {

  #process each item in the set. for loop
  foreach inputObject [ml $inputSet] {
    puts [ip type $inputObject]
  }
  puts "[set_count -set $inputSet] items in set."
}
```

The following example uses the `findItemsInSet` procedure and shows the output.

```
findItemsInSet [get_selection_set]
```

```
ctuRouteSegment
ctuRouteVia
ctuNet
ctuRoute
4 items in set.
```

Find Elements of Items in a Set

For each item in a set, this procedure prints the `type` property for the item and the `type` property for each of its elements, if it has elements.

```
proc findElementsOfItemsInSet { inputSet } {

  set i 1
  # inspect each item in the set.
  foreach item [ml $inputSet] {
    if {[hp elements $item]} {
      foreach elem [ml [ip elements $item]] {
        puts "Item $i [ip type $item] element: [ip type $elem]"
      }
    } else { puts "Item $i has no elements." }
  incr i
  }
}
```

The following is example output for a set containing a net and a route segment:

```
findElementsOfItemsInSet [get_selection_set]
```

```
Item 1 ctuNet element: ctuInstTerm
Item 1 ctuNet element: ctuInstTerm
Item 1 ctuNet element: ctuRoute
Item 1 ctuNet element: ctuSteiner
Item 1 ctuNet has 4 elements.
Item 2 ctuRouteSegement has 0 elements.
```

Create a Set of Net Elements of a Specific Type

For each net in the input set, puts all elements of a given type into an output set. Non-net items in the set are ignored.

```
proc createSetOfNetElementsType { inputSet typeVal } {

  set outputSet [create_set]

  foreach item [ml $inputSet] {
    if {[string compare [ip type $item] "ctuNet"] == 0} {

      #process only Nets here
      foreach elem [ml [ip elements $item]] {
        if {[string compare [ip type $elem] $typeVal] == 0} {
          add_object_to_set -object $elem -set $outputSet
        }
      }
   }
  }
  return $outputSet
}
```

The following example creates a set containing all the routes from the nets in the selected set.

```
set routeSet [createSetOfNetElementsType [get_selection_set] ctuRoute ]
```

Create a List of Net Elements of a Specific Type

For each net in the input set, puts all elements of a given type into an output list. This procedure is identical in function to `createSetOfNetElementsType` except this returns a list, instead of a set.

```
proc createListOfNetElementsType { inputSet typeVal } {

  set outputList [list]

  foreach item [ml $inputSet] {
    if {[string compare [ip type $item] "ctuNet"] == 0} {

       #process only Nets here
       foreach elem [ml [ip elements $item]] {
         if {[string compare [ip type $elem] $typeVal] == 0} {
            lappend outputList $elem
         }
       }
    }
  }
  return $outputList
}
```

Finding All Vias on Nets in a Set

Returns a set containing all vias on nets of the given set. First finds all the routes for nets in the set, then puts all the vias for those routes in the output set.

```
proc findViasInNets { inputSet } {

  set outputSet [create_set]
   foreach route [createListOfNetElementsType $inputSet ctuRoute ] {
     foreach elem [ml [ip elements $route ] ] {
       if {[string compare [ip type $elem] "ctuRouteVia"] == 0} {
          #found a via
#         puts "Net [ip net.name $elem], Via Master [ip master.name $elem]"
          add_object_to_set -object $elem -set $outputSet
       }

     }
  }
   return $outputSet
}
```

The following example creates a set containing all vias on nets in the HL1 highlight set.

```
set vset [findViasInNets [get_highlight -name HL1]]
```

Finding All Vias with a Given Master on Nets in a Set

Returns a set containing vias with the given master via that are contained in nets in the given set.

```
proc findViaMaster { inputSet masterVal } {

  set viaSet [create_set]
  set i 0
  foreach via [ml [findViasInNets $inputSet]] {
#   puts "Master $masterVal; Net [ip net.name $via], Via Master \
        [ip master.name $via]"
    if {[string compare [string trim [ip master.name $via] "\""] $masterVal] \
        == 0} {
      add_object_to_set -object $via -set $viaSet
      incr i
    }
  }
  puts "Found $i vias with master $masterVal."
  return $viaSet
}
```

The following example creates a set containing all M2_M1 vias on nets in the selected set.

```
set vset [findViaMaster [get_selection_set] M2_M1]
```

Get the Names of Nets in a Set

This procedure returns the list of names for nets in the given set.

```
proc getNetName { netSet } {
   set netNamesList [list]
   foreach net [ml $netSet] {
      set netType [string trim [ip type $net] "'"]
      if {[string compare $netType "ctuNet"] == 0} {
         set netName [string trim [ip name $net] "'"]
        lappend netNamesList $netName
      }
   }
   return $netNamesList
}
```

For example,

```
getNetName [get_highlight -name HL2]
```

```
net1 {in[0]} net2
```

Get the Names of Nets with Guides

This procedure returns a list of names of nets that contain guides (opens).

```
proc getGuideNetNames {} {
   set netNamesList [list]
   set guideSet [find_shape -shape_types guide -ignore_case true \
      -no_wildcard false -silent]
   foreach guide [ml $guideSet] {
      set guideType [string trim [ip type $guide ] "'"]
      if {[string compare $guideType "ctuGuide"] == 0} {
         set netName [string trim [ip net.name $guide ] "'"]
         lappend netNamesList $netName
      }
   }
   return $netNamesList
}
```

Create a Set Containing Nets with Guides

This procedure returns a set containing nets with guides.

```
proc getGuideNetSet {} {
   set netSet [create_set]
   set guideSet [find_shape -shape_types guide -ignore_case true \
      -no_wildcard false -silent]
   foreach guide [ml $guideSet] {
      set guideType [string trim [ip type $guide ] "'"]
      if {[string compare $guideType "ctuGuide"] == 0} {
         set net [ip net $guide ]
         add_object_to_set -set $netSet -object $net
      }
   }
   return $netSet
}
```

For example,

```
set guideSet [getGuideNetSet]
sel:d128040
getNetName $guideSet
net13 net25 net17
```

Select Nets from Route Segments

This procedure replaces the selected set with nets containing the route segments in the input set.

```
proc selectNetsFromRouteSegments { routeSegmentSet } {
  set netSet [create_set]
  foreach route [ml $routeSegmentSet] {
    set routeType [string trim [ip type $route ] "'"]
    if {[string compare $routeType "ctuRouteSegment"] == 0} {
      set net [ip net $route ]
      add_object_to_set -object $net -set $netSet
    }
  }
  replace_set -set1 $netSet -set2 [get_selection_set]
}
```

Create a Set of Nets Containing Route Segments from Another Set

This procedure returns a set of nets from the route segments in the input set.

```
proc getNetsFromRouteSegments { routeSegmentSet } {
  set netSet [create_set]
  foreach route [ml $routeSegmentSet] {
    set routeType [string trim [ip type $route ] "'"]
    if {[string compare $routeType "ctuRouteSegment"] == 0} {
      set net [ip net $route ]
      add_object_to_set -object $net -set $netSet
    }
  }
  return $netSet
}
```

Select Routes from Nets

This procedure replaces the selected set with routes from the input set of nets.

```
proc selectRoutesFromNets { netSet } {
  set routeSet [create_set]
  foreach net [ml $netSet] {
    set netType [string trim [ip type $net] "'"]
    if {[string compare $netType "ctuNet"] == 0} {
      foreach element [ml [ip elements $net]] {
        set elementType [string trim [ip type $element] "'"]
        if {[string compare $elementType "ctuRoute"] == 0} {
          add_object_to_set -object $element -set $routeSet
        }
      }
    }
  }
  replace_set -set1 $routeSet -set2 [get_selection_set]
}
```

Create a Set of Routes Containing Nets from Another Set

This procedure returns a set of routes from the nets in the input set.

```
proc getRoutesFromNets { netSet } {
  set routeSet [create_set]
  foreach net [ml $netSet] {
    set netType [string trim [ip type $net] "'"]
    if {[string compare $netType "ctuNet"] == 0} {
      foreach element [ml [ip elements $net]] {
        set elementType [string trim [ip type $element] "'"]
        if {[string compare $elementType "ctuRoute"] == 0} {
          add_object_to_set -object $element -set $routeSet
        }
      }
    }
  }
  return $routeSet
}
```

Create a Set Containing Instances Attached to Nets in a Set

This procedure returns a set of instances that are attached to nets in the given set.

```
proc findInstancesConnectedToNets {inputSet} {
  set instSet [create_set]
  foreach inputObject [ml $inputSet] {
    set inputObjectType [ip type $inputObject]
      if {[string compare $inputObjectType "ctuNet"] == 0} {
        foreach element [ml [ip elements $inputObject]] {
          set elementType [ip type $element]
          if {[string compare $elementType "ctuInstTerm"] == 0} {
            set elementInst [ip instance $element]
            add_object_to_set -object $elementInst -set $instSet
        }
      }
    } else { puts "$inputObjectType: this is not a net" }
  }
  puts "[set_count -set $instSet] instances are attached to selected nets."
  return $instSet
}
```

The following example uses the `findInstancesConnectedToNets` procedure and adds the instances that are found to the current highlight set.

```
replace_set -set1 [findInstancesConnectedToNets [get_selection_set]] -set2
[get_current_highlight]
```

Find Instance Connections for Nets Matching Net Name Expression

This procedure finds all nets matching the input net name expression, then prints a description for each instTerm connection on each net and the total number of instTerm connections for each net.

```
proc findNetConnections {netName} {
  set netsSet [ml [find_net -name $netName -silent true ] ]
  foreach net $netsSet {
    set netName [ip name $net]
    puts " "
    puts "Net $netName:"
    set netElements [ml [ip elements $net]]
    set instTermCount 0
    foreach netElement $netElements {
      set netElementType [ip type $netElement]
      if {[string compare $netElementType "ctuInstTerm"] == 0} {
        set termName [ip name $netElement]
        #set instName [ip name [ip instance $netElement]]
        set instName [ip instance.name $netElement]
        #set instTermPins [ml [ip elements [ip term $netElement]]]
        set instTermPins [ml [ip term.elements $netElement]]
        set connectionType [ip termType [ip term $netElement]]
        foreach instTermPin $instTermPins {
          set instTermPinShapes [ml [ip elements $instTermPin]]
          foreach pinShape $instTermPinShapes {
            set pinShapeBounds [ip bounds $pinShape ]
            #set pinShapeLayer [ip name [ip layer $pinShape ] ]
            set pinShapeLayer [ip layer.name $pinShape ]
            puts "  Pin shape on layer $pinShapeLayer for $termName \
                ($connectionType) on net $netName is located at $pinShapeBounds"
          }
        }
        incr instTermCount
      }
    }
    puts "  Net $netName has $instTermCount instTerm connections."
  }
}
```

Create Annotations for Items in a Set

This procedure is useful when you want a quick way to zoom to each item in the set. After invoking this procedure, you can use the Annotation Browser as the navigation mechanism.

Note: Only objects with a `bounds` property can be annotated using this method.

```
proc set2ann { myset } {

  set i 1
  foreach objInSet [ml $myset] {
    set objHasBounds [hp bounds $objInSet]
    if ($objHasBounds) {
      set objBounds [inspect_getprop -prop_name bounds -object $objInSet]
      puts "\# Msg: Creating rectangle annotation at $objBounds\n"
      add_rectangle -color Cyan -lineWidth 1 -rect [bounds2Bbox $objBounds]
```

```
     } else {
        puts "\# Msg: Error couldn’t find bounds prop on item $i. Skipping.\n"
     }
     incr i
  }
}
```

The following procedure is the equivalent to `set2ann` but it uses `ip` (`inspect_prop`) instead of `inspect_getprop` and does not create some intermediate variables.

```
proc set2annip { myset } {
  set i 1
  foreach objInSet [ml $myset] {
    if {[hp bounds $objInSet]} {
      add_rectangle -color Cyan -lineWidth 1 \
          -rect [bounds2Bbox [ip bounds $objInSet]]
    } else {
      puts "\# Msg: Error - couldn’t find bounds prop on item $i. Skipping.\n"
    }
  incr i
  }
}
```

Annotate All Vias with a Given Master on Nets in a Set

This example annotates all `M2_M1` vias on nets in the selected set.

```
set2ann [findViaMaster [get_selection_set] M2_M1]
```