`

VC SpyGlass Lint®
User Guide

Version S-2021.09-SP2, March 2022

Copyright Notice and Proprietary Information

Destination Control Statement

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

Disclaimer

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

Trademarks

Free and Open-Source Software Licensing Notices

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

Contents

Introduction

This chapter provides an overview of VC SpyGlass Lint and includes the following sections:

- *About this Guide*
- *Contents of this Manual*

About this Guide

The VC SpyGlass Lint User Guide describes the concepts, features, usage, and tags of VC SpyGlass Lint, which enable you to use the Verilog or SystemVerilog designs against various coding standards and design tags.

Contents of this Manual

The VC SpyGlass Lint User Guide consists of the following sections:

Section	Description
Getting Started	Provides an overview of VC SpyGlass Lint.
Reading the Design	Describes how to read a design in VC SpyGlass Lint.
Working with Methodologies and Goals	Provides information on using methodologies and goals.
Using VC SpyGlass Lint	Provides reference information for the built-in Tcl commands implemented in VC SpyGlass Lint.
Using Tags in VC SpyGlass Lint	Provides detailed procedures for how to configure and use the prepackaged tags to check your HDL code.
VC SpyGlass Functional Lint	Describes the VC SpyGlass Functional Lint flow.
Analyzing VC SpyGlass Lint Results	Describes the VC SpyGlass Lint violations, how to debug Lint Violations Using Tcl, and reports generated by VC SpyGlass Lint.
Appendix A - Supported Commands	Describes the SDC commands, Tcl commands, configure commands and application variables supported by VC SpyGlass Lint.
Migrating Waivers	Describes migrating waivers in VC SpyGlass Lint.

Getting Started

This section introduces Verification Compiler (VC) Platform, VC SpyGlass, and VC SpyGlass Lint and is organized into the following sections:

- Verification Compiler (VC) Platform
- VC Static and Formal Solution
- VC SpyGlass Lint

Verification Compiler (VC) Platform

Today's electronic consumer market is driven by a huge demand for mobility, portability, and reliability. Additional functionality, performance, and bandwidth are very important for maximizing semiconductor sales in addition to faster time-to-market and product quality. The evolution of applications, such as cellular phones, laptops, PDAs, computers, mobile multimedia devices, and portable systems, has seen an exponential growth in battery operated systems.

The increase in design complexities and shrinking technologies, where more and more functionality is being added into smaller area of a chip has brought in a new set of challenges in System-on-Chip (SoC) verification. With adoption of advanced techniques and sophisticated tools, which helps in verifying SoC connectivity, signal integrity, power management, and functionality of analog components, hardware-software co-verification has become inevitable.

This brings in a need for a unified and integrated verification environment with seamless flow and reuse of the information across different domains/levels to achieve faster results.

Verification Compiler Platform is a next-generation verification solution that provides a scalable environment, where sophisticated tools work seamlessly with each other throughout the flow to accomplish various verification tasks using integration of technologies. It helps in optimizing design iterations and recompilations, shortens debug cycles, and enables steady integration and interoperability between individual verification tools.

VC Static and Formal Solution

Traditionally, simulation-based dynamic verification techniques have been the mainstay of functional verification. As modern day SoC designs become more complex, the adoption of static verification techniques is important.

Synopsys' VC Static and Formal verification solution offers the next-generation comprehensive VC Formal verification solution, VC SpyGlass Lint, VC SpyGlass CDC, VC SpyGlass RDC, and VC Low Power verification solution.

Synopsys' VC Static and Formal verification solution combines the best-in-class technologies for improved ease-of-use, accuracy, and performance. It also provides with low violation noise and excellent debug capabilities. This solution enables designers and verification engineers to quickly and easily find and fix bugs in RTL before simulation; therefore, reducing the time needed before software bring-up, hardware emulation, and prototyping.

VC LP is a multi-voltage, static low power rule checker that allows engineers to rapidly verify designs that use voltage control-based techniques for power management. VC LP is part of the Synospys Eclypse Flow. VC LP also helps in pipe-cleaning the power intent of the design that is captured in IEEE 1801 Unified Power Format (UPF) before such intent is used as a golden reference for implementation and other verification tools. Further, VC LP verifies the implemented power-intent later in the design flow.

VC Formal verification offers property checking that consists of mathematical techniques to test properties or assertions to ensure the correct functionality of RTL designs. For more information, see the VC Formal Verification User Guide.

VC SpyGlass Lint, a static verification tool, performs system-to-netlist verification using prepackaged tags to check Verilog, SystemVerilog, VHDL designs against various coding standards and design tags. After you elaborate your design in the VC Lint environment, you can use built-in Tcl queries, prepackaged checks, and a set of predefined procedures to run interactive queries on your design.

RTL code is verified for connectivity correctness between two nodes of a design using the VC Formal Connectivity Checking solution. For more information, see the VC Formal Connectivity Checking User Guide.

RTL is further verified for functionality and policy compliance. Model checking technique exhaustively and automatically checks whether a model adheres to a given specification and verifies correct properties of finite-state systems. For more information, see the VC Formal Verification User Guide.

VC SpyGlass RDC performs reset verification to report issues, such as metastability, glitches, and functional failures leading to silicon failure. It also provides advanced RDC capabilities, such as performing RDC synchronization in sequential crossing paths, memory modeling, and extracting reset order automatically from the simulation database. In addition, it generates RDC reports that you can use to identify synchronization issues in the design. You can also waive and filter violations.

VC SpyGlass Lint

VC SpyGlass Lint is a system-to-netlist checker tool that comes with prepackaged tags to check Verilog or SystemVerilog designs against various coding standards and design tags.

After you elaborate your design in the VC SpyGlass Lint environment, you can use the built-in Tcl shell and a set of predefined procedures to run interactive queries on your design. With full support for Verilog 2001 and IEEE Std 1800-2005 SystemVerilog design constructs, combined with extensive RTL and netlist checks, VC SpyGlass Lint can check your designs for errors that may cause problems in the downstream simulation, synthesis, and equivalence checking flows.The best way to learn the tool is to test one of your designs with the Checker and the prepackaged policies (sets of tags). The prepackaged tags are designed to meet the best coding guidelines and practices followed in the industry.

VC SpyGlass Lint is a design checker tool that comes with a rich set of prepackaged tags to check Verilog and SystemVerilog designs against various coding standards and design tags.

You use the Checker to designate Verilog or SystemVerilog input files that you want to compare against coding tags that you select with the click of a mouse. The Checker analyzes your Verilog and SystemVerilog source code against the selected tags and generates a report indicating which lines in the code violate the tags. You can then:

- Open the report in the GUI and debug the reported violations
- Generate schematics to debug structural checks.

- Navigate directly from the violation message in the GUI to its source code in your HDL files.
- Create filters and waivers for better report management

Licensing Requirements

VC SpyGlass Lint requires the 'VC-LINT-BASE & checker' licenses. Ensure that these licenses are available before you run VC SpyGlass Lint.

Key Features of VC SpyGlass Lint

The key features and benefits of using VC SpyGlass LINT for static verification in a typical design are as follows:

- Improved total runtime and PEAK memory
- Unified compilation
- Management of large memories and does not require options to handle large memories during synthesis
- Word-level traversal and avoidance of bit level traversal (if needed)
- Consistent handling of clocks and resets
- Multi-message and multi-line support
- Tag-specific reports
- Noise reduction using the allviol parameter
- V2k language support
- VHDL2008 support
- Complete IEEE 1800-2005 System Verilog design constructs
- Saving and restoring elaborated design view

VC SpyGlass Lint Methodology Flow

Figure 1 shows the VC SpyGlass Lint methodology flow.

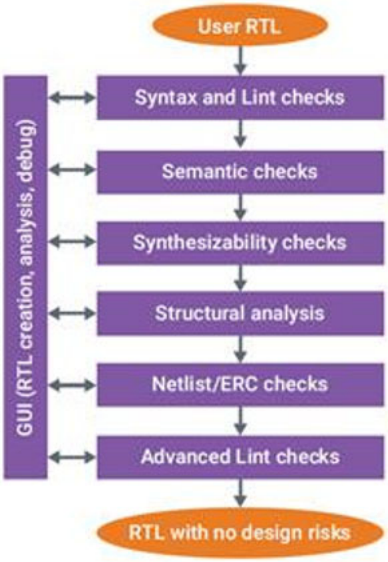


FIGURE 1. VC SpyGlass Lint Methodology Flow

Reading the Design

This section describes how you can get started with VC SpyGlass Lint.

Setting Up Design Environment

You must provide the synthesizable design files (RTL/Netlist) to perform Lint verification.

Licensing and Installation

This release of VC Static Platform is a standalone platform and must be installed in an empty directory, using the latest version of the Synopsys Installer. Do not install this release over an existing release of a Synopsys tool.

For installation instructions, see the `vc_static_INSTALL_README.txt` file in the product download directory. For detailed installation instructions, see the Synopsys Installation Guide at the following address:

`http://www.synopsys.com/install`

Before running Synopsys tools, you must have installed and configured the Synopsys Common Licensing (SCL) software, retrieved your license key file, and defined the license file environment variable. For detailed information about SCL installation and setup, see the Synopsys Licensing Quickstart Guide at the following address:

`http://www.synopsys.com/licensing`

For more information on the VC Static license keys, see section *VC Static Product Installation Notes*.

VC SpyGlass Fundamentals for Lint

VC SpyGlass Lint uses the synthesizable RTL files to perform Lint verification. The tool provides you with commands that you can use to create and verify Lint setup, perform various checks on the design.

VC SpyGlass Lint reports the results of the checks performed on the design in the form of violation tags. A violation tag can be an Error, Warning, or Info type depending on the type of violation reported.

Support for DesignWare (DW) Components

VC Static supports usage of DesignWare components in the RTL. To compile DW components, use the `dw_analyze` command as shown below:

```
dw_analyze -dwroot <DC-install-path> <dir-name>
```

Where:

- `<DC-install-path>` specifies the path to Design Compiler
- `<dir-name>` specifies the directory where the compiled DW is stored

For example, consider the following command:

```
dw_analyze -dwroot /global/apps/syn_2016.12-SP3
NG_DW_WORK_1712
```

In the above command, `/global/apps/syn_2016.12-SP3` specifies the path to DC and `NG_DW_WORK_1712` is the name of the directory where the compiled DW is generated.

Reusing the Pre-compiled DW Components

You can use DW components that are compiled in a previous major release by saving the compiled DW components to an appropriate location that is accessible by other users. Users can use this compile in all Service Pack releases of the major VC Static release. You need to recompile the DW components when you move to the next major VC Static release.

For example, if you compile DW components in the VC Static 2017.12 release and save it at a central location, all users can use the compile in all the VC Static 2017.12-SP* releases.

To use such compiled DW components, use the following command:

```
set_app_var vsi_dwroot  <path>/NG_DW_WORK_1712
```

Where, `<path>` specifies the directory where the compiled DW components are stored.

NOTE: *You can save the compiled DW components on your local machine and use it in successive SP* releases as well.*

Selecting DW Components for Elaboration

If both RTL and DW pre compiled libraries are present in a design, VC SPYGLASS, by default, uses the RTL definition for elaboration. This might lead to creation of blackboxes of the DesignWare components in the design.

Set the prefer_dw_over_rtl application variable to `true` to enable VC SpyGlass to use the DW pre-compiled libraries instead of RTL.

Language Support

VC Static platform supports the following industry standard HDLs:

- Verilog (1995, v2k)
- VHDL (‘87, ‘93, 2008)
- SystemVerilog 1800-2005, and 2009
- Mixed Language Design

Running the VC Static Shell

VC SpyGlass Lint uses the pivotal environment variable, `VC_STATIC_HOME`. This variable must be set to point to the installation directory as shown in the following code snippet. In the installation directory, you can find the bin, lib, doc and other directories.

```
% setenv VC_STATIC_HOME /tools/synopsys/vcst
```

Optionally, you can add $VC_STATIC_HOME/bin to your $PATH. To start the VC Static tool, execute the following command:

```
% $VC_STATIC_HOME/bin/vc_static_shell
```

To invoke the VC Static platform from the shell in the 64-bit mode (default mode), use the following command:

```
%vc_static_shell
```

VC Static Shell Command Line Options

The following command line options are available for VC Static. The options might be abbreviated by leaving out the text in parenthesis; for example, either `-f` or `-file` can be used to give the name of a script file to execute.

Syntax

```
%vc_static_shell -help
Usage: /~/Release/bin/vc_static_shell
[-batch]Start tool in batch mode (non-interactive).
[-cmd_log_file <log_name>]Name of command log file in current
directory.
[-f(ile) <file_name>]Script file to exec after setup.
[-gui]Start the GUI ActivityView.
[-h(elp)]Print this help message.
[-id | -ID] Give more information about application build/
env.
[-lic_wait <minutes>]Wait for license for #minutes.
[-no_init]Don't load .synopsys_vcst.setup files.
[-no_restore]Remove previous session and start a new one.
[-no_ui]Starts the tool without the GUI/UI process.
[-output_log_file <log_name>]Capture console output in given
log file.
[-read_only]Restore a previous session in read-only mode.
[-restore]Restore a previous session.
[-session <session_name>]Use the <session_name> directory
```

```
for the runtime database.
debug options:
[-echo]Echo the environment but do not invoke the executable.
```

Use Model

Using vc_static_shell in batch mode

```
%vc_static_shell -f vcst.tcl -batch
```

Using vc_static_shell in interactive mode

```
%vc_static_shell -f vcst.tcl
```

NOTE: *When you use the -batch option, VC Static automatically quits the shell even when quit is not explicitly specified in the vcst.tcl file or when an unexpected error occurs and the full run is not complete. This is useful for regression runs.*

Changing the VC Static Session Name and Location

After `vc_static_shell` is run in any user work directory, VC Static creates a default session (work database directory) in the current working area called *vcst_rtdb* [VC Static Run Time Data Base] along with default log files. The default session name is *vcst*.

You can change the name of the session and the location of the session at the time of invoking `vc_static_shell`.

```
%vc_static_shell -session my_path/my_session <other
commands>
```

This command creates a database directory named *my_session_rtdb* in the *./my_path* directory.

Sample Design Setup

The following script file is a sample script that you can use to run VC SpyGlass Lint flow in `vc_static_shell`.

```
### Note to set respective app_vars' and other settings in below
order.

# Basic VC SpyGlass LINT TCL file.
# Edit, fill in <options> and uncomment any settings/commands,
then save.

#Settings to enable VC SpyGlass LINT flowset_app_var
enable_lint true


#Following command reads Cell library files in .db format set
search_path < Path to directory which contain db files> set
link_library <db file list which located in search path> Refer
to the Reading the Liberty Files section for details.


### Enabling/Disabling VC Spyglass LINT tags
configure_lint_tag -enable -tag "<tag_name>" -goal <goal_name>

configure_lint_tag_parameter -tag "<tag_name>" -parameter
<parameter_name>

-value {<value>} -goal <goal_name>

###
configure_lint_setup -goal <goal_name>


### Set below settings after the "Enabling/Disabling VC
Spyglass LINT tags" settings
#### Design read

define_design_lib WORK -path ./WORK/VCS
      analyze -format <Verilog/sverilog/vhdl> {-f <file_list> }
-vcs {<vcs_command>}
elaborate <top_module_name>

#Command to check LINT
check_lint

#Following command generates verbose report
report_lint -verbose -file <File_name> -limit 0

# Show results in GUI
```

```
view_activity
```

Saving and Restoring Sessions Using save_session and restore_session

VC SpyGlass Lint enables you to save the results of the run after the *check_lint* TCL command and restore the results in the subsequent run from the same point. You can use GUI based debugging, applying waivers and report generation in the restored run.

When you exit the vc_static_shell, the results of the current session are not automatically saved. To save the session setup and run data, use the save_session command. The command uses the following syntax:

```
save_session
-session <session_name>
```

where, -session <session_name> specifies the name to be used to save the session.

To restore a saved session, use the restore_session command. The command uses the following syntax:

```
restore_session

[-session <session_name>]
```

where, -session <session_name> specifies the name of a previously saved session.

Updating Application Variable Settings

VC SpyGlass Lint offers a list of application variables that can be used as per your requirements. To see the list of all the available application variables and their current settings in the `vc_static_shell`, use the following command:

```
%vc_static_shell> printvar (or)

%vc_static_shell> report_app_var
```

The `printvar` command reports all variables including user-defined variables while the `report_app_var` command reports only the VC Static and Formal application variable settings.

Example 1

```
%vc_static_shell> printvar

......

......

....

    lint_enable_coverage_flow = "false"

    lint_enable_pgpins    = "false"

    lint_enable_smart_tag_execution = "false"

    lint_functional_mode = "false"

    lint_ignore_encrypted_violations = "false"

    lint_ignore_syncreset_for_asyncflop = "false"

...

...

....

.....
```

Example 2

```
vc_static_shell> report_app_var *lint*
Variable           Value     Type     Default     Constraints

---------         --------- ------- ---------- ---------------
enable_lint       false      bool     false
lint_debug          0         int      0
lint_enable_
coverage_flow      false      bool     false
lint_enable_
pgpins             false      bool     false
lint_enable_smart
_tag_execution     false      bool     false
```

```
lint_functional_
mode                  false      bool    false
```

Example 3

```
vc_static_shell> report_app_var lint_functional_mode
-verbose

Variable    Value     Type    Default    Constraints

--------- --------- ------- ---------- ----------------

lint_
functional_
mode         false     bool    false

# Enables formal aware lint
```

You can change the default behavior of VC SpyGlass Lint by changing the default settings of the application variables. You can use the `set_app_var` command to change the setting of an application variable. The following example shows how to set a variable:

```
vc_static_shell> set_app_var lint_enable_coverage_flow true
```

Reading the Liberty Files

For pure RTL designs, supplying liberty files is not necessary, However, supplying liberty files is necessary for RTL designs with pre-instantiated cells and for most logical/physical netlist designs. You must provide all the required liberty files using `search_path` and `link_library` application variables before reading the design. After the `link_library` and `search_path` commands are set, read the design using the `read_file` command.

The search_path and link_library Variable

The `search_path` application variable specifies a sequence of directories where VC SpyGlass Lint looks for the library (.db) files. The specified directories are searched before a new library file is loaded.

```
%vc_static_shell> set_app_var search_path <list of all the paths>
```

- Specify all the paths where VC SpyGlass Lint should search for the library files and design files. The paths might be absolute or relative to the directory from which VC SpyGlass Lint is invoked.
- If multiple paths are present, specify the paths as space separated values in double quotation marks.
- The `search_path` variable supports environment variables.
- The `search_path` variable does not support wildcard characters.

The `link_library` application variable specifies a list of .db library files to be searched when a cell instantiation is to be resolved.

```
%vc_static_shell> set_app_var link_library <list of .db files>
```

- Specify all the library files that should be read.
- Only Liberty .db files (not .lib files) are read in by the tool.
- If multiple .db files are present, specify the paths as space separated values in double quotation marks.
- The `link_library` variable does not support environment variables.

Example

```
%vc_static_shell> set_app_var search_path “. path1 path2 …"
%vc_static_shell> set_app_var link_library “lib1 lib2 … libN"
```

Using Black Boxes in VC SpyGlass Lint

A black box is either categorized as an automatic black box or a user-defined black box as described below:

- A user defined black box is created using the Tcl command `set_blackbox`. This needs to be done before using the `analyze`/ `read_file` commands.

For example, to black-box the module `moduleA`, use the following command:

```
set_blackbox -designs {moduleA}
```

```
read_file -format verilog -top top <design file name.v>
```

- An automatic black box is marked by the tool automatically. To see the designs marked as a black box, use the following command:

```
vc_static_shell> get_blackbox -designs
```

VC SpyGlass Lint issues the following message for each black boxed module instance:

```
Note-[SM_BB_SKIP] Skipping blackboxed Module/Entity
```

In a design that contains one or more black boxes, the black boxes are elaborated with the following implicit hardware:

- Each input port of a black box is connected inside the corresponding module entity to one input of one implicit combinatorial gate (of undefined type).
- Each output port of a black box is driven inside the corresponding module by one output of one implicit combinatorial gate (of undefined type).
- Each inout port of a black box has both characteristics above as an input and an output port.

VC SpyGlass Lint does not perform tag checking for black-boxed modules and all modules that are instantiated in a black-boxed module.

Reading the Design

VC SpyGlass Lint reads design in RTL (verilog, VHDL, SystemVerilog) and netlist (verilog) formats.

VC SpyGlass Lint provides the following commands to read a design:

- `read_file`: Read in design source files, and link design in memory. This command can be used to load design in a single language (Verilog/SV or VHDL). Using this command, you can specify all source files in one command in a single language environment. The files get analyzed and then elaborated. Upon completion of the command the complete design has been loaded and is ready to be used. The command returns 1 on success and 0 on failure.

  Syntax:

  ```
  %vc_static_shell> read_file -help

  Usage: read_file    # Reading design files

  [-top <top_design>]         (Name of the top design)

  [-library <library_name>]   (Remaps work library to library_name)

  [-define <list of verilog defines>](Verilog/SV defines)

  [-work <work_library>]      (alias for -library)

  [-netlist]                  (Verilog Netlist Reader)

  [-parameters <comma-separated list of ordered or named
                               parameters>](design parameters)

  [-vcs <vcs command line>]   (VCS Command line for reading design)

  [-vcs_elab <vcs elaborate command line>](VCS Command line for
                               elaborating design)

  [-format <file format>]     (Verilog/SV defines: Values: verilog,
                               sverilog, vhdl, mdb)

  [-sva]                      (Process SVA/PSL during compilation
                               using 2009 semantics)

  [-sva2005]                  (Process SVA/PSL during compilation
                               using 2005 semantics)

  [-v2kconfig <configuration-name>](Specifies the v2k
                               configuration)

  [-buildTop <dut name>]      (Specifies the DUT down from which
  ```

```
                                  synthesis model is generated)
[-multi_step]                    (Load design in multi-step mode)
[-cov <metric_type>]             (Enables coverage instrumentation
                                  during compilation)
[-llk <llk_type>]                (Creates livelock goals during
                                  compilation)
[-aep <aep_type>]                (Enables AEP extraction during
                                  compilation)
[-inject_fault <fault_type>](Injects behavioral faults in the
                                  design for doing sign-off with
                                  formal)
[-j <number_of_processes>]       (Specifies the number of processes to
                                  use for parallel compilation: Value
                                  >= 1)
[slist]                          (List of input files)
```

NOTE: *For details on how to compile a design using the VCS standard switches, see the VCS® MX/VCS® MXi™ User Guide. You can download this document from Solvnet.*

- `analyze`: Analyzes the specified HDL source files and stores the design templates into the specified library in a format that is ready to elaborate to form linkable cells of a full design. Using this command, you can specify multiple source files in a single language in one command. On completion of the command, all specified files are analyzed and are ready for elaboration. The command returns 1 on success and 0 on failure.

  Syntax:

```
%vc_static_shell> analyze -help

Usage: analyze    # Analyze the source files

[-format <file_format>]          (Specifyfileformat:
                                  Values: verilog, vhdl, sverilog,
                                  sysc, spi)
[-library <library_name>]        (Remaps the work library to
                                  library_name)
[-work <library_name>]            (Remaps the work library to
                                  library_name)
[-define <define_macros>]        (Spcify list of top-level macros)
```

```
[-vcs <vcs_cmd>]          (VCS Command line for reading design)
[design_file_list]        (List of source files)
```

NOTE: *When you use the -vcs switch, the vlogan and vhdlan arguments and switches must be enclosed in curly braces.*

Examples:

```
analyze -format verilog {test.v}
analyze -format vhdl {test.vhd}
analyze -format vhdl -vcs {-vhdl08} test.vhd
analyze -format vhdl -vcs {-f filelist_vhdl.f}
analyze -format verilog -vcs {+define+g90d -f sources_ng.f}
analyze -format verilog -vcs { +incdir+src/./sim_1 -f
sources_ng.f}
analyze -format verilog -vcs { +incdir+src/./sim_1 -f
sources_ng.f}
```

- `elaborate`: Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration. Using this command, the user can elaborate design from pre-analyzed design files, from a specified top module. This command returns 1 on success and 0 on failure.

  Syntax

```
%vc_static_shell> elaborate -help
Usage: elaborate    # Elaborate the design, which is analyzed
using analyze command

[-work <library_name>]      (Specifies the library name to which
                             work is to be mapped)
[-library <library_name>]   (Specifies the library name to which
                             work is to be mapped)
[-architecture <arch_name>](Specifies the name of the
                             architecture)
[-parameters <param_list>]  (Specifies a list of design
                             parameters enclosed in quotation
                             marks)
[-file_parameters <file_list>](Specifies a list of files that
                             contain parameter specifications)
[-vcs <vcs_cmd>]            (VCS Command line for elaborating
                                  design)
  [-sva]                          (Process SVA/PSL during compilation
                                  using 2009 semantics)
  [-sva2005]                      (Process SVA/PSL during compilation
                                  using 2005 semantics)
  [-v2kconfig <configuration-name>](Specifies the v2k
                                  configuration)
  [-buildTop <dut name>]          (Specifies the DUT down from which
                                  synthesis model is generated)
  [-cov <metric_type>]            (Enables coverage instrumentation
                                  during compilation)
  [-llk <llk_type>]               (Creates livelock goals during
                                  compilation)
  [-aep <aep_type>]               (Enables AEP extraction during
                                  compilation)
  [-inject_fault <fault_type>](Injects behavioral faults in the
                                  design for doing sign-off with
                                  formal)
  [-j <number_of_processes>]      (Specifies the number of processes to
                                  use for parallel compilation: Value
                                  >= 1)
  design_name                     (Specifies the name of the design to
                                  build)
```

NOTE: *(1) If there is one design top, it must not be passed using vcs arguments, that is,* `elaborate –vcs {designtop}`*. It must be passed as follows:* `elaborate designtop`
(2) For a model with testbench, you must pass the arguments as follows:
`elaborate dut_top -vcs "tb_top"`
Where, "`dut_top`*" is the design top, and "*`tb_top`*" is the testbench top.*
(3) Set the `cdc_number_of_processes` *application variable before the* `sg_read_project` *command to specify if value of the* `-j` *argument of the* `elaborate` *command should be generated in the* `internal.tcl` *and* `vc_setup.tcl` *files created by the* `sg_read_project` *command.*

- `read_verilog`: Reads in one or more design or library files in Verilog format.

  Syntax:

```
%vc_static_shell> read_verilog -help
Usage: read_verilog    # Read one or more verilog files
[-netlist]                 (Use structural Verilog netlist
                            reader)
[-rtl]                     (Use RTL Verilog)
file_names                 (Files to read)
```

- `read_vhdl`: Reads in one or more designs or library files in VHDL format.

  Syntax

  ```
  %vc_static_shell> read_vhdl -help
  Usage: read_vhdl    # Read one or more vhdl files
  [-netlist]                 (Use structural VHDL netlist reader)
  file_names                 (Files to read)
  ```

- `read_sverilog`: Reads in one or more design or library files in SystemVerilog format.

  Syntax

  ```
  %vc_static_shell> read_sverilog -help
  Usage: read_sverilog    # Read one or more systemverilog files
  [-netlist]                 (Use structural Verilog netlist
                              reader)
  [-rtl]                     (Use RTL Systemverilog)
  file_names                 (Files to read)
  ```

Application Variables that Impact Reading a Design

There are few application variables that affect the design read and database generation. Before you start reading the design, ensure that you review and set these application variables as per your design read requirements.

- `analyze_skip_translate_body`
- `hierarchy_delimiter`
- `sh_continue_on_error`

- language_check_hierarchy_format
- enable_timing_arc_for_bbox_detection
- prefer_lib_or_rtl_model

For details on each of these application variables, refer to the man pages.

Performing VC SpyGlass Lint Checks

Performing VC SpyGlass Lint checks involves performing language checks, performing structural checks.

To perform LINT checks, use the following command:

```
%vc_static_shell> check_lint

%vc_static_shell> check_lint -help
Usage: check_lint  # Invoke VC SpyGlass Lint application
```

Analyzing Reports

After performing VC SpyGlass Lint checks, analyze the reports that VC SpyGlass Lint generates.

Working with Methodologies and Goals

GuideWare is the testing platform to check the functionality of various goals by running them on different types of designs. The designer keeps modifying and adding goals until the desired coverage is achieved from these goals.

This section explains the following topics:

- "Terminology"
- "Development Phases and Methodologies:"
- "RTL Handoff"
- "Layout Handoff"
- "Understanding GuideWare Goals"
- "Setting Up Methodology/Goals"
- "Running Custom Goals"

Terminology

This section defines some commonly used terms that have a specific meaning in the VC SpyGlass environment.

- Tag: In VC SpyGlass environment, a 'Tag' represents the atomic unit of RTL analysis and checking performed by the VC SpyGlass software. Although a 'Tag' can be configured, it cannot be further sub-divided to select what analysis is performed.
- Violation Message: A violation message (or simply a message) is unit of VC SpyGlass reporting. When a VC SpyGlass 'Tag' detects a design condition not consistent with the rule requirement, it reports each such occurrence as a (violation) message. In addition to text message, such report usually contains other supporting data, such as back-reference in RTL source code where such problem originates, schematic highlight of the problem, detailed tables and graphs (as in power activity over time), waveform for a formal 'witness' (such as a false path proven to be not a false path), and so on.
- Parameter: In VC SpyGlass environment, a 'Parameter' is like an option to a tag that dictates the tag behavior. Parameters are typically used to make the tag do specific or detailed analysis of the RTL.
- Goal: A VC SpyGlass goal is a collection of relevant tags that are grouped together to perform a specific task. In addition to the tag list, a goal may further configure the parameters and redefine severity labels assigned to these tags. VC SpyGlass software release contains a useful set of many widely applicable goals. However, a user may fine-tune existing goals or create new goals to meet their specific design and workflow needs.
- Methodology: A VC SpyGlass methodology is a set of relevant goals that are grouped together to achieve a particular design goal. In addition to software, these sub-methodologies contain detailed documentation to assist customer in understanding specific usage and debug nuances.
- Severity: A VC SpyGlass violation message is tagged with an attribute, called severity, which helps to identify the criticality of reported message, within the context of a goal and sub-methodology being run. VC SpyGlass supports four main severity classes: FATAL, ERROR, WARNING, and INFO. A VC SpyGlass tag or goal can define a (severity) text label belonging to one of the above classes, and attach it to a tag.

- Waivers: A VC SpyGlass 'waiver' is a method for user to review a tag (violation) message and flag a specific occurrence (or set of occurrences) as acceptable in context of their design and workflow. This is a very important mechanism to flag an apparently non-compliant design scenario as intended and verified by actual design or verification engineer. In the SoC design workflow, the VC SpyGlass waivers play a very significant role both in Block regressions and in Block handoff to SoC integration and implementation teams.
- SGDC: SGDC is an abbreviation for 'VC SpyGlass Design Constraints', and is used to capture additional designer intent of the block/SoC functionality which are not obvious at RTL/netlist. SGDC is used for capturing a wide variety of design intent, related with clock domain crossing, power, testability, etc.

Development Phases and Methodologies:

A VC SpyGlass methodology is a set of relevant goals which represents a sign-off phase of ASIC development. Methodology includes goals which are arranged with pre-defined parameter.

Development Levels.

Currently we use lint at two major development levels, block level SoC level.

Block level Development

The process of the development of a new RTL goes through progressive RTL refinement starting with simpler goals that meet the functional requirements, such as functional correctness and simulation and synthesis readiness of the code. As the RTL code and design constraints mature, the design goals evolve to performance, testability, and meeting handoff requirements. In this field of use, this design phase recommends three methodology flow.

SoC level Integration and Implementation

During SoC design or a subset of design (sub-system) that has been integrated by using various blocks, consistency across blocks is required.

This field of use involves checks related to inter-block/inter-IP issues. In addition, it ensures that block constraints are consistent with SoC constraints. In this field of use, this design phase recommends a four-stage flows.

Development phases

Initial RTL Development

The initial RTL design goal set contains a set of checks for the stage of the project when the RTL is still in coding development and may not be functionally complete. The idea is not necessarily to be clean all at once but provide a starting point for getting to the clean RTL.

The design team faces the following lint related challenges during this stage:

- Issues related with correct code capture
- Issues related with simulation and synthesis
- Issues with basic connectivity
- Issues related with basic structure like combinational loops and multiple drivers

Initial RTL Development on block level

During this stage, an initial version of the RTL is completed, and an initial set of SGDC constraints are available. This stage involves basic structural and sanity checks of the design (and constraints, wherever appropriate). In addition, issues related to connectivity, synthesizability, preliminary clocks, and reset integrity issues, such as glitches and clock-MUXing are also checked during this stage.

For this stage, methodology recommends a set of goals that can be used by individual RTL designers to correct the issues within their own desktop environment before simulation and synthesis tasks can begin. These goals are recommended to be used quite frequently. In some cases, designers use these goals before checking-in their RTL code. Waivers, if any, should be captured on an ongoing basis.

This stage may involve some micro-architectural changes related with bus widths, RAM/ROM usage, and clock phase/frequency refinements. It is important to ensure that the proposed micro-architectural changes are reflected in the RTL without any adverse impact on the implementation issues.

RTL Handoff

The rtl_handoff goals are a super-set of the initial_rtl goals. This stage contains the complete set of recommended RTL Handoff checks.

The design team faces the following lint related challenges during this stage:

- Issues related with verification regressions and associated bug fixes
- Issues related with incomplete handoff
- Providing closure on various implementation issues, such as synthesizability, timing, constraints, clock domain crossings, testability, congestion/routing, and power management

An incomplete handoff results in expensive and unpredictable error-prone iterations during the SoC integration phase. Handoff is assumed to be the hand-off from the RTL design team to the post-synthesis implementation team or hand-off to System Integration (sub-system or SoC) integration.

Since the hand-off process is typically iterative, it is not necessarily expected that all goals will be clean at the first hand-off, but at least the issues will be known and can be communicated to the consumers downstream.

RTL Handoff with block level

This is the final completion and handoff stage for the RTL. By this stage, it is assumed that the RTL has already been refined as per the methodology.

Most checks are applicable at this point before backend implementation begins. During this stage, the micro-architecture and majority of the logic is stable. VC SpyGlass goals are used to perform handoff checks with appropriate waiver definitions.

At this milestone, the block is expected to be clean and all the necessary inputs are expected to be in place before you perform the final VC SpyGlass run. It is also expected that the user is able to share the setup, constraints, waivers, reports, and so on, with the customer.

RTL Handoff with SoC level

During this stage, the SoC/sub-system integration team assembles the RTL blocks and IPs to form a SoC/sub-system. These RTL blocks are usually designed by different teams. The design teams may also use third party or legacy IPs.

The goals used during this stage target the following objectives:

- Check the complete design intent captured in individual blocks and their assembly
- Correct various inter-block issues, such as combinational loops and unconnected ports
- During this stage, the intent is to clean the RTL before production level synthesis begins.

Netlist Handoff

The netlist_handoff goals are designed to check post-synthesis netlist prior to layout. These checks are ideal for hand-off to the backend physical implementation team or ASIC handoff

Netlist Handoff on block level

This stage when the handoff RTL is synthesized, and netlist is handed off for backend implementation. All structural checks at RTL hand off are applicable here. In addition, certain ERC checks are appropriate at this stage. This netlist is used by many groups as a starting point for their tasks (such as floor planning, test insertion, power estimation, and reduction analysis). VC SpyGlass goals are used to perform handoff checks with appropriate waiver definitions.

The following table describes recommended Base VC SpyGlass goals for each of the three stages of the new Block/IP development.

Netlist Handoff on SoC level

This netlist is used by many groups as a starting point for their tasks (such as floor planning, test insertion, power estimation, and reduction analysis). During this stage, third party tools modify the preliminary netlist for scan and BIST insertion and power-related gating. This version of netlist is known as pre-layout netlist by most of the design teams. The goals used

during this stage ensure that the original design intent is not adversely impacted during these modifications. The goals and sub-methodologies recommended for this stage ensure the integrity of the complete SoC-level netlist from ERC perspective.

Layout Handoff

This is only available for SoC level.

During this phase, the SoC post layout netlist is closest to silicon. It is important to ensure final integrity of this post-layout netlist before tape-out.

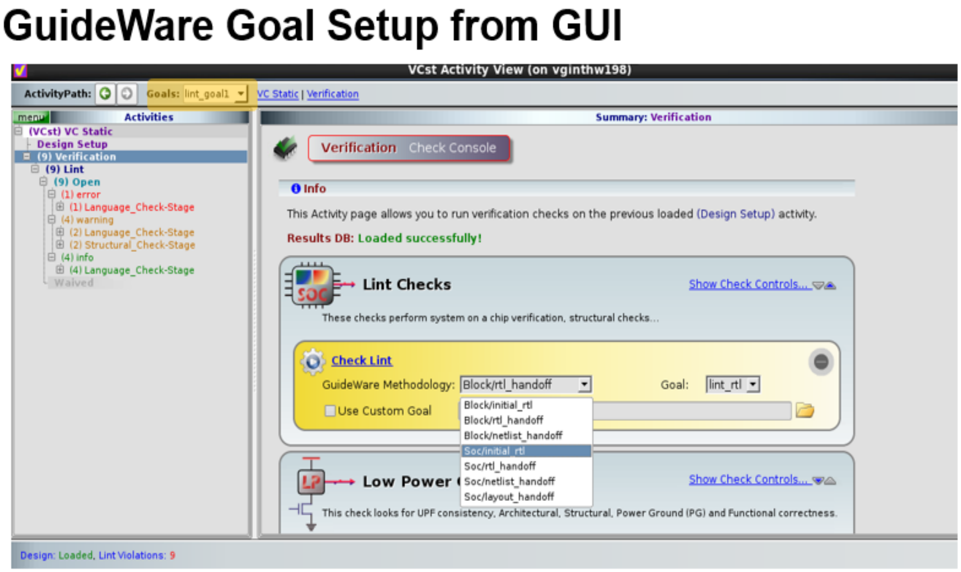
Recommended goals allow the designer to ensure integrity of post-layout netlist during the ECOs and before the final handoff for tape-out.

Understanding GuideWare Goals

A goal is a collection of relevant tags that are grouped together to perform a specific task. In addition to the tag list, a goal may further configure the tag parameters and redefine severity labels assigned to these tags. VC SpyGlass Lint contains a useful set of many widely applicable goals. However, you can fine-tune existing goals or create new goals to meet their specific design and workflow needs.

Setting Up Methodology/Goals

To view the default methodology and goal, select the expand button for Check Lint panel, as shown in the following figure:



GuideWare Goal Setup from GUI

This Activity page allows you to run verification checks on the previous loaded (Design Setup) activity.

Results DB: Loaded successfully!

These checks perform system on a chip verification, structural checks...

GuideWare Methodology: Block/rtl_handoff Goal: lint_rtl

□ Use Custom Goal

This check looks for UPF consistency, Architectural, Structural, Power Ground (PG) and Functional correctness.

Design: Loaded, Lint Violations: 9

FIGURE 1. Select Methodology

When you select a methodology in the GuideWare Methodology section, the goals in the Goal section change accordingly.

When you click on the Check Lint option after you select a methodology and a goal, the corresponding Tcl commands are run.

For example, in FIGURE 1. , the Block/inital_rtl methodology and the lint_rtl goal is selected. Now, when you click the Check Lint option, the following Tcl commands are run:

```
#configure_lint_methodology  -path $::env(VC_STATIC_HOME)/
auxx/monet/tcl/GuideWare//block/netlist_handoff/lint/ -goal
lint_netlist ;
```

The following table lists the supported methodologies in VC SpyGlass Lint and their corresponding goals:

TABLE 1 Goals and Methodologies

Methodology	**Goals**	**Description**
block/ initial_rtl	lint_rtl, lint_rtl_enhanced	This goal performs following checks on block RTL ● Basic connectivity issue in the design, such as floating input, width mismatch, etc. ● simulation issues in the design, such as incomplete sensitivity list, incorrect use of block/ nonblocking assignments, potential functional errors and possible simulation hang & simulation race cases. ● Unsynthesizable constructs in the design and code that can cause RTL vs. gate simulation mismatch. ● structural issues in design that affect the post-implementation functionality or performance of the design. Examples include multiple drivers, high fan-in MUX, and synchronous/ asynchronous use of resets. ● These checks should be run after every change in RTL code prior to code-check-in
block/ rtl_handoff	lint_rtl, lint_rtl_enhanced	This is similar to block lint_rtl but applicable only for netlist_handoff.
netlist_handoff	lint_netlist	This is similar to lint_rtl but applicable only for netlist_handoff.

TABLE 1 Goals and Methodologies

Methodology	Goals	Description
soc/ initial_rtl	lint_rtl, lint_rtl_enhnced	This goal performs following checks on SoC RTL ● Basic connectivity issue in the design, such as floating input, width mismatch, etc. ● simulation issues in the design, such as incomplete sensitivity list, incorrect use of block/ non-blocking assignments, potential functional errors and possible simulation hang & simulation race cases. ● unsynthesizable constructs in the design and code that can cause RTL vs. gate simulation mismatch. ● structural issues in design that affect the post-implementation functionality or performance of the design. Examples include multiple drivers, high fan-in MUX, and synchronous/ asynchronous use of resets. These checks should be run after every change in RTL code prior to code-check-in.
soc/ rtl_handoff		
soc/ netlist_handoff	lint_netlist	This is similar to SoC lint_rtl but applicable only for netlist_handoff
soc/ layout_handoff	lint_netlist	This goal checks basic connectivity issues in the design, such as floating input and width mismatch. These checks should be run after every change. This goal performs netlist integrity checks such as connectivity, simulation and structural issues. This goal must be run when synthesis is complete and after each major step in SoC implementation, such as scan-insertion, and power-gating.
lint_formality	block/ initial_rtl	This goal clubs all the formality related rules and helps user to run all these formality related rule in one go.
	block/ rtl_handoff	Similar to block lint_formality but applicable to netlist handoff

TABLE 1 Goals and Methodologies

Methodology	Goals	Description
lint_dccompat	block/initial_rtl	This goal clubs all the rules to perform lint check for DC compatibility and helps user to run all these rule in one go.
	block/rtl_handoff	similar to block lint_dccompat but applicable for netlist handoff.
lint_synth	block/initial_rtl	This goal clubs all the rules to perform earlier identification of simulation synthesis mismatch and helps user to run all these rules in one go.
	Block/rtl_handoff	Similar to block lint_synth but applicable for netlist handoff.

The following table lists the commands used to setup the goals/ methodologies:

TABLE 2 Goal/Methodology Setup Commands

Command	Description
`configure_lint_methodology`	Sets the current methodology.
`configure_lint_tag -enable -tag <tag-name> [-goal <goal-name>]`	Enables the tag for the respective goal name.
`configure_lint_tag_parameter -tag <tag-name> -parameter <param-name> -value <value> [-goal <goal-name>]`	Configure the parameter for the respective goal name.
`check_lint`	Invoke VC SpyGlass Lint checker to perform Lint Tag checks.

TABLE 2 Goal/Methodology Setup Commands

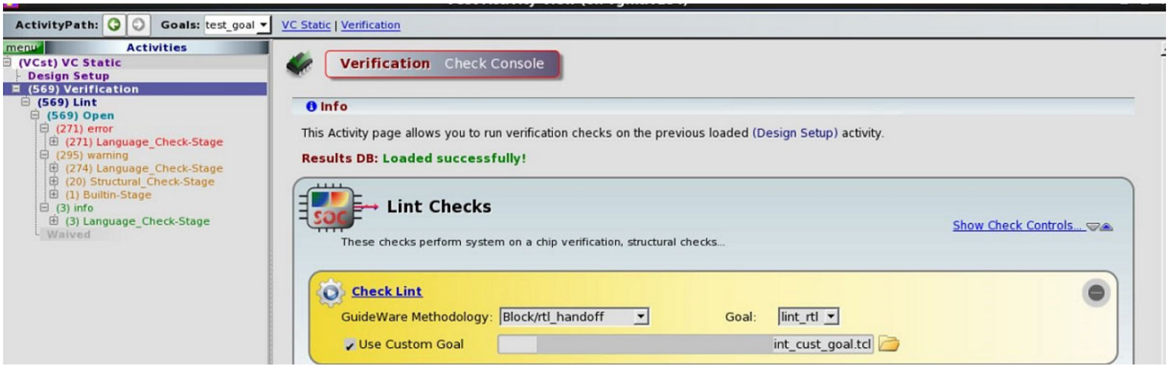
Command	Description
`configure_lint_setup [-goal <goal-name>] [-j <+ve integer>]`	To specify a goal to run in the early shift flow, use the following command: **Note**: Setup the current goal using -goal option and number of cores using -j (default: 4). If you are defining a custom methodology and using the name of the goal as standard goal name "lint_rtl or lint/lint_rtl". This gets conflicted with tool default goal name and tool picks up the default rule set. If you want to continue with using the name as "lint_rtl or lint/lint_rtl" they can use below command "configure_lint_methodology" and parse the setup file instead of "source goal_setup.tcl" configure_lint_methodology goal_setup.tcl configure_lint_setup -goal lint_rtl
`report_lint [-filter <goal-name>]`	Reports results for a given goal. Without this option report_lint reports all available results from every goal which has been executed.

NOTE: *If check_lint command is used without configure_lint_tag command, the lint_rtl GuideWare goal, is executed.*

Running Custom Goals

To run a custom goal, in the Lint Checks dialog, specify a Tcl file, which contains goal details.

The following figure illustrates how to specify the custom goal through the GUI:



This Activity page allows you to run verification checks on the previous loaded (Design Setup) activity.

Results DB: Loaded successfully!

These checks perform system on a chip verification, structural checks...

GuideWare Methodology: Block/rtl_handoff

Goal: lint_rtl

✓ Use Custom Goal int_cust_goal.tcl

FIGURE 2. Custom Goals

Using VC SpyGlass Lint

This section provides reference information for the built-in Tcl commands implemented in VC SpyGlass Lint. You can use these Tcl commands to configure tags, manage projects, and control your VC SpyGlass Lint Checker runs.

This section provides you the information on using Tcl Shell under the following topics:

- "Invoking VC SpyGlass Lint in Tcl Shell Mode"
- "Sourcing a Tcl Script in VC SpyGlass Lint"
- "Building a Design File"
- "Using Built-in Tcl Commands"
- "Running Lint Checks"
- "Reporting Violations"
- "Reporting Same or Similar Tags"
- "Using Multi-Cores"
- "Module-based Reporting"
- "Using Waivers in VC SpyGlass Lint"

Invoking VC SpyGlass Lint in Tcl Shell Mode

To run VC SpyGlass Lint, set the environment variable $VC_STATIC_HOME as follows:

```
setenv VC_STATIC_HOME <installation_directory>
set path = ($VC_STATIC_HOME/bin  $path)
```

Sourcing a Tcl Script in VC SpyGlass Lint

You can write the run script as a .tcl file, for example, vc_lint.tcl, and then pass this file to the command line as follows:

```
$VC_STATIC_HOME/bin/vc_static_shell -f vc_lint.tcl
```

When running VC SpyGlass Lint Tcl scripts the -restore option is also supported in VC SpyGlass Lint.

Building a Design File

You can compile and elaborate the RTL design in the following two ways:

- "Building a Design Using Analyze and Elaborate"
- "Building a Design With read_file"

Building a Design Using Analyze and Elaborate

The following Tcl commands shows how you can build an elaborated database in VC SpyGlass Lint using the analyze and elaborate commands:

```
search_path "./DB"

link_library "<library name.db>"

analyze -format verilog <design file name.v>

elaborate top
```

VCS mode to read files is also supported. Add the VCS options into the double quote followed by vcs as follows:

```
analyze -format verilog –vcs "-f file_list"

elaborate top
```

Building a Design With read_file

The following Tcl commands shows how you can build an elaborated database in VC SpyGlass Lint using the `read_file` command:

```
read_file -format verilog -top top <design file name.v>
```

This command is generally used for reading netlist design.

VCS mode format:

```
read_file -format verilog -top top –vcs {-f file_list}
```

Using Built-in Tcl Commands

The following built-in Tcl commands are available with VC SpyGlass Lint:

- check_lint
- configure_lint_tag
- configure_lint_tag_parameter
- report_lint
- waive_lint

Running Lint Checks

The following Tcl commands shows how you can run both language and structural checks in VC SpyGlass Lint:

```
configure_lint_tag  -enable  –tag <TAG_NAME>

analyze -format verilog <design file name.v>

elaborate top

check_lint
```

Reporting Violations

The following Tcl commands report all the violations into the report_hdl.txt file. If you just want to get the summary of violations, you can turn off the switch, -verbose.

By default, `report_lint` reports up to 100 violations for each tag, if you want to report all the violations, you can use option –limit 0 in `report_lint`.

```
set search_path "./"
set link_library " "
set quick_lint_mode true
configure_lint_tag -enable -tag <TAG_NAME>
analyze -format verilog <design file name.v>
elaborate top
check_lint
report_lint -file report_hdl.txt -verbose -limit 0
quit
```

Ignoring Encrypted Modules

To ignore the violations reported for the encrypted modules, use the `ignore_encrypted_module_violations` app var, as shown below:

```
set_app_var ignore_encrypted_module_violations true
```

When the value of the `ignore_encrypted_module_violations` app var is false, violations belonging to encrypted modules are reported.

Reporting Same or Similar Tags

This features enables reporting of same and similar VC SpyGlass Lint tags. To enable this feature, use the following app var:

```
set_app_var lint_report_same_similar_rules true
```

This generates the `samesimilarrulereport.rpt` report, which contains information on same or similar tags in VC SpyGlass Lint.

Using Multi-Cores

You can enable the multi-core feature using the following command:

```
configure_lint_setup -j <+ve integer >
```

NOTE: *By default, the early shift flow is set to ON in SGUM.*

Here, j represents the number of cores used during Lint checks. The default value of j is 4. You can override this value using the -j switch in the Tcl file.

NOTE: *The -j and -goal arguments of the check_lint will be deprecated in a future release. It is recommended to use -j and -goal arguments of the configure_lint_setup command.*

VC SpyGlass Lint Flow

VC SpyGlass Lint requires at least two cores to run. By default, one core is assigned to the language check and remaining cores are assigned to the structural checks. If more than one core is assigned to the language checks, the additional cores are taken out from structural checker.

Multiprocessing in the language check is not supported, by default.

To set the number of processes, by which the language check is split in lint, use the lint_no_of_lang_processes app var as shown below:

```
set_app_var lint_no_of_lang_processes <+ve integer>
```

where, <+ve integer> specifies the number of processes.

The default value of this app var is 1.

Functional Lint Flow

In the functional lint mode, at least three cores are required. By default, one core is assigned to the language check, one core is assigned to the structural check, and the remaining cores are assigned to the formal engine.

To set the number of processes by which the formal engine is split in functional lint, use the lint_no_of_formal_processes app var as shown below:

```
set_app_var lint_no_of_formal_processes <+ve integer>
```

where, <+ve integer> specifies the number of processes.

The default value of this app var depends upon the number of cores (j) and is equal to (j-2). By default, value of j is 4 and accordingly, the default value of this app var is 2.

The functional lint mode does not support multiprocessing in the language checks and therefore, only one core is assigned to the language check. If the number of cores assigned to the formal engine is reduced, those cores are added to the structural checker.

Violation Messages Reported for the Multi-core Feature

In case of an improper usage of multi core feature, VC Spyglass Lint automatically configures the optimum values that best fits the user requirement and generates an Error/Warning messages as shown in the following table:

TABLE 3 Violation Messages for Improper Usage of the Multi-core Feature

Scenario	Severity	Message
Total number of cores are not sufficient to allocate the required number of cores to formal checks, specified through the lint_no_of_formal_processes app var. Here, maximum possible number of cores are allocated to formal checks.	Warning	LINT_RESOURCE_ALLOCATION_ISSUE: Not enough cores available to honour the value set through lint_no_of_formal_processes app_var. Continuing run with optimal allocation possible!
Number of cores specified through the -j switch are less than the minimum number of cores required.	Error	LINT_MINIMUM_REQUIREMENT_FAILURE: Minimum <num> core required to run Lint. Aborting run.
The lint_no_of_formal_processes app var is used in non-functional lint mode.	Warning	LINT_REDUNDANT_APP_VAR: Command Ignored! This app var can be used only in functional lint mode.

TABLE 3 Violation Messages for Improper Usage of the Multi-core Feature

Scenario	Severity	Message
When the total number of cores are not sufficient to allocate the required number of cores to language checks, using the lint_no_of_lang_processes app var. In that, maximum possible number of cores will be allocated to language checks.	Warning	LINT_RESOURCE_ALLOCATION_ISSUE: Not enough cores available to honour the value set through lint_no_of_lang_processes app_var. Continuing run with optimal allocation possible!
When the lint_no_of_lang_processes app var is used for multiple language processes, in the functional lint mode	Warning	LINT_REDUNDANT_APP_VAR: Command Ignored! MultiProcessing is not supported in functional lint mode

Module-based Reporting

To generate module-based reporting in the moresimple.rpt report, perform the following steps:

1. Source Tcl proc in the vc_static_shell as shown below:

```
$vc_static_shell>> source monet/Release/auxx/monet/tcl/
lint_module_based_report.tcl
```

2. Fetch violations based on module as shown below:

```
$vc_static_shell>> get_module_violations -goal
<Name_of_goal> -session vcst_rtdb -useModel SGUM
```

This generates the module-based reports gets in the following directory:

```
vcst_rtdb/reports/module_based_reports/<$module>
```

Using Waivers in VC SpyGlass Lint

VC SpyGlass Lint provides a waiver mechanism that lets you selectively waive-off specific categories of violations based on your requirement. This functionality is provided with the waive_lint command. Using this command, you can select the violations to be waived-off based on the stage, family, severity or ID.

VC SpyGlass Lint provides the following waiver mechanisms:

- Native VC SpyGlass Lint Waivers
- Convert SpyGlass Waivers to VC SpyGlass Lint Waivers

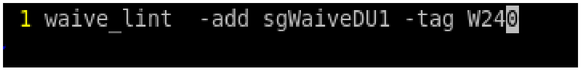
Native VC SpyGlass Lint Waivers

You can add basic waivers to waive any particular rule in VC SpyGlass Lint using the following command:

```
waive_lint -tag [rule_name] -add [waiver_name]
```

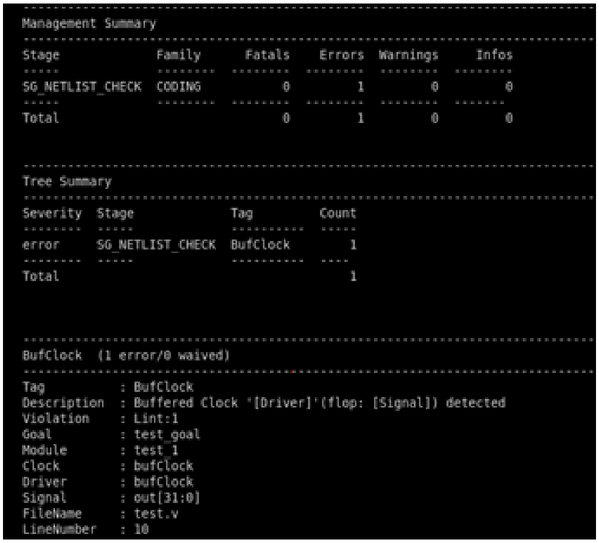
Example

Consider the following waive command:



```
waive_lint  -add sgWaiveDU1 -tag W240
```

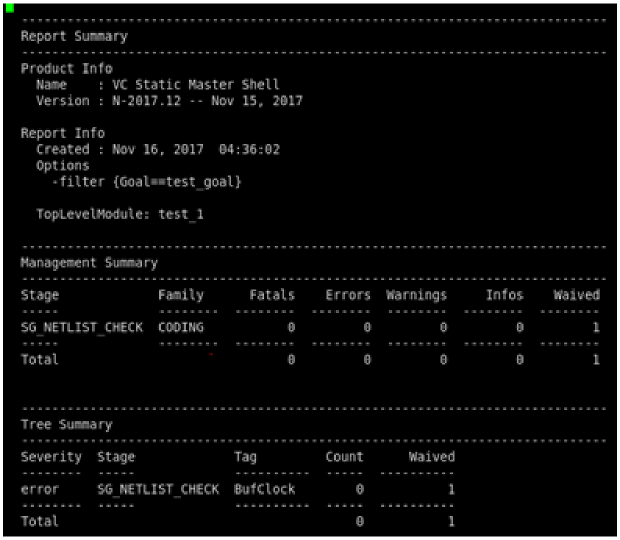
The following is the report file generated before applying the waiver:



```
-----------------------------------------------------------------------------
Management Summary
-----------------------------------------------------------------------------
Stage              Family      Fatals    Errors  Warnings     Infos
-----              --------  --------  --------  --------  --------
SG_NETLIST_CHECK   CODING           0         1         0         0
-----              --------  --------  --------  --------  -------
Total                               0         1         0         0


-----------------------------------------------------------------------------
Tree Summary
-----------------------------------------------------------------------------
Severity  Stage              Tag         Count
--------  -----              ----------  -----
error     SG_NETLIST_CHECK   BufClock        1
--------  -----              ----------  ----
Total                                        1



-----------------------------------------------------------------------------
BufClock  (1 error/0 waived)
-----------------------------------------------------------------------------
Tag          : BufClock
Description  : Buffered Clock '[Driver]'(flop: [Signal]) detected
Violation    : Lint:1
Goal         : test_goal
Module       : test_1
Clock        : bufClock
Driver       : bufClock
Signal       : out[31:0]
FileName     : test.v
LineNumber   : 10
```

FIGURE 3. Report Before Waiver

However, the following report is generated after applying the waiver:



```
-----------------------------------------------------------------------------
Report Summary
-----------------------------------------------------------------------------
Product Info
  Name    : VC Static Master Shell
  Version : N-2017.12 -- Nov 15, 2017

Report Info
  Created : Nov 16, 2017  04:36:02
  Options
    -filter {Goal==test_goal}

  TopLevelModule: test_1

-----------------------------------------------------------------------------
Management Summary
-----------------------------------------------------------------------------
Stage             Family      Fatals    Errors  Warnings     Infos    Waived
-----             --------  --------  --------  --------  --------  --------
SG_NETLIST_CHECK  CODING           0         0         0         0         1
-----             --------  --------  --------  --------  --------  --------
Total                              0         0         0         0         1


-----------------------------------------------------------------------------
Tree Summary
-----------------------------------------------------------------------------
Severity  Stage             Tag         Count      Waived
--------  -----             ----------  -----  ----------
error     SG_NETLIST_CHECK  BufClock        0           1
--------  -----             ----------  -----  ----------
Total                                       0           1
```

FIGURE 4. Report After Waiver

Using Tags in VC SpyGlass Lint

This section provides detailed procedures for how to configure and use the prepackaged tags to check your HDL code under the following topics:

- "About VC SpyGlass Lint tags"
- "Creating the Configuration File"
- "Configuring Tag Parameters"
- "Inferring Hanging Clocks"
- "Support for STARC and STARC02 Tag Mapping"

About VC SpyGlass Lint tags

The VC SpyGlass Lint tool offers a rich set of prepackaged tags. You can select and configure these tags to ensure that a design strictly follows those tags. A tag is a coding guideline that should be either followed or avoided strictly.

Creating the Configuration File

Create the configuration file where you can specify the list of tags that the design has to be checked against. You can select or deselect a tag based on your requirement.

Use the Tcl command `configure_lint_tag` to select/deselect a tag. For example,

```
configure_lint_tag -enable/-disable -tag <TAG_NAME>
```

Configuring Tag Parameters

Some of the tags have parameters that provide the capability to either extend or minimize the scope the tag. All the parameters of such tags have certain default values. If you wish to use the parameter to extend/reduce the scope of the tag, use the Tcl command `configure_lint_tag` with the -parameter option as follows:

```
configure_lint_tag -enable -tag <TAG_NAME>

configure_lint_tag_parameter -tag <TAG_NAME> parameter
<PARAMETER> value <VALUE>

analyze -format verilog <design file name.v>

elaborate top
```

Inferring Hanging Clocks

Use the lint_dump_hanging_clocks app var to infer hanging clocks during clock inference as shown below:

```
set_app_var lint_dump_hanging_clocks true
```

The default value of this app var is true. In this case, hanging clocks are also inferred in auto clock inference.

Set the value of the app var to false for backward compatibility.

Support for STARC and STARC02 Tag Mapping

VC SpyGlass lint currently only supports STARC05 tags. Therefore, in the default mode, VC SpyGlass Lint reports the following violation message when STARC/STARC02 tags are used:

```
ReportObsoleteTag "Select Tag 'STARC05-1.4.3.2' instead of
obsolete Tag 'STARC-1.4.3.2'"
```

In default mode, review the `ReportObsoleteTag` messages and make the changes in the tag set accordingly.

However, if you have STARC/STARC02 tags in the flow, you can enable automatic mapping of STARC/STARC02 tags to STARC05 in VC SG Lint.

To do so, use the `lint_enable_smart_tag_execution` app var, as shown below:

```
set_app_var lint_enable_smart_tag_execution true
```

When enabled, the following violation message is reported:

```
ReportObsoleteTag "Running Tag 'STARC05-1.4.3.4' instead of
obsolete Tag 'STARC-1.4.3.
```

The following tags are supported under auto-mapping under the `lint_enable_smart_tag_execution` app var:

TABLE 4 STARC/02/05 Rule Mapping

STARC/STARC02 Rule	Corresponding STARC05 Rule
STARC-1.4.3.2	STARC05-1.4.3.2
STARC02-1.1.5.2b	STARC05-1.1.5.2b
STARC02-1.2.1.2	STARC05-1.2.1.2
STARC02-1.3.1.7	STARC05-1.3.1.7
STARC02-2.4.1.4	STARC05-2.4.1.4
STARC02-2.4.1.5	STARC05-2.4.1.5
STARC02-2.5.1.2	STARC05-2.5.1.2
STARC02-2.5.1.6	STARC05-2.5.1.6
STARC02-2.5.1.7	STARC05-2.5.1.7

TABLE 4 STARC/02/05 Rule Mapping

STARC/STARC02 Rule	**Corresponding STARC05 Rule**
STARC02-2.5.1.8	STARC05-2.5.1.8
STARC02-2.5.1.9	STARC05-2.5.1.9
STARC02-3.3.1.4b	STARC05-3.3.1.4b
STARC02-3.3.2.2	STARC05-3.3.2.2
STARC02-3.3.2.3	STARC05-3.3.2.3
STARC02-3.3.3.1	STARC05-3.3.3.1
STARC02-3.3.6.2	STARC05-3.3.6.2
STARC-1.1.5.1	STARC05-1.1.5.1
STARC-1.1.5.2a	STARC05-1.1.5.2a
STARC-1.1.5.2b	STARC05-1.1.5.2b
STARC-1.1.5.2c	STARC05-1.1.5.2c
STARC-1.1.5.3	STARC05-1.1.5.3
STARC-1.1.5.4	STARC05-1.1.5.4
STARC-1.2.1.1a	STARC05-1.2.1.1a
STARC-1.2.1.1b	STARC05-1.2.1.1b
STARC-1.2.1.3	STARC05-1.2.1.3
STARC-1.3.1.3	STARC05-1.3.1.3
STARC-1.3.1.6	STARC05-1.3.1.6
STARC-1.3.1.7	STARC05-1.3.1.7
STARC-1.3.2.1	STARC05-1.3.2.1a
STARC-1.3.2.2	STARC05-1.3.2.2
STARC-1.3.3.4	STARC05-1.3.2.1b
STARC-1.4.1.1	STARC05-1.4.1.1
STARC-1.4.3.1b	STARC05-1.4.3.1c
STARC-1.4.3.2	STARC05-1.4.3.2

TABLE 4 STARC/02/05 Rule Mapping

STARC/STARC02 Rule	Corresponding STARC05 Rule
STARC-1.4.3.4	STARC05-1.4.3.4
STARC-1.4.4.2	STARC05-1.4.4.2
STARC-1.5.1.1	STARC05-1.5.1.1
STARC-1.5.1.2	STARC05-1.5.1.2
STARC-1.6.1.2	STARC05-1.6.1.2
STARC-1.6.2.1	STARC05-1.6.2.1
STARC-1.6.2.2	STARC05-1.6.2.2
STARC-1.6.2.2a	STARC05-1.6.2.2a
STARC-1.6.3.1	STARC05-1.6.3.1
STARC-1.6.3.2	STARC05-1.6.3.2
STARC-2.1.1.1	STARC05-2.1.1.1
STARC-2.2.1.3	STARC05-2.2.1.2
STARC-2.2.3.1	STARC05-2.2.3.1
STARC-2.3.1.1	STARC05-2.3.1.1
STARC-2.3.1.3	STARC05-2.3.1.3
STARC-2.3.2.1	STARC05-2.3.2.1
STARC-2.3.4.1	STARC05-2.3.3.2b
STARC-2.3.4.3	STARC05-3.3.1.4b
STARC-2.3.5.1	STARC05-2.3.5.1
STARC-2.3.6.1	STARC05-2.3.6.1
STARC-2.4.1.2	STARC05-2.4.1.2
STARC-2.4.1.3	STARC05-2.4.1.3
STARC-2.5.1.1	STARC05-2.5.1.1
STARC-2.5.1.2	STARC05-2.5.1.2
STARC-2.5.1.4	STARC05-2.5.1.4

TABLE 4 STARC/02/05 Rule Mapping

STARC/STARC02 Rule	Corresponding STARC05 Rule
STARC-2.5.1.5a	STARC05-2.5.1.5a
STARC-2.5.1.5b	STARC05-2.5.1.5b
STARC-2.5.2.1	STARC05-2.5.2.1
STARC-3.3.2.2a	STARC05-3.3.1.1
STARC-3.3.2.2b	STARC05-3.3.1.4a

VC SpyGlass Functional Lint

VC Spyglass Lint provides a powerful static Lint analysis solution in RTL verification industry. This tool is used to detect issues in the RTL code by using static analysis only. As a result of the issues found in the RTL code, the tool reports a violation to check the validity as desired in the static checks. With this new feature, referred as VC Spyglass Functional Lint, the results of traditionally dominant Linting is enhanced with formal technology. The objective is to leverage comprehensive and widely used lint checks in Formal flow while reducing the noise and improve results accuracy with formal techniques. This formal support is enabled only for Verilog/System Verilog designs only.

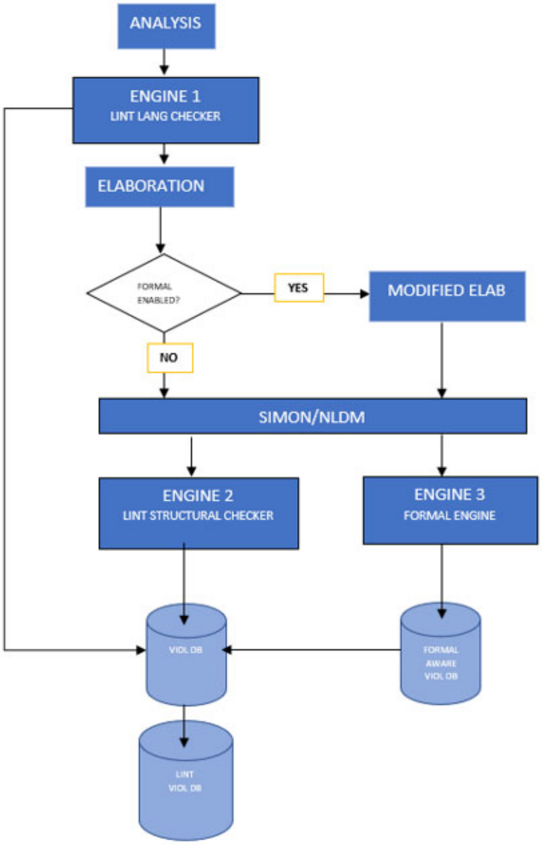
Key Features

Following are some of the key features of VC SpyGlass Functional Lint:

- Easy setup with automatic invocation of formal in existing lint setup
- Lint and Functional Lint checks can be run simultaneously with minimum change in the setup
- Vast Lint tag set with better design coverage and parameter flexibility
- Vast Lint tag set with better design coverage and parameter flexibility

Flow

The following figure illustrates the flow of VC SpyGlass Functional Lint:



Terminology

The following terminologies are used in VC SpyGlass Functional Lint:

- **Proven**: The violations, which are found as issue-free by VC SpyGlass Functional Lint tool, are referred as proven violations. These violations are removed from the standard reports.
- **Falsified**: The violations, which are found valid by VC SpyGlass Functional Lint tool, are referred as falsified violations. These violations are included in the standard reports.

Invoking VC SpyGlass Functional Lint

Use VC Spyglass Lint to invoke Formal analysis as follows:

```
# enable vc spyglass lint formal
```

```
set_app_var lint_functional_mode true
```

To run the functional lint analysis selectively only on specific tags, instead of the complete formal tag set, specify the -formal switch with the tag.

Example 1

```
set_app_var enable_lint true
set_app_var lint_functional_mode true

#### Enabling/Disabling translated Spyglass rules
configure_lint_tag -enable -tag "W164a" -goal test_goal
configure_lint_tag -enable -tag "W116" -goal test_goal
configure_lint_setup -goal test_goal
```

Example 2

```
set_app_var enable_lint true
set_app_var lint_functional_mode true

#### Enabling/Disabling translated Spyglass rules
configure_lint_tag -enable -tag "W164a" -goal test_goal -
formal
configure_lint_tag -enable -tag "W116" -goal test_goal
configure_lint_setup -goal test_goal

For Test setup 1 functional lint analysis will execute on
both W164a, W116 tags. For Test setup 2 it will execute only
on W164a tag.
```

The view_fl_viol_summary Command

You can generate the following reports using the view_fl_viol_summary command:

- *The lint_formal_viol_summary.txt Report*
- *The viol_breakup.csv Report*

The lint_formal_viol_summary.txt Report

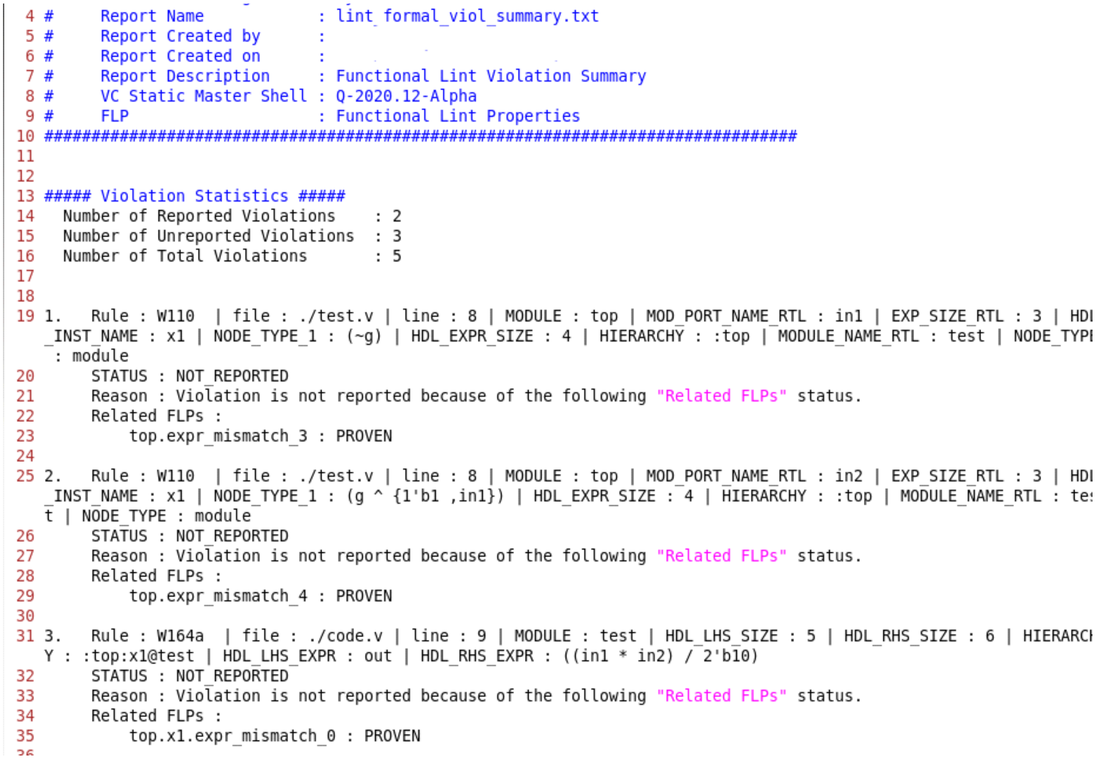
By default, the view_fl_viol_summary command generates the lint_formal_viol_summary report.

The lint_formal_viol_summary.txt report contains the details of proven and falsified violations, including user-waived violations, using the formal engine.

It contains the following details as a top summary:

- Number of Reported Violations
- Number of Unreported Violations
- Number of Total Violations

The lint_formal_viol_summary.txt file also contains information about the Status (Proven, Falsified), Reason and related FLP (Functional Lint Properties) for every violations.The following figure illustrates a sample:

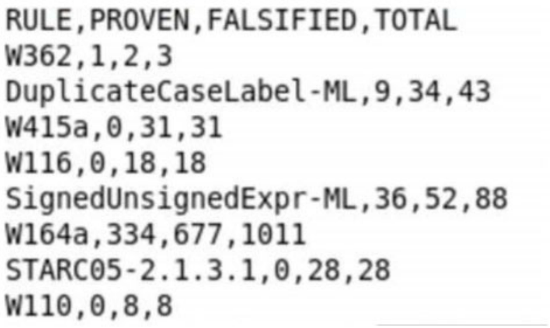


```
#     Report Name             : lint_formal_viol_summary.txt
#     Report Created by       :
#     Report Created on       :
#     Report Description      : Functional Lint Violation Summary
#     VC Static Master Shell  : Q-2020.12-Alpha
#     FLP                     : Functional Lint Properties
################################################################################


##### Violation Statistics #####
  Number of Reported Violations    : 2
  Number of Unreported Violations  : 3
  Number of Total Violations       : 5


1.   Rule : W110  | file : ./test.v | line : 8 | MODULE : top | MOD_PORT_NAME_RTL : in1 | EXP_SIZE_RTL : 3 | HDI
_INST_NAME : x1 | NODE_TYPE_1 : (~g) | HDL_EXPR_SIZE : 4 | HIERARCHY : :top | MODULE_NAME_RTL : test | NODE_TYPI
 : module
     STATUS : NOT_REPORTED
     Reason : Violation is not reported because of the following "Related FLPs" status.
     Related FLPs :
         top.expr_mismatch_3 : PROVEN

2.   Rule : W110  | file : ./test.v | line : 8 | MODULE : top | MOD_PORT_NAME_RTL : in2 | EXP_SIZE_RTL : 3 | HDI
_INST_NAME : x1 | NODE_TYPE_1 : (g ^ {1'b1 ,in1}) | HDL_EXPR_SIZE : 4 | HIERARCHY : :top | MODULE_NAME_RTL : tes
t | NODE_TYPE : module
     STATUS : NOT_REPORTED
     Reason : Violation is not reported because of the following "Related FLPs" status.
     Related FLPs :
         top.expr_mismatch_4 : PROVEN

3.   Rule : W164a  | file : ./code.v | line : 9 | MODULE : test | HDL_LHS_SIZE : 5 | HDL_RHS_SIZE : 6 | HIERARCH
Y : :top:x1@test | HDL_LHS_EXPR : out | HDL_RHS_EXPR : ((in1 * in2) / 2'b10)
     STATUS : NOT_REPORTED
     Reason : Violation is not reported because of the following "Related FLPs" status.
     Related FLPs :
         top.x1.expr_mismatch_0 : PROVEN
```

FIGURE 5. The lint_formal_viol_summary.txt Reportt

The viol_breakup.csv Report

Use the -csv switch of the view_fl_viol_summary command to generate the viol_breakup.csv report. This report generates a tag-wise breakup of the formally verified tags in a .csv format.The following figure illustrates a sample viol_breakup.csv report:



```
RULE,PROVEN,FALSIFIED,TOTAL
W362,1,2,3
DuplicateCaseLabel-ML,9,34,43
W415a,0,31,31
W116,0,18,18
SignedUnsignedExpr-ML,36,52,88
W164a,334,677,1011
STARC05-2.1.3.1,0,28,28
W110,0,8,8
```

FIGURE 6. The viol_breakup.csv Report

Analyzing VC SpyGlass Lint Results

This chapter is organized into the following sections:

Understanding VC SpyGlass Lint Violation Database

This section describes the following:

- Configuring Message Tags

Configuring Message Tags

VC SpyGlass LINT provides many LINT checks. All these checks have message tags and a predefined reporting format. Based on collective user feedback, by default, VC SpyGlass LINT has certain checks enabled and certain checks disabled. Also, the severity is predefined for each tag of a violation.

VC SpyGlass LINT provides the flexibility for you to pick and choose which checks are relevant for your design. You can change the default enable/ disable status of check and the severity of a check by using the *configure_lint_tag* command. In summary, you must configure the VC SpyGlass LINT tags in the following cases:

- When you want to permanently skip certain tags without the local administrative overhead of a waiver.
- When you want to change the severity of some messages between error and warning.

It is recommended that you use the *configure_lint_tag* command before reading a design.

NOTE: *Configuring the LINT checks for a given design/run is one of the most important review that must be done by the design engineer.*

The *configure_lint_tag* command uses the following syntax:

```
%vc_static_shell> configure_lint_tag -help

Usage: configure_lint_tag # Enable/Disable check tags for
VC_STATIC Lint checker

        [-tag <tag list>] (Define the tag(s) operated on)

        [-enable] (The tag which user enables)

        [-disable] (The tag which user disables)
```

```
   [-severity <fatal|error|warning|info>]
  (Sets the tag(s) severity level:  Values: all, fatal,
error, warning, info)
    [-module <module name or pattern>]
  (Tag will be enabled only for that module or module
pattern)
  [-clear] (Restores all tags to their original state)
  [-tcl] (Displays changes to the lint tag set in a TCL
format suitable for replay)

  [-regexp] (Allows regexp expressions in the tag list
(default glob-style))

  [-all] (Displays all messages, even with default enable and
severity status)

 [-verbose] (Displays short description for each message)

 [-formal]  (Run formal version of rule)

 [-goal <Goal-Name>]    (Specify Goal Name)
```

Examples:

The following example shows VC SpyGlass LINT reports of a design before and after configuration of certain tags/checks.

The following is the output without the use of the configure_lint_tag command:

```
%vc_static_shell>report_lint


  Management Summary

  -------------------------------------------------------------
-------------------

  Stage             Family      Fatals    Errors  Warnings
Infos

  -----              --------  --------  --------  --------  -
-------

  BUILTIN_CHECK     CODING           0         0         3
```

```
0
  LANGUAGE_CHECK    CODING          0        4        2
1
  STRUCTURAL_CHECK  CODING          0        1        1
0
  -----             --------  --------  --------  --------  -
-------
  Total                           0        5        6
1



  ----------------------------------------------------------
-------------------
  Tree Summary
  ----------------------------------------------------------
-------------------
  Severity  Stage             Tag                        Count
  --------  -----             ------------------------   -----
  error     LANGUAGE_CHECK    W122                           4
  error     STRUCTURAL_CHECK  UndrivenInTerm-ML              1
  warning   BUILTIN_CHECK     CheckDelayTimescale-ML         3
  warning   LANGUAGE_CHECK    W287a                          2
  warning   STRUCTURAL_CHECK  STARC05-1.4.3.4                1
  info      LANGUAGE_CHECK    ReportPortInfo-ML              1
  --------  -----             ------------------------   -----
  Total                                                     12
```

The following is the output after the use of the configure_lint_tag - severity error command:

```
Report Summary


------------------------------------------------------------
-----------------
  Management Summary
  ----------------------------------------------------------
-------------------
  Stage            Family     Fatals    Errors  Warnings
Infos
  -----             --------  --------  --------  --------  -
-------
  LANGUAGE_CHECK   CODING             0       4        0
0
  STRUCTURAL_CHECK CODING             0       1        0
0
  -----             --------  --------  --------  --------  -
-------
  Total                             0       5        0
0


  ----------------------------------------------------------
-------------------
  Tree Summary
  ----------------------------------------------------------
-------------------
  Severity  Stage             Tag                  Count
  --------  -----             -------------------  -----
  error     LANGUAGE_CHECK    W122                     4
  error     STRUCTURAL_CHECK  UndrivenInTerm-ML         1
  --------  -----             -------------------  -----
  Total                                                 5
```

Debugging Lint Violations Using Tcl

You can debug the violation reported by VC SpyGlass LINT using the Tcl commands described in the sections below. This section describes the following:

- Examples of Violation Fields
- Filtering Messages
- Operations on Tag Definitions

The report_lint command reports the messages generated after LINT analysis is performed on the design. The report_lint command is the main output command for clock domain crossing checks. By default, a summary of the messages and the waiver report is reported.

Syntax

```
vc_static_shell> report_lint -help

Usage: report_lint    # Report HDL checks information

        [-no_summary] (Suppresses summary information)

        [-list]     (List all messages in simple form)

        [-verbose]  (List all messages in detail form)

       [-limit <count>]  (Limit the number of output records
per rule)

        [-include_waived]  (Include waived messages in the
report)

        [-only_waived] (Report on waived messages)

        [-all_tags]  (Include all tested tags)

        [-sort]    (Report order match tree summary order)

        [-tag <tag>]  (Select violations based on tag)

        [-waived <list>] (Select violations based on waiver
name)

     [-id <rule>] (Select violations based on IDs)

    [-stage <stage>] (Select violations based on stage:
```

```
 Values: Builtin_Check, Custom,Debug_Cause, Debug_Cluster,
Language_Check, Lint_Formal_Check, Netlist, Quick_Lint,
Sg_Builtin_Check,
Spyglass_Check, Structural_Check)
     [-family <family>] (Select violations based on family:
      Values: CLK, CODING, CONN, NAMING, RST, SIM, SIMSYN,
SVSYN, SYN, TESTBENCH, UVM, XPROP, all)
   [-severity <list>]  (Select violations based on severity:
   Values: all, fatal, error, warning, info)
   [-filter <expression>] (Select violations based on
expression)
  [-regexp]  (Indicates filter expression type to be regular
expression)
[-file <filename>] (Write the results to the designated file)
[-append]  (Append results to the designated file)
 [-perf_report] (Generate a performance report)
 [-gen_empty]   (Generate a empty log file if has no
violations)
[-report <report-type>] (Generate a report depending upon the
set option)
 [-ignore_viol_state list of viol states] (Ignore count for
given states in report:
Values: Acknowledged, Ignore, NeedsInfo, Open, Waived,
Waived_Temp)
[-nocase] (Case will be ignored when matching string values)
[-include_viol_state <include_viol_state>] (Include messages
with given state(s) in the report:
Values: Acknowledged, Ignore, NeedsInfo, Open, Waived,
Waived_Temp)
[-viol_state <viol_state>]
(Violations belonging to given state(s) will be dumped in
report:
Values: Acknowledged, Ignore, NeedsInfo, Open, Waived,
Waived_Temp)
[-id_list] (Sets Matching violation id's as command result)
[-format <report format>] (Report will be dumped with this
format.)
[-separator <separator>] (Multiple field values will be
dumped using this separator)
[-include_compressed]  (Include compressed messages in the
report)
[-display_compressed  <compression name>] (Dump report
corresponding to given compression)
[-skip_full_path_for_waiver_file] (Displays only base name
for waiver file)
```

Use Model

Using the various options that the report_lint command provides, you can define the content of the report.

For example, if you specify report_lint in the shell, VC SpyGlass LINT displays a summary view of all messages, as shown below:

```
-------------------------------------------------------------
-----------------

  Management Summary

  -----------------------------------------------------------
-------------------

  Stage             Family     Fatals    Errors  Warnings
Infos

  -----              --------  --------  --------  --------  -
-------

  BUILTIN_CHECK     CODING           0        0        3
0

  LANGUAGE_CHECK    CODING           0        4        2
1
```

```
  STRUCTURAL_CHECK  CODING          0        1        1
0
  -----             --------  --------  --------  --------  -
-------
  Total                            0        5        6
1



  -----------------------------------------------------------
-------------------
  Tree Summary
  -----------------------------------------------------------
-------------------
  Severity  Stage             Tag                       Count
  --------  -----             ------------------------  -----
  error     LANGUAGE_CHECK    W122                          4
  error     STRUCTURAL_CHECK  UndrivenInTerm-ML             1
  warning   BUILTIN_CHECK     CheckDelayTimescale-ML        3
  warning   LANGUAGE_CHECK    W287a                         2
  warning   STRUCTURAL_CHECK  STARC05-1.4.3.4               1
  info      LANGUAGE_CHECK    ReportPortInfo-ML             1
  --------  -----     ------------------------  -----
  Total                                                   12
```

You can define the content of the report to make it verbose. The following shows an example of verbose reporting:

```
%vc_static_shell> report_lint -verbose -tag W287a -limit 0


  -----------------------------------------------------------
-------------------
```

```
  Management Summary
  ----------------------------------------------------------
-------------------
  Stage           Family      Fatals    Errors   Warnings
Infos
  -----           --------  --------  --------  --------  --
------
  LANGUAGE_CHECK  CODING           0         0         2
0
  -----           --------  --------  --------  --------  --
------
  Total                           0         0         2        0



  ----------------------------------------------------------
-------------------
  Tree Summary
  ----------------------------------------------------------
-------------------
  Severity  Stage           Tag      Count
  --------  -----           -------  -----
  warning   LANGUAGE_CHECK  W287a        2
  --------  -----           -------  -----
  Total                                  2



  ----------------------------------------------------------
-------------------
  W287a  (1 warning/0 waived)
  -----------------------------------------------------------
-------------------
  Tag            : W287a
  Description    : Input '[Signal]' of instance '[InstName]'
is undriven.[Hierarchy: '[HIERARCHY]']
  Violation      : Lint:7
  Goal           : test_goal
  Module         : ethmac
  FileName       : minsoc/rtl/verilog/ethmac/rtl/verilog/
ethmac.v
  LineNumber     : 639
  Statement      :   .dbg_dat(wb_dbg_dat0),
  Signal         : wb_dbg_dat0
  InstName       : ethreg1
  HIERARCHY      : minsoc_top.ethmac

  -----------------------------------------------------------
-------------------
  W287a  (1 warning/0 waived)
  -----------------------------------------------------------
-------------------
  Tag            : W287a
  Description    : Input '[Signal]' of instance '[InstName]'
is unconnected.[Hierarchy: '[HIERARCHY]']
  Violation      : Lint:3
  Goal           : test_goal
  Module         : minsoc_top
  FileName       : minsoc/rtl/verilog/minsoc_top.v
  LineNumber     : 406
  Statement      : adbg_top dbg_top  (
  Signal         : wb_rst_i
  InstName       : dbg_top
  HIERARCHY      : minsoc_top
```

By default, verbose reporting in the report_lint command limits 100 messages per violation tag. If you want to apply a different limit of the number of violations per ID, use the -limit option. For example, report_lint -limit 0 -verbose -file report_lint.log gives a verbose report of all message IDs.

Examples of Violation Fields

```
-------------------------------------------------------------
-----------------
  W287a  (1 warning/0 waived)
  -----------------------------------------------------------
-------------------
  Tag            : W287a
  Description    : Input '[Signal]' of instance '[InstName]'
is undriven.[Hierarchy: '[HIERARCHY]']
  Violation      : Lint:7
  Goal           : test_goal
  Module         : ethmac
  FileName       : minsoc/rtl/verilog/ethmac/rtl/verilog/
ethmac.v
  LineNumber     : 639
  Statement      :   .dbg_dat(wb_dbg_dat0),
  Signal         : wb_dbg_dat0
  InstName       : ethreg1
  HIERARCHY     : minsoc_top.ethmac
```

The short name, severity, and count of the messages are shown on the first line. The Tag field shows the tag name. The Description field shows the design-data dependent Description. The Violation field shows the violation ID which can be used for reference in this run.

Next, the important fields, which are contained in the dynamic Description, are shown. The remaining lines are less important fields which might be useful for debugging.

Filtering Messages

You can filter messages by using the `-filter <expression>` option in the `report_lint` command based on the following:

- Family
- Severity
- Filter based on debug fields
- Wildcards and expressions

Example 1

```
%vc_static_shell> report_lint -tag W287a

%vc_static_shell> report_lint -family CODING

%vc_static_shell> report_lint -severity error
```

Example 2

Usage of report_lint with –filter

Consider a scenario where you:

- Need a report of all messages related to STARC05-1.4.3.4
- Need a filter for a DesignObjSignal containing the string clk_adjust.clk_int

Use the following command to get the required report:

```
%vc_static_shell> report_lint -tag STARC05-1.4.3.4 -filter
{(DesignObjSignal=="clk_adjust.clk_int")}
```

Operations on Tag Definitions

You can change the violation tags/violations and the sequence of debug fields present in these violations. VC Static provides the following Tcl commands to operate tags/violations. For example, renaming of tag/ violation, reorder/disable debug fields of a tag/violation.

- **`get_tags:`** Returns tag names. If a single tag name is given as an argument, then it prints the same tag name. If used with a wildcard character it expands the wildcard character to a list of matching tags. It lists the tag/tags independently of the violations/tags present in the current run of the report database. It does not accept a list of tags, but wildcard characters are accepted. For example:

  ```
  vc_static_shell> get_tags W12*
  ```

  ```
  W122L W122L W122L W120 W120 W120 W120 W120 W128 W128 W123
  W123 W122 W129 W127 W126 W123 W123 W123 W121 W121 W121 W121
  W122 W122 W120 W120 W120 W120
  ```

- **`get_tag_info:`** Accepts a single tag at a time and lists the information about the tag. It does not accept a list of tags or wildcard characters. It runs independent of the violations/tags present in current run of report database. For example:

  ```
  vc_static_shell> get_tag_info  W123
  ```

  ```
  Lint info LANGUAGE_CHECK CODING disabled 0 0 1
  ```

  In the above example, the numeric characters indicate the following:

  - ❒ The first numeric character (`0` in the above example) indicates the total count of violations
  - ❒ The middle numeric character (`0` in the above example) indicates the total count of waived violations
  - ❒ The last numeric character indicates the count of built-in (`1` in the above example) and user-defined (not shown in the above example) tags.

- **`get_tag_fields:`** Lists all the fields for a tag. It does not accept wildcard characters and list of tags. It lists the tag fields independent of the violations/tags present in current run of the report database. For example:

```
vc_static_shell> get_tag_fields W287a
{Msg ID} Tag Signature Goal RTL_Instance FileName LineNumber
Statement Signal Module InstName ObjNodeName
```

- **`get_violation_tags`**: Returns ordered list of tags for the violations present in the report database. This command requires no arguments. For example:

  ```
  %vc_static_shell> get_violation_tags W287a
  ```

- **`rename_tag:`** Takes a single tag and replaces its name with the new alias. This command should be used BEFORE the `check*` commands. It does not accept a list of tags or wildcard characters. For example:

  ```
  %vc_static_shell>rename_tag W287a New_W287a
  ```

- **`disable_tag_field:`** Disables tag fields and prevents them from getting printed in the report. This command should be used only AFTER check* commands. It does not accept a list of tags or list of fields or wildcard characters. For example:

  ```
  %vc_static_shell> disable_tag_field W287a Signal
  ```

- **`reorder_tag_field:`** Generates a report where info fields appear just after the design element fields, use the reorder_tag_fields command to change the order of the fields reported. This command does not accept wildcard characters and list of tags. It lists the tag fields independent of the violations/tags present in the current run of the report database. For example:

  ```
  %vc_static_shell> set order [get_tag_fields W287a]
  %vc_static_shell> set new_order [lsort $order]
  %vc_static_shell> reorder_tag_fields W287a
  $new_order
  %vc_static_shell> report_lint -verbose
  ```

Reports Generated by VC SpyGlass Lint

VC SpyGlass LINT generates a summary report after each run. A sample of the generated report is shown below. For information on how to read and use the report, see Debugging Lint Violations Using Tcl.

```
Report Summary

Product Info

Name: VC Static Master Shell Version :

Report Info Created :

------------------------------------------------------------

Management Summary

------------------------------------------------------------

Stage       Family      Fatals    Errors  Warnings     Infos

-----      --------  --------  --------  --------  --------

BUILTIN_
CHECK      CODING            0         0         3         0

LANGUAGE_
CHECK      CODING            0         4         2         1

STRUCTURAL_
CHECK      CODING            0         1         1         0

 ----   --------  --------  --------  --------  --------

 Total                        0         5         6         1

------------------------------------------------------------

Tree Summary

------------------------------------------------------------

Severity   Stage       Tag                       Count

-------   -----        ------------------------  -----

error     LANGUAGE_CHECK    W122                          4
error     STRUCTURAL_CHECK  UndrivenInTerm-ML               1
warning   BUILTIN_CHECK     CheckDelayTimescale-ML          3
warning   LANGUAGE_CHECK    W287a                           2
warning   STRUCTURAL_CHECK  STARC05-1.4.3.4                 1
info      LANGUAGE_CHECK    ReportPortInfo-ML               1
--------  -----             ------------------------  -----
Total                                                      12
------------------------------------------------------------
UndrivenInTerm-ML  (1 error/0 waived)
------------------------------------------------------------
Tag                : UndrivenInTerm-ML
Description        : Detected undriven input terminal
[DesignObjSignal]
Violation          : Lint:2
Goal               : test_goal
Module             : ethmac
FileName           : minsoc/rtl/verilog/ethmac/rtl/verilog/
ethmac.v
LineNumber         : 572
Statement          : eth_registers ethreg1
DesignObjSignal    : ethmac.ethreg1.dbg_dat[31:0]
String1            :
------------------------------------------------------------
W122  (4 errors/0 waived)
------------------------------------------------------------
Tag             : W122
Description     : The signal/variable '[Signal]' (or some of
its bits) read in the block is not in the sensitivity
list[Hierarchy: '[HIERARCHY]']
```

```
Violation      : Lint:4
Goal           : test_goal
Module         : or1200_alu
FileName       : minsoc/rtl/verilog/or1200/rtl/verilog/
or1200_alu.v
LineNumber     : 212
Statement      : result = result_cust5;
Signal         : result_cust5
HIERARCHY      : minsoc_top.or1200_top.or1200_cpu.or1200_alu
------------------------------------------------------------
Tag            : W122
Description    : The signal/variable '[Signal]' (or some of
its bits) read in the block is not in the sensitivity
list[Hierarchy: '[HIERARCHY]']
Violation      : Lint:8
Goal           : test_goal
Module         : eth_registers
FileName       : minsoc/rtl/verilog/ethmac/rtl/verilog/
eth_registers.v
LineNumber     : 879
Statement      :         `ETH_DBG_ADR          :
DataOut=dbg_dat;
Signal         : dbg_dat
HIERARCHY      : minsoc_top.ethmac.ethreg1
------------------------------------------------------------
Tag            : W122
Description    : The signal/variable '[Signal]' (or some of
its bits) read in the block is not in the sensitivity
list[Hierarchy: '[HIERARCHY]']
```

```
Violation      : Lint:5
Goal           : test_goal
Module         : or1200_alu
FileName       : minsoc/rtl/verilog/or1200/rtl/verilog/
or1200_alu.v
LineNumber     : 222
Statement      :         result = result_csum;
Signal         : result_csum
HIERARCHY      : minsoc_top.or1200_top.or1200_cpu.or1200_alu
-------------------------------------------------------------
Tag            : W122
Description    : The signal/variable '[Signal]' (or some of
its bits) read in the block is not in the sensitivity
list[Hierarchy: '[HIERARCHY]']
Violation      : Lint:6
Goal           : test_goal
Module         : or1200_alu
FileName       : minsoc/rtl/verilog/or1200/rtl/verilog/
or1200_alu.v
LineNumber     : 255
Statement      :         result = flag ? a : b;
Signal         : flag
HIERARCHY      : minsoc_top.or1200_top.or1200_cpu.or1200_alu


-------------------------------------------------------------
CheckDelayTimescale-ML  (3 warnings/0 waived)
-------------------------------------------------------------
Tag            : CheckDelayTimescale-ML
Description    : Delay used without timescale compiler
```

```
directive
Violation     : Lint:11
Goal          : test_goal
Module        : minsoc_onchip_ram_top
FileName      : minsoc/rtl/verilog/minsoc_onchip_ram_top.v
LineNumber    : 118
Statement     :     ack_we <= #1 1'b1;
------------------------------------------------------------
Tag           : CheckDelayTimescale-ML
Description   : Delay used without timescale compiler
directive
Violation     : Lint:12
Goal          : test_goal
Module        : tc_mi_to_st
FileName      : minsoc/rtl/verilog/minsoc_tc_top.v
LineNumber    : 1326
Statement     :     req_r <= #1 3'd0;
------------------------------------------------------------
Tag           : CheckDelayTimescale-ML
Description   : Delay used without timescale compiler
directive
Violation     : Lint:10
Goal          : test_goal
Module        : minsoc_top
FileName      : minsoc/rtl/verilog/minsoc_top.v
LineNumber    : 285
Statement     :     rst_r <= #1 1'b0;
------------------------------------------------------------
```

```
STARC05-1.4.3.4  (1 warning/0 waived)
------------------------------------------------------------
Tag                     : STARC05-1.4.3.4
Description             : Clock signal '[DesignObjSignal]'
used as a non-clock (Used with name '[NonClkPath]')
Violation               : Lint:1
Goal                    : test_goal
FileName                : minsoc/rtl/verilog/
minsoc_clock_manager.v
LineNumber              : 27
Statement               :     clk_int <= ~clk_int;
DesignObjSignal         : clk_adjust.clk_int
NonClkPath              : clk_adjust.clk_int
Module                  : minsoc_clock_manager
DesignFlop1             : wb_rst
PathNodeObj
SignalInfo
DesignSignalName     : clk_adjust.clk_int
FileName             : minsoc/rtl/verilog/
minsoc_clock_manager.v
LineNumber           : 27
------------------------------------------------------------
W287a  (1 warning/0 waived)
------------------------------------------------------------
Tag            : W287a
Description    : Input '[Signal]' of instance '[InstName]' is
undriven.[Hierarchy: '[HIERARCHY]']
Violation      : Lint:7
Goal           : test_goal
```

```
Module         : ethmac
FileName       : minsoc/rtl/verilog/ethmac/rtl/verilog/
ethmac.v
LineNumber     : 639
Statement      :   .dbg_dat(wb_dbg_dat0),
Signal         : wb_dbg_dat0
InstName       : ethreg1
HIERARCHY      : minsoc_top.ethmac
------------------------------------------------------------
W287a  (1 warning/0 waived)
------------------------------------------------------------
Tag            : W287a
Description    : Input '[Signal]' of instance '[InstName]' is
unconnected.[Hierarchy: '[HIERARCHY]']
Violation      : Lint:3
Goal           : test_goal
Module         : minsoc_top
FileName       : minsoc/rtl/verilog/minsoc_top.v
LineNumber     : 406
Statement      : adbg_top dbg_top  (
Signal         : wb_rst_i
InstName       : dbg_top
HIERARCHY      : minsoc_top
------------------------------------------------------------
ReportPortInfo-ML  (1 info/0 waived)
------------------------------------------------------------
Tag            : ReportPortInfo-ML
Description    : Port Information for top design unit has been
```

```
generated. For details see report ReportPortInfo.rpt
Violation      : Lint:9
Goal           : test_goal
FileName       : ./vcst_rtdb/spyglass/vc_lint0/minsoc_top/
VC_GOAL0/spyglass_reports/morelint/ReportPortInfo
LineNumber     : 1
Statement      : #     Comment          : Report Top Level
Module Port Info
```

Appendix A - Supported Commands

This appendix briefly describes the SDC commands, Tcl commands, configure commands and application variables supported by VC SpyGlass Lint:

- Application Variables
- LINT Commands
- LINT Configure Commands
- Database Commands
- Common Commands
- Command Sanity Checks

Application Variables

This section describes the application variables used by VC SpyGlass Lint.

enable_lint

Type

string

Default Value

False

Description

This application variable configures the platform for running VC-SpyGlass LINT checks and to specify the license tier. The following are the valid values for this app var:

- true
- base
- false

enable_clk_rst_infer_potential

Type

bool

Default Value

True

Description

This application variable disables the optimization in clock tree and detects any extra potential clock roots.

infer_unique_bbox

Type

bool

Default Value

false

Description

Enables VC SpyGlass to infer and create a unique representative personality for all black boxes in the design instead of inferring multiple copies of black boxes for the same master definition.

language_check_hierarchy_format

Type

bool

Default Value

true

Description

This application variable changes the hierarchy format of language rules.

lang_check_report_input_path

Type

bool

Default Value

false

Description

When this application variable is set, it enables reporting of path (which can be a link) provided for RTL, in language checks.

lint_debug

Type

int

Default Value

0

Description

When this application variable is set, it will report different log messages for debug. This application variable can any value from 1 to 8.

lint_dump_hanging_clocks

Type

bool

Default Value

true

Description

When this application variable is set to true, the hanging clocks are also inferred in the auto clock inference.

lint_enable_coverage_flow

Type

bool

Default Value

False

Description

Enables functional lint coverage checks for the following rules:

- DeadCode-ML
- NoExitFsmState
- NotReachableFsmState
- MissingFsmStateTransition
- RegisterStuckInResetState-ML

lint_enable_pgpins

Type

bool

Default Value

false

Description

This application variable enables power and ground pin information to be taken into consideration from the specified physical libraries.

lint_enable_smart_tag_execution

Type

bool

Default Value

false

Description

Enables mapped smart rules tag.

lint_formal_disable_stage_name

Type

bool

Default Value

false

Description

Disables stage name modification of the supported lint checks in VC SpyGlass lint formal rule violations.

lint_functional_mode

Type

bool

Default Value

false

Description

Enables the formal mode for lint checks. Noisy lint checks like Width-Mismatch, Index-Overflow, DeadCode, and FSM will leverage formal technology.

lint_ignore_syncreset_for_asyncflop

Type

bool

Default Value

true

Description

When this application variable is set to true, synchronous resets are not detected for flops having asynchronous resets.

lint_ignore_redundant_field_waiver

Type

bool

Default Value

true

Description

Ignores non mandatory fields while waiver translation.

lint_load_goal_results

Type

bool

Default Value

false

Description

Load goal-specific results in activity viewer. By default it loads consolidated results for all the executed goals. When this app var is enabled, the activity viewer would load goal-specific results.

lint_memory_threshold

Type

bool

Default Value

false

Description

This application variable sets the threshold size (in bit-width) of a bus that must be recognized as the memory bus. If the size of an MDA in the design exceeds the specified value, it is recognized as the memory bus.

lint_no_of_formal_processes

Type

int

Default Value

-1

Description

Sets the number of processes in which the formal engine is split in the functional lint. It's value will be equal to j-2, j dependent, unless specified otherwise. Since j's default value is 4, so it's default value will be 2 here.

lint_no_of_lang_processes

Type

int

Default Value

1

Description

Sets the number of processes in which the language checks are split in lint.

lint_report_all_paths

Type

bool

Default Value

false

Description

When this application variable is set to true, VC SpyGlass lint prints every node of the path. When this application variable is set to false, VC SpyGlass Lint prints 3 nodes of one path.

lint_report_same_similar_rules

Type

bool

Default Value

false

Description

This app var enables reporting of same and similar lint rules.

lint_report_whole_path

Type

int

Default Value

3

Description

When this application variable is set to true, VC Lint reports the whole path in lint message. By default, only 3 elements are reported for one path.

lint_spyglass_waiver_report

Type

string

Default Value

empty string

Description

This app var is used to convert waivers during the sg_read_waiver by comparing with the SpyGlass waiver.rpt report.

lint_traverse_depth

Type

int

Default Value

5000

Description

When this application variable is set, you can limit the bit traverse depth in VC SpyGlass Lint.

quick_lint_mode

Type

bool

Default Value

false

Description

If enabled, VC Lint executes rules which are not dependent on Hardware Inference.

Command Line Example

```
set_app_var link_library "" ""
set_app_var quick_lint_mode true
set_app_var enable_verdi_debug true
compress_hdl -enable
configure_hdl_tag -enable -stage ""QUICK_LINT ""
analyze -format verilog -vcs ""test.v -sverilog ""
elaborate top -verbose
check_hdl -lang -structure
```

```
report_hdl -file report_soc.txt -verbose
quit
```

Example

```
module top(a, b);
input [2:1]a;
output [2:1]b;
reg [2:1]b;
always @ (a[1] or a[0]) // VC-Lint flags here
b<=a;
endmodule
```

Report file output

All quick lint rules are flagged in the report:

```
------------------------------------------------------
Management Summary
-------------------------------------------------------
Stage Family Fatals Errors Warnings Infos
----- -------- -------- -------- -------- --------
QUICK_LINT CODING 0 0 4 0
QUICK_LINT SIMSYN 0 0 1 0
QUICK_LINT SYN 0 1 0 0
----- ------- -------- -------- -------- --------
Total 0 1 5 0
```

elab_summary_report_max_inst

Type

int

Default Value

5

Description

Use this app var to customize printing number of instances per design unit in the elab_summary.rpt report. Specify an integer value as an input to the app var.

The default value of the app var is 5, that is, a maximum of 5 instances per design unit is displayed in the elab_summary.rpt report. Specify any positive integer value as an input.

If set to -1, it displays all instances.

If set to any other negative number then the default behavior is applied, that is, a maximum of 5 instances per design unit is displayed.

If set to 0, it displays zero instances.

Examples

The following example shows the usage of the elab_summary_report_max_inst command:

```
set_app_var elab_summary_report_max_inst 10
```

```
set_app_var elab_summary_report_max_inst 100
```

```
set_app_var elab_summary_report_max_inst -1
```

enable_generate_label_naming

Type

bool

Default Value

true

Description

This variable lists names in the report containing generate label for VHDL. Note that this app var is available only for language checks.

ignore_encrypted_module_violations

Type

bool

Default Value

false

Description

This variable lists names in the report containing generate label for VHDL. Note that this app var is available only for language checks.

enable_gw_optional_tag

Type

bool

Default Value

false

Description

When this application variable is set, GW optional tag is enabled.

report_all_hdl_errors

Type

bool

Default Value

false

Description

Use this app var to report all Verilog syntax errors in a single run. By default, only the first syntax error is reported but when this variable is set to true, all syntax errors are reported. For VHDL designs, all syntax errors are reported at once irrespective of this application variable.

Examples

The following example shows the usage of the report_all_hdl_errors command:

```
report_all_hdl_errors
```

See Also

report_all_hdl_errors

LINT Commands

This section describes the Lint-specific commands used by VC SpyGlass Lint.

check_lint

Description

Invoke VC SpyGlass Lint checker to perform Lint Tag checks

Syntax

```
check_lint
```

Examples

The following example shows the usage of the check_lint command:

```
check_lint
```

See Also

report_lint

report_lint

Description

Prints any violations identified while performing LINT checks.

Syntax

```
report_lint
   [-no_summary]
   [-list]
   [-verbose]
   [-limit <count>]
```

```
[-include_waived]
[-only_waived]
[-all_tags]
[-sort]
[-tag <tag>]
[-waived <list>]
[-id <tag>]
[-stage <stage>]
[-family <family>]
[-severity <list>]
[-filter <regular_expression>]
[-regexp]
[-file <filename>]
[-append]
[-perf_report]
[-gen_empty]
[-report <report-type>]
[-ignore_viol_state { list of violation states }]
[-include_viol_state { list of violation states } ]
[-viol_state { list of violation states } ]
[-nocase]
[-id_list]
[-format]
[-separator]
[-include_compressed]
[-display_compressed]
[-skip_full_path_for_waiver_file]
[-include_cause_viols]
```

Arguments

-no_summary

Suppresses the two summary tables which list the number of violations in each family and stage.

-list

In addition to the summary tables, print a one sentence description of each violation with the design data fields filled in. Useful for generating a file with one line per violation.

-list

In addition to the summary tables, print a one sentence description of each violation with the design data fields filled in. Useful for generating a file with one line per violation.

-verbose

In addition to the summary tables, print a number of lines of detail about each violation. This verbose format includes the description, basic design detail fields for the violation, and also detailed debugging fields for the violation. Useful for getting all details of the violation.

-limit <count>

When used with list or verbose mode, print only this number of violations for each tag. Useful to limit the file size for designs with a large violation count.

-include_waived

By default, any violation which is waived is not included in the report. Use this switch to include the waived messages in the report.

-only_waived

By default, any violation which is waived is not included in the report. Use this switch to invert the display so that only waived messages are included in the report.

-all_tags

Reports the tags that are run but resulted in 0 violations. This option is available for Tree Summary of report.

-sort

Use this switch to make report order match tree summary order in the report.

-tag <tag>

To focus on only certain tags, use this switch with a list of tag names. Only violations whose tag is on this list will be displayed.

-waived <list>

To focus on only certain waivers, use this switch with a list of waive names. Only violations which are waived by a waiver on this list will be displayed.

-id <tag>

To generate the report based on stage.

-stage <stage>

Use this switch with the list of stages to generate the stage-based report.

-family <family>

To focus on only messages from certain families, use this switch with a list of families.

-severity <list>

To focus on only messages with a certain severity, use this switch with a list of severities. The valid severities are: error, info, warning.

-filter <regular_expression>

This switch allows you to specify complex criteria-based on pattern matching. Only violations matching the filter expression will be shown. An expression may contain several terms separated with a double ampersand. Each term has a field name, a comparison operator, and a target string. The field name may be any field name shown in the verbose report; for a field inside a record, use a colon to separate the record path components. The comparison operator is any of the standard operators such as ==!= =~. See the examples section for examples.

-regexp

Use this switch to indicate that the filter expression type is a regular expression. The default is glob-style.

-file <filename>

Write the results to the designated file.

-append

Append results to the designated file.

-perf_report

Generate a performance report.

-gen_empty

Generate a empty log file if has no violations

-report <report-type>

Generate a report depending upon the set option.

-ignore_viol_state {list of violation states}

Ignore viol state(s) in report. Possible states are: Acknowledged, NeedsInfo, Open, Waived, Waived_Temp, Ignore

-include_viol_state {list of violation states}

Include violations for given states. Possible states are: Acknowledged, NeedsInfo, Open, Waived, Waived_Temp, Ignore.

-viol_state {list of violation states }

Show violations for given states. Possible states are: Acknowledged, NeedsInfo, Open, Waived, Waived_Temp, Ignore.

-nocase

Use this switch to indicate that case will be ignored when matching string values.

-id_list

Sets Matching violation id's as command result.

-format

Report will be dumped with this format.

-separator

Multiple field values will be dumped using this separator.

-include_compressed

By default, any violation which is compressed is not included in the report. Use this switch to include the compressed messages in the report.

-display_compressed

Dump report corresponding to given compression.

-skip_full_path_for_waiver_file

Specify this option to display only base name for waiver file.

-include_cause_viols

Specify this option to include cause violations reported in the Machine Learning Root Cause Analysis (MLRCA) flow.

Examples

The following example shows the usage of the report_lint command:

```
report_lint -verbose -file test.log -filter {(Module =~
"*top*")}
```

The following command generates a report using user given format and multiple field values are dumped using seperator.

```
report_lint -format "%ObjSrcNet%"  -separator ","
```

report_violations

Description

Prints any violations identified while performing LINT checks.

Syntax

```
report_violations
    - app {lint}
   [-no_summary]
   [-list]
   [-verbose]
   [-limit <count>]
   [-include_waived]
   [-only_waived]
   [-all_tags]
   [-sort]
   [-tag <tag>]
   [-waived <list>]
   [-id <tag>]
   [-stage <stage>]
   [-family <family>]
   [-severity <list>]
   [-filter <regular_expression>]
   [-regexp]
   [-file <filename>]
   [-append]
   [-perf_report]
   [-gen_empty]
   [-report <report-type>]
   [-ignore_viol_state { list of violation states }]
   [-include_viol_state { list of violation states } ]
   [-viol_state { list of violation states } ]
   [-nocase]
   [-id_list]
   [-format]
   [-separator]
   [-include_compressed]
   [-display_compressed]
   [-skip_full_path_for_waiver_file]
   [-include_cause_viols]
```

Arguments

-app {lint}

Specify lint to print any violation for Lint.

For details on remaining arguments see, *Arguments*.

Examples

The following example shows the usage of the report_violations command:

```
report_violations  -app {lint} -include_waived -gen_empty -
verbose -file report_hdl.txt
```

```
report_violations  -app {lint} -limit 2000 -verbose -
include_waived -file report_hdl.txt -gen_empty -report all -
filter {Goal==test_goal}
```

waive_lint

Description

Manages waivers for SoC violation.

Syntax

```
waive_lint
[-add <name>]
[-append <name>]
[-delete <name>]
[-delete_all]
[-tcl]
[-force]
[-comment <comment_string>]
[-stage <stage/list_of_stages>]
[-family <family/list_of_families>]
[-severity <list>]
[-tag <tag/list_of_tags>]
[-id <id/list_of_ids>]
```

```
[-filter <regular_expression>]
[-regexp]
[-nocase]
[-msg]
[-ip ip_module_name(s)]
[-preview]
[-not_applied]
[-ignore]
[-skip_full_path_for_waiver_file]
[-posix_regex]
```

Arguments

-add <name>

Add waiver (add mode) is used to create a new waiver. Name is required and must be unique. Comment, tags, ids, stage, families, severities and filter regular expressions are permitted.

-append <name>

Append waiver (append mode) is used to add additional selection criteria to an existing waiver. Name is required and must have been previously declared in add mode. Tags, ids, stages, families, severities and filter regular expressions are permitted. The comment option is not permitted.

[-delete <name>]

Deletes the named waiver (delete mode) from the set of waivers. No other options are permitted.

[-delete_all]

Deletes all waivers (delete mode) from the set of waivers. No other options are permitted.

[-tcl]

TCL mode displays all currently defined waivers in a form suitable for re-application at some later point in the same or sub-sequent session. No other options are permitted. You can use the redirect option to capture this output to a file.

-force

Creates a container for waive_lint append operations.

-comment <comment_string>

Attaches your comment_string to the waiver for descriptive purposes. The comment option may only be used in add mode.

-stage <stage/list_of_stages>

One or more stages for which this waiver applies. The stage option is only valid in add or append modes. Accepted values are Language_Check, Netlist, Quick_Lint and all.

-family <family/list_of_families>

One or more families for which this waiver applies. The family option is only valid in add or append modes. Accepted values are CODING, CONN, NAMING, SIM,SIMSYN, SVSYN, SYN, TEMP, XPROP and all.

-severity <list>

Waive violations based on severity. Accepted values are all, error, info and warning.

-tag <tag/list_of_tags>

One or more tags for which this waiver applies. The tag option is only valid in add or append modes.

-id <id/list_of_ids>

One or more ids for which this waiver applies. The id option is only valid in add or append modes.

-filter <regular_expression>

Used in conjunction with the -regexp boolean option the filter option may be used to select specific data fields from violations. The id option is only valid in add or append modes.

-regexp

Used in conjunction with the -filter option to distinguish between glob and Unix-style regular expression syntax. The -regexp option is valid in add or append modes.

-nocase

Ignores case while matching string values.

-msg

Waive violations based on the message pattern. Message-pattern will be matched with violation message dumped using report_lint -list.

-ip ip_module_name(s)

Waive violations based on the Module inside which it lies.

-preview

If this option is set, waiver will not be added into database and matching violation ids will be set as command output.

-not_applied

Display the waivers which do not apply.

-ignore

Waiver will be applied but will not be shown in waiver report.

-skip_full_path_for_waiver_file

Specify this option to display only base name for waiver file.

-posix_regex

Specify this option to enable POSIX regular expression matching in waivers.

Examples

The following creates two waivers, the first waiver being the union of two sets of violations. In this example, violations covered by the append selection criteria will take precedence over those covered by waive_2 since the append criteria was added to waive_1 which was declared before waive_2.

```
waive_lint -add filt1 -comment " My first waiver"   -tag
CODING_SIGNAL_LOWER_CASE -filter {(Module =~ "*top*")}
```

```
waive_lint -add Waive_error_tag  -comment " Waiving all error
severity rules"  -severity error
```

```
waive_lint -add Waive_fmly_CODING -comment " Waiver all the
rule tags of CODING family"  -family CODING
```

view_fl_viol_summary

Description

This command is used to generate summary report of formally verified Lint violations. It contains all the information regarding each violation verified using the formal engine. Successful execution of this command generates lint_formal_viol_summary.txt report in the run directory.

This command is supported in functional lint only. When the command is applied successfully, 1 is returned; 0 otherwise. Use the -csv switch of the view_fl_viol_summary command to generate the viol_breakup.csv report. This report generates a rule-wise breakup of the formally verified rules in a .csv format.

Syntax

```
view_fl_viol_summary
```

```
[-csv]
```

Arguments

[-csv]

Use this switch to generate the .csv report.

get_flp_summary

Description

Use this command to fetch various attributes of a property. Currently, the following attributes are supported:

- lint_information: Relevant lint information encapsulated in a particular property.
- engine: Name of the engine used to solve a particular property.
- solve_time: Time taken by the engine to solve the property.

`This command is supported in lint_functional_mode only.`

When the command is applied successfully, 1 is returned; 0 otherwise.

Syntax

```
get_flp_summary
    [-prop <prop_name>]
    [-attr <lint_information|solve_time|engine>]
```

Arguments

[-prop <prop_name>]

property name

[-attr <lint_information|solve_time|engine>]

Attribute to be checked

Examples

The following example shows how this command can be used to fetch the solve time of a property `top.m1.expr_mismatch_1`:

```
prompt> get_flp_summary -prop top.m1.expr_mismatch_1 -attr
solve_time
```

LINT Configure Commands

This section describes the commands for configuring VC SpyGlass Lint.

configure_lint_tag

Description

Configures VC Lint checker violation tags by enabling, disabling, changing severity.

Note: configure_lint_tag must be set before running any check_* commands.

Syntax

```
configure_lint_tag
[-tag <tag list>]
[-enable]
[-disable]
[-severity <fatal|error|warning|info>]
[-module <module name or pattern>]
[-clear]
[-tcl]
[-regexp]
[-formal]
[-goal <Goal-Name>]
[-all]
[-verbose]
[-disable_turbo]
```

Arguments

[-tag <tag list>]

Define the tag(s) operated on.

[-enable]

Enables the specified tag(s)

[-disable]

Disables the specified tag(s)

[-severity <fatal|error|warning|info>]

Sets the tag(s) severity level: Values: error, fatal, info, warning

[-module <module name or pattern>]

Rule will be enabled only for that module or module pattern

[-clear]

Restores all tags to their original state

[-tcl]

Displays changes to the lint tag set in a TCL format suitable for replay.

[-regexp]

Allows regexp expressions in the tag list (default glob-style)

[-formal]

Run formal version of rule.

[-goal <Goal-Name>]

Specify Goal Name.

[-all]

Displays all messages, even with default enable and severity status.

[-verbose]

Displays short description for each message

[-disable_turbo]

Disables the turbo mode for specified tag(s)

Examples

The following example shows some usages of configure_lint_tag command:

```
configure_lint_tag -enable -tag "RegOutputs" -goal test_goal
-severity warning
```

configure_lint_tag_parameter

Description

Sets the parameters for the VC Lint tags.

Syntax

```
configure_lint_tag_parameter
  -tag <tag_name>
  -parameter <parameter_name>
  -value <value>
  -goal <Goal-Name>
```

Arguments

-tag <tag_name>

Tag selected by the user. It is mandatory to specify the `-tag` argument.

-parameter <parameter_name>

Parameter selected by the user

-value <value>

Value of the parameter selected

-goal <Goal-Name>

Specify Goal Name

Examples

The following example shows the usage of the configure_lint_tag_parameter command:

```
configure_lint_tag_parameter -tag "RegOutputs" -parameter
CHECK_REGOUTPUT_MODULES -value "" -goal test_goal
```

configure_lint_functional_setup

Description

This command is used for passing various options for Formal Model Creation and Verification.

Syntax

```
configure_lint_functional_setup
   [-scope <block|chip>]
   [-seqDepth <seq_depth>]
   [-opWidth <op_width>]
   [-recipe <func_lint_recipe>]
   [-regInit]
   [-resetInit]
   [-print]
```

Arguments

[-scope <block|chip>]

Specify the scope for which formal verification needs to be done.

[-seqDepth <seq_depth>]

Specify the sequential depth upto which the fan-in cone of a property is to be considered.

[-opWidth <op_width>]

Specify the operator width upto which the fan-in cone of a property is considered.

-modDepth

Specify the module depth upto which the fan-in cone of a property is considered for performing the formal analysis.

[-recipe <func_lint_recipe>]

Specify the formal recipe to be used for solving properties.

[-regInit]

Specify whether register initialization should be done or not.

[-resetInit]

Specify whether the design should be brought to non-reset state before Formal Model Creation and Verification.

[-print]

Print all the arguments and their respective values.

Examples

The following example shows how this command can be used to set seq depth to 4, op width to 10 and scope to chip.

```
prompt>  configure_lint_functional_setup -seqDepth 4 -
opWidth 10 -scope chip
```

configure_lint_methodology

Description

Set options to specify current lint methodology.

If you want to use a custom methodology but standard goal name, such as, `lint_rtl` or `lint/lint_rtl`, it causes conflict with the default goal name and VC SpyGlass Lint picks up the default rule set.

To continue with using the name as `lint_rtl` or `lint/lint_rtl`, use the `configure_lint_methodology` command and parse the setup file instead of source `goal_setup.tcl` file, as shown below:

```
configure_lint_methodology goal_setup.tcl
```

```
configure_lint_setup -goal lint_rtl
```

Syntax

```
configure_lint_methodology
  [-path <Path Name>]
  [-goal <Goal-Name>]
```

Arguments

`[-path <Path Name>]`

Methodology path.

`[-goal <Goal-Name>]`

Specify Goal Name

Examples

The following example shows some usages of configure_lint_methodology command:

```
configure_lint_methodology -goal "test_goal"
```

configure_lint_rca

Description

Specify this command to enable/disable/modify root cause tags and their corresponding tags. Also, used to specify stage or pattern list

Syntax

```
configure_lint_rca
  -help
  [-rca <rca list>]
  [-enable]
  [-disable]
  [-tag <tag list>]
  [-clear]
  [-stage <Stage-Id>]
  [-pattern <pattern list>]
  [-severity <severity>]
```

Arguments

-help

View the help for the command.

[-rca <rca list>]

Define the RCA(s) for which tags are configured.

[-enable]

Specify the tag to enable.

[-disable]

Specify the tag to disable.

[-tag <tag list>]

Specify the tag(s).

[-clear]

Restores all the tags to their original state.

[-stage <Stage-Id>]

Specify stage id to consider tags from that stage.

[-pattern <pattern list>]

Specify the pattern to match.

[-severity <severity>]

Specify violations from which category need to be clustered. Supported for fatal, error, warning and info.

Examples

```
configure_lint_rca -enable -rca {DEBUG_LINT_COMMON_MODULE
DEBUG_LINT_OPEN_PORT} -tag {UnloadedOutTerm-ML}
```

configure_lint_setup

Description

Specifies which goal to run in early shift flow.

If you want to use a custom methodology but standard goal name, such as, `lint_rtl` or `lint/lint_rtl`, it causes conflict with the default goal name and VC SpyGlass Lint picks up the default rule set.

To continue with using the name as `lint_rtl` or `lint/lint_rtl`, use the `configure_lint_methodology` command and parse the setup file instead of source `goal_setup.tcl` file, as shown below:

```
configure_lint_methodology goal_setup.tcl
```

```
configure_lint_setup -goal lint_rtl
```

Syntax

```
configure_lint_setup
  [-goal <Goal-Name>]
  [-j <+ve integer>]
```

Arguments

[-goal <Goal-Name>]

Specify Goal Name

[-j <+ve integer>]

Number of cores to use for parallel analysis.

Examples

The following example shows some usages of configure_lint_setup command:

```
configure_lint_setup -goal "test_goal"
```

```
configure_lint_setup -goal "test_goal" -j 8
```

Database Commands

This section describes the database commands used by VC SpyGlass Lint.

all_clock_gates

Description

This command creates a collection of clock gating cells. If -clock_pins option is specified, this command creates a collection of clock gating cells clock pins instead of the collection of clock gating cells.

Syntax

```
all_clock_gates
[-no_hierarchy]
[-clock_pins]
```

Arguments

- `[ -no_hierarchy]`: Searches for clock gating cells or their clock pins at the current level of hierarchy. By default, the search is hierarchical.
- `[ -clock_pins]`: Creates a collection of clock gating cells clock pins. By default, this command creates a collection of clock gating cells.

all_clocks

Description

This command returns all clocks of the current design.

all_connected

Description

The `all_connected` command returns a collection of objects connected to the specified net, port, pin, net instance, or pin instance. A net instance is a net in the hierarchy of the design. A pin instance is a pin on a cell in the hierarchy of a design.

If the -leaf option is used, a list of leaf pins of the net is returned.

To connect nets to ports or pins, use the `connect_net` command. To break connections, use the `disconnect_net` command.

Syntax

```
all_connected <object>

[-leaf]
```

Arguments

- `<object>`: Specifies the object whose connections are returned. The object must be a net, port, pin, net instance, or pin instance.
- `[ -leaf]`: Specifies that only leaf pins are returned for a hierarchical net. For non-hierarchical nets, there is no difference in output.

Examples

The following example uses `all_connected` to return the objects connected to MY_NET:

```
prompt> all_connected MY_NET

prompt> connect_net MY_NET OUT3
Connecting net 'MY_NET' to port 'OUT3'.

prompt> connect_net MY_NET U65/Z
Connecting net 'MY_NET' to pin 'U65/Z'.

prompt> all_connected MY_NET
{OUT3 U65/Z}

prompt> all_connected OUT3
{MY_NET}

prompt> all_connected U65/Z
{MY_NET}
```

This example uses `all_connected` to associate net load capacitance with the net n47, which is connected to the pin instance C/Z:

```
prompt> set_load 0.147 [all_connected [get_pins U0/U1/C/Z]]
Set "load" attribute to 0.147 for net "U0/U1/n47"
```

all_designs

Description

The `all_designs` command returns a collection containing the designs in the current design hierarchy in bottom-up order. You must set the current design using the `current_design` command before using `all_designs`.

Syntax

```
all_designs
```

all_fanin

Description

The all_fanin command reports the fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the fanin of a sink if there is a path through combinational logic from the pin to that sink. The fanin report stops at the pins of registers (sequential cells).

Syntax

```
all_fanin
[-to <sink_list>]
[-startpoints_only]
[-only_cells]
[-flat]
[-levels <count>]
[-pin_levels <pin_count>]
[-step_into_hierarchy]]
```

```
[-stop <stop_point_list>]
```

Arguments

- `[ -to <sink_list>]`: Reports a list of sink pins, ports, or nets in the design and a fanin of each sink in the sink_list. If you specify a net, the effect is the same as listing all driver pins on the net.
- `[ -startpoints_only]`: Returns only the startpoints.
- `[ -only_cells]`: Results in a set of all cells in the fanin of the sink_list.
- `[ -flat]`: Specifies to function in the flat mode of operation. The two major modes in which all_fanin functions are hierarchical (the default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current sink are returned. Thus, pins within a level of hierarchy lower than that of the sink are used for traversal but are not reported.
- `[ -levels <count>]`: Stops traversal when reaching the perimeter of the search of count hops, where counting is performed over the layers of cells that are equidistant from the sink.
- `[ -pin_levels <pin_count>]`: Specifies the number of pins in the design.
- `[ -step_into_hierarchy]]`: You can only use this option in hierarchical mode and with either the -levels or -pin_levels option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the input pins are considered a single level away from the related output pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related output pins.
- `[ -stop <stop_point_list>]`: Specifies the custom stop points list at which the traversal must stop. You can specify a collection of names, any hierarchical name (pin, port, net, instance), design cell names or a collection of these. When a match is found in the traversal, the traversal stops at the specified point.

Examples

The following examples show the fanin of a port in the design. The design comprises three inverters in a chain named iv1, iv2, and iv3. The iv1 and iv2 inverters are hierarchically combined in a larger cell named ii2.

```
prompt> all_fanin -to tout
{ii2/hin iv3/in iv3/out tin ii2/hout tout}

prompt> all_fanin -to tout -flat
{ii2/iv1/U1/a ii2/iv2/U1/z tin iv3/U1/a ii2/iv1/U1/z
ii2/iv2/U1/a iv3/U1/z tout}
```

The following example shows the fanout of a port in the design. The design comprises the following three inverters in a chain named iv1, iv2, and iv3. The iv1 and iv2 inverters are hierarchically combined in a larger cell named ii2.

```
prompt> all_fanin -to tout
{"ii2/hout", "iv3/out", "ii2/hin", "iv3/in", "tin", "tout"}

prompt> all_fanin -to tout -flat
{"ii2/iv1/U1/Z", "ii2/iv1/U1/A", "ii2/iv2/U1/Z", "ii2/iv2/U1/
A", "iv3/U1/Z", "iv3/U1/A", "tin", "tout"}

prompt> all_fanin -to tout -stop ii2/hout
{"ii2/hout", "iv3/out", "iv3/in", "tout"}

prompt> all_fanin -to tout -stop {iv3/U1/A ii2/hout}
{"iv3/out", "tout", "iv3/out"}
```

all_fanout

Description

The `all_fanout` command reports the fanout of specified source pins, ports, or nets in the design. A pin is considered to be in the fanout of a sink if there is a path through combinational logic from that source to the pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified by using the -from source_list option.

Syntax

```
all_fanout

[-from <source_list>]

[-endpoints_only]
```

```
[-only_cells]
[-flat]
[-levels <count>]
[-pin_levels <pin_count>]
[-step_into_hierarchy]
[-stop <stop_point_list>]
[-exclude_resetless]
[-only_resetless]
```

Arguments

- `[ -from <source_list>]`: Specifies a list of source pins, ports, or nets in the design. The fanout of each source in the source_list is reported. If a net is specified, the effect is the same as listing all load pins on the net. The -clock_tree and -from options are mutually exclusive.
- `[ -endpoints_only]`: Returns only endpoints as a result.
- `[ -only_cells]`: Results in a set of all cells in the fanout of the source_list, rather than a set of pins or ports.
- `[ -flat]`: Specifies to function in the flat mode of operation. The two major modes in which all_fanout functions are hierarchical (the default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current source are returned. Thus, pins within a level of hierarchy lower than that of the source are used for traversal but are not reported.
- `[ -levels <count>]`: Stops traversal when reaching the perimeter of the search of count hops, where counting is performed over the layers of cells that are equidistant from the source.
- `[ -pin_levels <pin_count>]`: Specifies the number of pins in the design.
- `[ -step_into_hierarchy]`: You can use this option only in hierarchical mode with either the -levels or -pin_levels option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the output pins are considered a single level away from the related input pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design

were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related input pins.

- `[ -stop <stop_point_list>]`: Specifies the custom stop points list.
- `[ -exclude_resetless]`: Specifies to exclude fanout objects that are reset-less flops.
- `[ -only_resetless]`: Specifies to report only reset-less flops as fanout objects.

Examples

The following example shows the fanout of a port in the design. The design comprises the following three inverters in a chain named iv1, iv2, and iv3. The iv1 and iv2 inverters are hierarchically combined in a larger cell named ii2.

```
prompt> all_fanout -from tin
{iv3/out tout iv3/in ii2/hin ii2/hout tin}

prompt> all_fanout -from tin -flat
{tout ii2/iv2/U1/z ii2/iv1/U1/a iv3/U1/z iv3/U1/a
ii2/iv2/U1/a ii2/iv1/U1/z tin}

prompt> all_fanout -from tin -levels 1 -only_cells
{iv3 ii2}
```

all_inputs

Description

The all_inputs command returns a collection of all input or inout ports in the current design, unless one of the options limits the search. The all_inputs command is usually used with a command that places attributes on input ports. To get detailed information on ports in the current design, use the report_port command.

Syntax

```
all_inputs
```

Examples

The following example lists all input ports in the current design:

```
prompt> all_inputs
{A1 A2 BIDIR1}
```

The following example sets the drive value of all the input ports on the current design to 10:

```
prompt> set_drive 10.0 [all_inputs]
```

The following example marks with a multicycle value of 0 all paths from inputs having level-sensitive input delay relative to PHI1 to level-sensitive registers clocked by PHI1:

```
prompt> set_multicycle_path 0 \\
-from [all_inputs -clock PHI1 -level_sensitive] \\
-to [all_registers -data_pins -clock PHI1]
```

all_instances

Description

Create a collection of all instances of a design (stub)

Syntax

```
all_instances [<>]
```

```
[-hierarchical]
```

Arguments

- `[<>]`: Target design or lib cell name.
- `[ -hierarchical]`: Search for instances hierarchically

all_outputs

Description

The all_outputs command returns a collection of all output or inout ports in the current design, unless one of the options limits the search. This command is usually used with a command that places attributes on output ports. To get detailed information on ports in the current design, use the report_port command.

Syntax

```
all_outputs
```

Examples

The following example lists all output ports:

```
prompt> all_outputs
{OUT1 OUT2 BIDIR1}
```

change_link

Description

The `change_link` command specifies a design for which to change the link for a cell. If you specify a cell in the object list, the command changes it to one occurrence of the specified design. You can change the cell link only to a compatible design.For example, the design must have the same number of ports with the same name and direction as the cell or reference.

Syntax

```
change_link <object_name_list>

[-force]

[-all_instances]

[-pin_map <pin map table>]
```

Arguments

- `<object_name_list>`: Specifies the cells or references in the current design for which to change the link.
- `<design_name>`: Specifies the name of the design to which to link the cells or references in the object list.

- `[ -force]`: Enables the command to allow mismatched pin counts, as long as any mismatched cells in the object list have fewer pins than the specified design.
- `[ -all_instances]`: Enables the command to accept instance cells in the object list.
- `[ -pin_map <pin map table>]`: Specifies the pin mapping used to map old pin names to new pin names.The pin name must be the reference cell pin name.To specify the pin mapping, use the following Syntax {{old_pin1 new_pin1} ... {old_pin_n new_pin_n}} New pins maintain the same net connections as the corresponding old pins.

Examples

The following example creates a cell named cell1 under the sub design corresponding to the mid1 cell:

```
vc_static_shell> create_cell cell1 my_lib/AND2
[Info] ADD_CELL: Creating cell 'cell1' in design 'mid1'.
```

configure_mem_macro_inference

Description

This command is used to specify the constraint for Memory Macro Inference in synthesis.

Syntax

```
configure_mem_macro_inference

-mthresh int

-infer_1dmem

-skip_infer
```

Arguments

- `-mthresh int`: Use this option to specify the width of signal beyond which memory inference will get triggered.The default value is 4096.
- `-infer_1dmem`: Use this option to specify the memory inference for vector signal.

- `-skip_infer`: Use this option to skip the memory macro inference.

connect_net

Description

Connects the specified net to the specified pins or ports. This command connects a net to the specified pins or ports at the same hierarchical level. The net can be at any level of hierarchy but the pins or ports must be at the same level. A net can be connected to many pins or ports; however, you cannot connect a pin or port to more than one net. To disconnect objects on a net, use the disconnect_net command. To display pins and ports on a net, use either the all_connected or get_nets -of $net command. Multicorner-Multimode Support This command has no dependency on scenario-specific information.

Syntax

```
connect_net <net_name>
```

Arguments

- `<net_name>`: Specifies the net to connect. The net must be a scalar (single bit) net, and must exist in the current design.
- `<object_list>`: Specifies a list of pins and ports to which the net is to be connected. Pins and ports must be at the same hierarchical level as the specified net, and must exist in the current design. If a specified pin or port is already connected, the tool issues an error message. Use this option to specify the list of nets

Examples

The following example uses `connect_net` to connect net NET0 to ports A1 and A2 and pin U1/A. The `all_connected` command returns the objects connected to net NET0.

```
vc_static_shell> connect_net NET0 [get_ports {A1 A2}]
vc_static_shell> connect_net NET0 [get_pins U1/A]
vc_static_shell>all_connected NET0
{A1 A2 U1/A}
```

create_bus

Description

The create_bus command creates a bus object of the type port or net. The number of objects in the list determines the bus width. Buses appear as multibit ports on a design or as multiwire nets in a design. This command groups any number of ports or any number of nets into a bus object. Unless the -sort or -no_sort option is selected, the specified objects are sorted by reverse alphanumeric order of names before the bus is created. When using Design Vision to create a design schematic, bused nets are inferred from bused ports or pins in the design. Bused nets only appear as nets in a schematic if they are connected to a bused port. If the start bit and end bit for the bus are specified with the -start and -end options, the bus is created with these start and end indices. If only the start bit is specified, an upward-going bus starting at that start bit is built. If only the end bit is specified, an upward going bus ending at that end bit is built.

Syntax

```
create_bus <net_name_list>

[-start <start bit>]

[-end <end bit>]
```

Arguments

- `<net_name_list>`: Specifies a list of ports or nets to be put into a bus. If both ports and nets have the same names, the ports are put into the bus. This option is required.
- `<bus_name>`: Specifies the name of the bus. This name cannot be the same as any other bus or object of the same type. Port bus names must be different from the names of ports, and net bus names must be different from the names of nets. This option is required.
- `[ -start <start bit>]`: Specifies the start bit for the bus.
- `[ -end <end bit>]`: Specifies the end bit for the bus.

Examples

This example groups the A1, A2, and A3 ports into a bus named A. The ports must already exist. The order in which the ports appear in the bus is A3, A2, A1.

```
vc_static_shell> create_bus {A1 A2 A3} A
```

This example groups the existing D1, D2, and D3 ports into a bus named D and assigns it the index range 6-to-4. Port D1 is assigned index 6, port D2 to index 5, and port D3 to index 4.

```
vc_static_shell> create_bus {D1 D2 D3} D -start 6 -end 4
```

create_cell

Description

This command creates new leaf orhierarchical cells in the current design or its subdesigns based on the cell_list argument. New leaf cells are the instantiation of an existing design or library cell. New hierarchical cells are the instantiation of a new design. New cells are the instantiation of an existing design, a library cell, a logic 0 generator, or a logic 1 generator. The created cells are unplaced. To be viewed properly in the GUI, the cells must be placed either manually by using the set_cell_location command or automatically by using placement commands. To remove cells from the current design, use the remove_cell command. Although the reference_name argument accepts names in the format library/library_cell, the command might not instantiate the actual library cell from the specified library. The actual library cell to be used is determined by the current link library settings.

Syntax

```
create_cell <reference> [<reference>]

[-only_physical]

[-hierarchical]

[-logic <logic_value>]
```

Arguments

- `<reference>`: Specifies the design or library cell that new cells reference. You must specify the reference_name. Ports on the reference determine the name, number, and direction of pins on the new cell.

- `[<name_list>]`: Specifies the names of cells created in the current design. Each cell name must be unique within the current design.
- `[ -only_physical]`: Creates a new physical or physical-only cell using a reference from the physical library. The -only_physical option sets the is_physical_only attribute on the created physical-only cell. After you create a physical-only cell, you must assign a location to that cell. Unlike physical cells with logic functions, the tool does not assign a location to a physical-only cell during synthesis. To assign a location, use the set_cell_location command, Tcl commands, or a DEF file. By default, this option is off.
- `[ -hierarchical]`: Creates hierarchical cell instances and designs with the name given by cell_list if the reference_name is not specified. If you specify the reference_name, the cell_list must have a single element. The command creates the hierarchical cell instance with the name given by the single cell_list and also creates the design with the name given by reference_name.
- `[ -logic <logic_value>]`: Specifies that the new cell generates a logic 0 or logic 1 value. The logic value must be either 0 or 1. By using this option, the cell contains a single output pin. By default, this option is off.

Examples

The following example creates a cell named cell1 under the subdesign corresponding to the mid1 cell:

```
vc_static_shell> create_cell {mid1/cell1} my_lib/AND2
[Info] ADD_CELL: Creating cell 'cell1' in design 'mid1'.
```

The following example creates a cells with -hierarchical option

```
vc_static_shell> create_cell -hierarchical {H2 H3} my_lib/AND2
[Info] ADD_CELL: Creating cell 'H2' in design 'test'.
[Info] ADD_CELL: Creating cell 'H3' in design 'test'.
```

create_net

Description

The create_net command creates new net objects in the current design or its subdesign based on the net_list argument. The create_net command creates only scalar (single bit) nets. To bundle scalar nets into buses, use the create_bus command. Nets connect pins and ports in a design. When you create nets with create_net, they are not connected. To establish this connection, use the connect_net command. To remove nets from the current design, use the remove_net command.

Syntax

```
create_net <name_list>
```

Arguments

- `<name_list>`: Specifies the names of the created nets. If you specify a hierarchical net name, the net is created in the specified instance. The net name must be unique within the current design or the subdesign where it is created. If you use a hierarchical net name, the parent instance must be unique; it cannot be an instance of a multiply-instantiated design. This option is required.

Examples

The following example uses create_net to create net objects in the current design :

```
vc_static_shell> create_net {N1 N2 N3 N4}
[Info] ADD_NET: Creating net 'N1' in design 'test'.
[Info] ADD_NET: Creating net 'N2' in design 'test'.
[Info] ADD_NET: Creating net 'N3' in design 'test'.
[Info] ADD_NET: Creating net 'N4' in design 'test'.
```

The following example uses create_net to create net objects in the mid subdesign. Note that mid1 is an instance of the mid design. The mid design must be unique for the create_net command to succeed.

```
vc_static_shell> create_net {mid1/N1 mid1/N2 mid1/N3 mid1/N4}
[Info] ADD_NET: Creating net 'N1' in design 'mid1'.
[Info] ADD_NET: Creating net 'N2' in design 'mid1'.
[Info] ADD_NET: Creating net 'N3' in design 'mid1'.
[Info] ADD_NET: Creating net 'N4' in design 'mid1'.
```

create_port

Description

The create_port command creates new port objects in the current design or its subdesign. The create_port command creates only scalar or single bit ports. To bundle scalar ports into buses, use the create_bus command. Ports are the external connection points on a design. To connect ports to nets inside a design, use the connect_net command. To remove ports from the current design, use the remove_port command. Multicorner-Multimode Support This command has no dependency on scenario-specific information.

Syntax

```
create_port <name_list>
[-direction <dir>]
```

Arguments

- `<name_list>`: Specifies names of ports created in the current design. Each port name must be unique within the current design.
- `[ -direction <dir>]`: Specifies the signal flow of the created port. The possible values are in, out, or inout. The default is in.

Examples

The following example uses create_port to create ports in the current design

```
vc_static_shell> create_port -direction "in" {A1 A2 A3 A4}
[Info] ADD_PORT: Creating port 'A1' in design 'test'.
[Info] ADD_PORT: Creating port 'A2' in design 'test'.
[Info] ADD_PORT: Creating port 'A3' in design 'test'.
[Info] ADD_PORT: Creating port 'A4' in design 'test'.
```

The following example uses create_port to create ports in the U1 subdesign.

```
vc_static_shell> create_port -direction "in" {U1/B1 U1/B2 U1/B3
U1/B4}
[Info] ADD_PORT: Creating port 'B1' in design 'U1'.
[Info] ADD_PORT: Creating port 'B2' in design 'U1'.
[Info] ADD_PORT: Creating port 'B3' in design 'U1'.
[Info] ADD_PORT: Creating port 'B4' in design 'U1'.
```

define_user_attribute

Description

This command defines a new attribute. Use the list_attributes command to list the attributes that you have defined.

The definition for a user-defined attribute can be persistent if it has been set on a specified object and stored in the database.

Syntax

```
define_user_attribute <attr_name>

[-type <string | int | float | double | boolean>]

-classes <cell | clock | design | lib_cell | lib_pin | net |
net_word | pin | pin_word | port | port_word>

[-range_min <min>]

[-range_max <max>]

[-one_of <values>]
```

Arguments

- `<attr_name>`: Specifies the name of the attribute.
- `[ -type <string | int | float | double | boolean>]`: Specifies the data type of the attribute. Specifies the classes for the new user-defined attribute. Valid classes are design, port, cell, net, etc.
- `-classes <cell | clock | design | lib_cell | lib_pin | net | net_word | pin | pin_word | port | port_word>`: Specifies the type of the object on which the new attribute is created.
- `[ -range_min <min>]`: Specifies the minimum value for numeric ranges. This option is valid only when the data type of the attribute is int or double. Specifying a minimum constraint without a maximum constraint creates an attribute that accepts a value greater than or equal to min.
- `[ -range_max <max>]`: Specifies the maximum value for numeric ranges. This option is valid only when the data type of the attribute is int or double. Specifying a maximum constraint without a minimum

constraint creates an attribute that accepts a value less than or equal to max.

- `[ -one_of <values>]`: Provides a list of allowable strings. This option is valid only when the data type of the attribute is string.

Examples

The following example defines an attribute named attr_1, which has a data type of double with a minimum value of 2 and a maximum value of 3.2 and can be set on cells or nets.

```
prompt> define_user_attribute -class {cell net} -type double \\
-range_max 3.2 -range_min 2 attr_1
```

The following example defines an attribute named attr_2, which has a data type of string with valid values of true or false and can be set on nets.

```
prompt> define_user_attribute -class net -type string \\
-one_of {true false} attr_2
```

The following example shows how to list the attribute definitions using the list_attributes command:

```
prompt> list_attributes
```

disconnect_net

Description

The disconnect_net command breaks the connections between a net or a net instance and its pins or ports. The net, pins, and ports are not removed. This command accepts only scalar (single bit) nets, and not bused nets. To connect nets, use the connect_net command. To display the pins and ports connected to a net, use the all_connected command.

Syntax

```
disconnect_net <net_name>
```

Default Argument <net_name>

Specifies the net name or net instance name to disconnect. A net must exist in the current design.

- **Default Argument:** <object_name_list>

Specifies the pins and ports disconnected from the net. Only pins and ports existing in the current design are specified. Either object_list or -all must be specified.

Examples

The following examples disconnect nets using the disconnect_net command:

```
disconnect_net NET0 [get_ports A1]
Disconnecting net 'NET0' from port 'A1'
```

```
disconnect_net NET0 [get_pins U1/A]
Disconnecting net 'NET0' from pin 'U1/A'.
```

find

Description

Finds objects of specific object type

Syntax

```
find [<>]
```

```
[-hierarchy]
```

- **Default Argument:** [<>]

Specifies object type: Values: cell, design, lib_cell, lib_pin, net, pin, port

- **Default Argument:** [<>]

Specifies a list of names of design or library objects

Arguments

- `[ -hierarchy]`: Specifies to return all objects matching type and name_listwithin the current design hierarchy

get_cells

Description

This command creates a collection of cells that match the specified criteria. By default, the command creates the collection of cells from the current design, relative to the current instance.

If the command cannot find any cells that match the criteria and the current design is not linked, the design automatically links.

The command returns a collection if any cell matches the criteria. If no object matches the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_cells [<patterns>]

[-hierarchical]

[-quiet]

[-regexp]

[-nocase]

[-exact]

[-filter <expression>]

[-of_objects <objects>]
```

Arguments

- **Default Argument:** [<patterns>]: Creates a collection of cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

- `[ -hierarchical]`: Searches for cells level-by-level, relative to the current instance. The name of the object at a particular level must match the patterns. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder". By default, the search is not hierarchical.
- `[ -quiet]`: Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive. You can specify only one of these options.
- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each cell in the collection, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result. To see the list of cell attributes that you can use in the expression, use the list_attributes -application -class cell command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -of_objects <objects>]`: Creates a collection of cells connected to the specified objects. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command

uses the * (asterisk) as the default pattern. In addition, you cannot use the -hierarchical option with the -of_objects option.

Examples

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is only a display.

```
prompt> get_cells "o*" -filter "@ref_name == FD2"
{o_reg1 o_reg2 o_reg3 o_reg4}
```

The following example queries the cells connected to a collection of pins.

```
prompt> set pinsel [get_pins o*/CP]
{o_reg1/CP o_reg2/CP}

prompt> get_cells -of_objects $pinsel
{o_reg1 o_reg2}
```

The following example queries the cells connected to a collection of nets.

```
prompt> set netsel [get_nets tmp]
{tmp}

prompt> get_cells -of_objects $netsel
{b c}
```

get_designs

Description

The get_designs command creates a collection of designs from those currently loaded into the tool that match certain criteria. The command returns a collection if any designs match the patterns and pass the filter (if specified). If no objects match your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl regexp command. When using -regexp, take care in the way you quote the patterns and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the get_designs command at the command prompt, or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the get_designs result to a variable.

When issued from the command prompt, get_designs behaves as though query_objects has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the collection_result_display_limit variable.

The implicit query property of get_designs provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the query_objects options (for example, if you want to display the object class), use get_designs as an argument to query_objects.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_designs
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter <expression>]
```

Arguments

- `[ -hierarchical]`: Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. The use of this option does not force an auto link.
- `[ -quiet]`: Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- `[ -regexp]`: Uses the patterns argument as real regular expressions rather than simple wildcard patterns.
- `[ -nocase]`: Makes matches case-insensitive.

- `[ -exact]`: Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.
- `[ -filter <expression>]`: Filters the collection with expression. For any designs that match patterns, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

Examples

The following example queries the designs that begin with mpu. Although the output looks like a list, it is just a display. A complete listing of designs is available using the list_designs command.

```
prompt> get_designs mpu*
{mpu_0_0 mpu_0_1 mpu_1_0 mpu_1_1}
```

The following example shows that, given a collection of designs, you can remove those designs:

```
prompt> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

get_lib_cells

Description

This command creates a collection of library cells from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the link_library variable into memory the first time you run the get_libs, get_lib_cells, or get_lib_pins command.

If you use the -scenarios option, only libraries associated with the specified scenarios are included in the search.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_lib_cells <patterns>

[-quiet]

[-regexp]

[-exact]

[-nocase]

[-filter <expression>]

[-of_objects <objects>]
```

Arguments

- **Default Argument:** <patterns>: Creates a collection of library cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. The patterns and -of_objects arguments are mutually exclusive; you must specify one.
- `[ -quiet]`: Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is

recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.

- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each library cell in the collection, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result. To see the list of library cell attributes that you can use in the expression, use the list_attributes -application -class lib_cell command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -of_objects <objects>]`: Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. Each object is either a named library pin, a netlist cell, a library pin collection, or a netlist cell collection. The patterns and -of_objects arguments are mutually exclusive; you must specify one.

Examples

The following example queries all library cells that are in the misc_cmos library whose names begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*
{misc_cmos/AN2 misc_cmos/AN2P}
```

The following example shows one way to determine the library cell used by a particular cell instance:

```
prompt> get_lib_cells -of_objects [get_cells o_reg1]
{misc_cmos/FD2}
```

get_lib_pins

Description

This command creates a of library cell pins from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the link_library variable into memory the first time you run the get_libs, get_lib_cells, or get_lib_pins command.

The command returns a collection if any library cell pins match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_lib_pins <patterns>

[-quiet]

[-regexp]

[-exact]

[-nocase]

[-filter <expression>]

[-of_objects <objects>]
```

Arguments

- **Default Argument:** <patterns>: Creates a collection of library cell pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. The patterns and -of_objects arguments are mutually exclusive; you must specify one.

- `[ -quiet]`: Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each library cell pin in the collection, the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library cell pin is included in the result. To see the list of library cell pin attributes that you can use in the expression, use the list_attributes -application -class lib_pin command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -of_objects <objects>]`: Creates a collection of library cell pins referenced by the specified netlist Each object is either a named library cell, netlist pin, library cell collection, or a netlist pin collection. The patterns and -of_objects arguments are mutually exclusive; you must specify one.

Examples

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> get_lib_pins misc_cmos/AN2/*
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist:

```
prompt> get_lib_pins -of_objects o_reg1/Q
{misc_cmos/FD2/Q}
```

get_lib_timing_arcs

Description

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing.

Syntax

```
get_lib_timing_arcs

[-to <to_list>]

[-from <from_list>]

[-of_objects <cell list>]

[-filter <expression>]
```

Arguments

- `[ -to <to_list>]`: Specifies the "to" library pins, or ports. All backward library arcs from the specified library pins or ports are considered.
- `[ -from <from_list>]`: Specifies the "from" library pins, or ports. All forward library arcs from the specified library pins or ports are considered.
- `[ -of_objects <cell list>]`: Specifies library cells or timing arcs. If a library cell is specified, all library cell arcs of that cell are considered. If a timing arc collection is given in the object list, the corresponding library timing arc is considered.

- `[ -filter <expression>]`: Specifies the filter expression. A filter expression is a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of library arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value, by means of an operator.

Examples

The following examples show the usage of the get_lib_timing_arcs command:

```
vc_static_shell> get_lib_timing_arcs -of_objects [get_lib_cells
*/*]
{"tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/
INVD20BWP16P90LVT/I
tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/INVD20BWP16P90LVT/
ZN", "tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/
INVD4BWP16P90LVT/I
tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/INVD4BWP16P90LVT/
ZN"}
```

The following examples show the usage of the get_lib_timing_arcs command:

```
vc_static_shell> get_lib_timing_arcs -from
tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/INVD20BWP16P90LVT/
I -to tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/
INVD20BWP16P90LVT/ZN -filter "object_class == lib_timing_arc"
{"tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/
INVD20BWP16P90LVT/I
tcbn16ffllbwp16p90lvtffgnp0p6v125c_ccs_cchp0/INVD20BWP16P90LVT/
ZN"}
```

get_libs

Description

This command creates a collection of libraries from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the link_library variable into memory the first time you run the get_libs, get_lib_cells, or get_lib_pins command.

The command returns a collection if any library matches the criteria. If no library matches the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_libs [<patterns>]
[-quiet]
[-regexp]
[-exact]
[-nocase]
[-filter <expression>]
[-of_objects <objects>]
```

Arguments

- **Default Argument:** [<patterns>]: Creates a collection of libraries whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. The patterns and -of_objects arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.
- `[ -quiet]`: Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the

behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.

- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each library in the collection, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result. To see the list of library attributes that you can use in the expression, use the list_attributes -application -class lib command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -of_objects <objects>]`: Creates a collection of libraries that contain the specified objects. Each object is either a named library cell or a library cell collection. The patterns and -of_objects arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

Examples

The following example queries all loaded libraries. Use the list_libs command to get a complete listing of the libraries.

```
prompt> get_libs
{misc_cmos misc_cmos_io}
```

get_link

Description

This command returns the designs built with reason code using Simon/ VNR.

Syntax

```
get_link
```

```
-reasons <reason_list>
```

Arguments

- `-reasons <reason_list>`: Use this option to specify the reason list. The values are AutoBB, AutoGB, DirtyData, Empty, PhysicalCell, SimDirection, SimMatch, SimPort, SimWidth, Synthesis, Unresolved, UserBB, and UserGB.

get_nets

Description

This command creates a collection of nets in the current design relative to the current instance that match the specified criteria.

The command returns a collection if any nets match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_nets <patterns>
```

```
[-regexp]
```

```
[-exact]
[-nocase]
[-filter <expression>]
[-hierarchical]
[-filter <expression>]
[-quiet]
[-top_net_of_hierarchical_group]
[-segments]
[-of_objects <objects>]
```

Arguments

- **Default Argument:** <patterns>: Creates a collection of nets whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.

- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each net in the collection, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result. To see the list of net attributes that you can use in the expression, use the list_attributes -application -class net command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -hierarchical]`: Searches for nets level-by-level relative to the current instance. The name of the object at a particular level must match the patterns. For example, if there is a net named block1/muxsel, a hierarchical search finds it using muxsel.
- `[ -filter <expression>]`: Filters the collection with the value of the expression argument. For any nets that match the specified criteria, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.
- `[ -quiet]`: Suppresses messages if no objects match. Syntax error messages are not suppressed.
- `[ -top_net_of_hierarchical_group]`: Keeps only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved in the collection. In the case of multiple nets at the same level, the first net specified is kept. To get the top hierarchical net connected to a single net, need to use this option in combination with the -segments option.
- `[ -segments]`: Modifies the initial search that matches the patterns or -of_object argument to include all global segments for the matching nets. Global net segments are all the net segments that are physically connected across all hierarchical boundaries. This option is best used with a single net. When you use the -segments option with the -top_net_of_hierarchical_group option, you can isolate the highest net segment of a physical net.
- `[ -of_objects <objects>]`: Creates a collection of nets connected to the specified objects. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern. You cannot use the -hierarchical option with the -of_objects option.

Examples

The following example queries the nets that begin with NET in a block named block1. Although the output looks like a list, it is a display.

```
prompt> get_nets block1/NET*
{block1/NET1QNX block1/NET2QNX}
```

The following example queries the nets connected to a collection of pins.

```
prompt> current_instance block1
block1
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{o_reg1/QN o_reg2/QN}
prompt> get_nets -of_objects $pinsel
{NET1QNX NET2QNX}
```

The following example queries the nets connected to a collection of cells.

```
prompt> current_instance block1
block1
prompt> set cellsel [get_cells {o_reg1 o_reg2}]
{o_reg1 o_reg2}
prompt> get_nets -of_objects $cellsel
{NET1QX NET1QNX NET1DX NET2QX NET2QNX NET2DX}
```

The following examples use the design shown below to show the behavior of the get_nets command with various options. There is a buffer instance named buffer in the L3 instance.

```
+-------------------------------------+
| L1 |
| +-----------------------+ |
| | L2 | |
| | +-------------+ | |
| | | L3 | | |
| | | | | |
net1 | net2 | net3 | net4 | | |
--------+------+------+------> | | |
| | | | | |
| | +-------------+ | |
| | | |
| +-----------------------+ |
| |
+-------------------------------------+
```

When you use the -of_objects option by itself, the command gets only the net that connects to a pin directly.

```
prompt> get_nets -of_objects L1/L2/L3/buffer/A
{L1/L2/L3/net_4}
```

When you also use the -segments option, the command gets all the hierarchical nets that connect together through the hierarchical pins.

```
prompt> get_nets -of_objects L1/L2/L3/buffer/A -segments
{L1/L2/L3/net_4 L1/L2/net_3 L1/net_2 net_1}
```

When you use the -segments and -top_net_of_hierarchical_group option together, the command returns only the net in the topmost hierarchical net group.

```
prompt> get_nets -of_objects L1/L2/L3/buffer/A -segments \
-top_net_of_hierarchical_group
{net_1}
```

The following examples use the patterns argument to select the nets.

```
prompt> get_nets net*4 -hierarchical
{L1/L2/L3/net_4}

prompt> get_nets net*4 -hierarchical -segments
{L1/L2/net_3 net_1 L1/L2/L3/net_4 L1/net_2}

prompt> get_net net*4 -hierarchical -segments -filter
name==net_2
{L1/net_2}
```

get_object_name

Description

This command returns a list of names of the objects in a collection.

Syntax

```
get_object_name <collection>
```

Default Argument <collection>

Specifies the name of the collection that contains objects whose names are requested.

Examples

The following example returns the name top as the object contained in the collection returned by the current_design command.

```
prompt> get_object_name [current_design]
Current design is 'top'.
top
```

get_pins

Description

This command creates a collection of pins in the current design relative to the current instance that match the specified criteria.

arguments do not match any objects and the current design is not linked, the design automatically links.

The command returns a collection if any pins match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

Syntax

```
get_pins <patterns>

[-hierarchical]

[-quiet]

[-regexp]

[-exact]

[-nocase]
```

```
[-filter <expression>]
[-of_objects <objects>]
[-leaf]
[-all]
```

Arguments

- **Default Argument:** <patterns>: Creates a collection of pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.
- `[ -hierarchical]`: Searches for pins level-by-level, relative to the current instance. The name of the object at a particular level must match the patterns. The search is similar to that of the UNIX find command. For example, if there is a pin named block1/adder/D[0], a hierarchical search finds it by using adder/D[0]. The -hierarchical option is mutually exclusive with use the -of_objects option; you can use only one.
- `[ -quiet]`: Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.

- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -nocase]`: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- `[ -filter <expression>]`: Filters the collection with the specified expression. For each pin in the collection, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result. To see the list of pin attributes that you can use in the expression, use the list_attributes -application -class pin command. For more information about how to use the -filter option, see the filter_collection man page.
- `[ -of_objects <objects>]`: Creates a collection of pins connected to the specified objects. By default, the command considers only pins connected to the specified nets at the same hierarchical level. To consider only pins connected to leaf cells on the specified nets, use the -leaf option. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern. You cannot use the -hierarchical option with the -of_objects option.
- `[ -leaf]`: Includes only those pins that are on leaf cells connected to the nets specified in the -of_objects option. The tool crosses hierarchical boundaries to find pins on leaf cells. You can use this option only if you also use the -of_objects option.
- `[ -all]`: Includes power and ground pins.

Examples

The following example queries the CP pins of cells that begin with o. Although the output looks like a list, it is only a display.

```
prompt> get_pins o*/CP
{o_reg1/CP o_reg2/CP o_reg3/CP o_reg4/CP}
```

The following example queries the pins connected to a collection of cells:

```
prompt> set csel [get_cells o_reg1]
{o_reg1}

prompt> get_pins -of_objects $csel
{o_reg1/D o_reg1/CP o_reg1/CD o_reg1/Q o_reg1/QN}
```

The following example shows the difference between getting the local pins of a net and the leaf pins of net. In this example, NET1 is connected to the i2/aP and reg1/QN. Cell i2 is hierarchical. Within cell i2, port a is connected to U1/A and U2/A.

```
prompt> get_pins -of_objects [get_nets NET1]
{i2/a reg1/QN}

prompt> get_pins -leaf -of_objects [get_nets NET1]
{i2/U1/A i2/U2/A reg1/QN}
```

The following example shows how to create a clock using a collection of pins:

```
prompt> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

get_ports

Description

This command creates a collection of ports by selecting ports from the current design that match the specified criteria.

The command returns a collection if any ports match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as query_objects. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the query_objects command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the collection_result_display_limit variable.

For information about collections and the querying of objects, see the collections man page.

In addition, see the man pages for the all_inputs and all_outputs commands, which also create collections of ports.

Syntax

```
get_ports <patterns>
```

```
[-quiet]
[-regexp]
[-exact]
[-nocase]
[-filter <expression>]
[-hierarchical]
[-of_objects <objects>]
```

Arguments

- **Default Argument:** <patterns>: Creates a collection of ports whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page. Pattern matching is case sensitive unless you use the -nocase option. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.
- `[ -quiet]`: Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- `[ -regexp]`: Views the patterns argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl regexp command. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; you can use only one.
- `[ -exact]`: Considers wildcards to be plain characters, and does not interpret their meaning as wildcards. The -regexp and -exact options are mutually exclusive; you can use only one.

- [-nocase]: Makes matches case-insensitive, both for the patterns argument and for the ==, =~, and !~ filter operators.
- [-filter <expression>]: Filters the collection with the specified expression. For each port in the collection, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the port is included in the result. To see the list of port attributes that you can use in the expression, use the list_attributes -application -class port command. For more information about how to use the -filter option, see the filter_collection man page.
- [-hierarchical]: Searches for ports level-by-level, relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to that of the UNIX find command. For example, if there is a port named block1/adder/D[0], a hierarchical search finds it by using adder/D[0]. The -hierarchical option is mutually exclusive with use the -of_objects option; you can use only one.
- [-of_objects <objects>]: Creates a collection of ports connected to the specified objects. Each object can be a net, terminal, bound, or via region. arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

Examples

The following example queries all input ports beginning with "mode". Although the output looks like a list, it is only a display.

```
prompt> get_ports mode* -filter {@port_direction == in}
{mode[0] mode[1] mode[2]}
```

The following example sets the driving cell for ports beginning with

```
prompt> set_driving_cell -lib_cell FD2 -library my_lib \\
[get_ports in*]
```

The following example reports ports connected to nets that match the pattern "bidir*".

```
prompt> report_port [get_ports -of_objects [get_nets bidir*]]
```

The following example get the ports connected to terminals that match the pattern "CC*".

```
prompt> get_ports -of_objects [get_terminals CC*]]
```

```
{CC CCEN}
```

The following example shows you can get bus ports by their base name. A[0], A[1], and A[2] are bus ports with the base name A; A[3] is not a bus port.

```
prompt> get_ports A
{A[0] A[1] A[2]}

prompt> get_ports A[3]
{A[3]}
```

get_timing_arcs

Description

This command returns the collection of timing arc objects.

Syntax

```
get_timing_arcs
-from <from>
-to <to>
-of_objects obj_list
-filter <expression>
-onlyseq
-onlycombo
```

Arguments

- `-from <from>`: Use this option to specify the source objects in timing arc objects.
- `-to <to>`: Use this option to specify the destination objects in timing arc objects.
- `-of_objects obj_list`: Use this option to specify the object list (net/cell).

- `-filter <expression>`: Use this option to specify the filters in those timing arcs of which attributes matches with expression.
- `-onlyseq`: Use this option to get only the sequential timing arcs.
- `-onlycombo`: Use this option to get only the combinational timing arcs.

Examples

The following examples show how to use get_timing_arcs command

```
vc_static_shell> get_timing_arc -from INST/IN -to INST/PAD
{"INST/IN --> INST/PAD"}

vc_static_shell> get_timing_arc -of [get_cells INST]
{"INST/IN1 --> INST/PAD", "INST/IN --> INST/PAD", "INST/NOE -->
INST/PAD", "INST/PAD --> INST/OUT"}

vc_static_shell> get_timing_arc -of [get_cells INST] -onlyseq
{"INST/IN1 --> INST/PAD"}
```

insert_buffer

Description

This command adds a buffer at one specified net or pin. A library cell with a single input and single output can be used as the buffer or inverter, as long as output has the same or inverted logic function of the input.

Syntax

```
insert_buffer <object_name>

[-no_of_cells < number of cells>]

[-inverter_pair]

-new_net_cells <new net cells>

-new_cell_names <new cell names>
```

Arguments

- **Default Argument:** <object_name>: Net or Pin which need to be buffered.
- **Default Argument:** <buffer lib cell>: Reference buffer lib cell to be instantiated.
- `[ -no_of_cells < number of cells>]`: Number of level(s) of buffers to be introduced.
- `[ -inverter_pair]`: Insert a pair of inverter cells for each buffer.
- `-new_net_cells <new net cells>`: Specifies the name of the new net.
- `-new_cell_names <new cell names>`: Specifies the name of the new buffer.

Examples

The following example creates a buffer named cell1 under the sub design corresponding to the mid1 cell:

```
vc_static_shell> insert_buffer -new_net_names n1 -
new_cell_names buf1 i tiny/BUF
[Info] NET_BUFFERING: Buffering net 'i' in design 'top'.
[Info] NET_DISCONNECTED: Disconnecting net 'i' to object 'I' in
design 'top'
[Info] ADD_NET: Creating net 'n1' in design 'top'.
[Info] ADD_CELL: Creating cell 'buf1' in design 'top'.
[Info] NET_CONNECTED: Connecting net 'i' to object 'I' in
design 'top'
[Info] NET_CONNECTED: Connecting net 'n1' to object 'Z' in
design 'top'
[Info] NET_CONNECTED: Connecting net 'n1' to object 'I' in
design 'top'
```

list_designs

Description

list_designs command lists designs that have been loaded.

Syntax

```
list_designs [<design_list>]
```

Arguments

- **Default Argument** [<design_list>]: Use this option to specify the list of designs

Examples

The following examples show the usage of the list_designs command:

```
vc_static_shell> list_designs
ALU BLENDER CLOCK_GEN CONTEXT_MEM CONTEXT_MEM_DW01_inc_6_0
CONTEXT_MEM_DW01_inc_6_1 CONTROL DATA_PATH INSTRN_LAT ORCA (*)
ORCA_TOP PARSER PCI_CORE PCI_RFIFO PCI_WFIFO
PCI_W_MUX PRGRM_CNT_TOP REG_FILE RESET_BLOCK RISC_CORE
1
```

The following example shows the usage of the list_designs command with a query pattern

```
vc_static_shell> list_designs ALU
ALU
1
```

list_instance

Description

list_instance command lists the instances in the current design or current instance.

Syntax

```
list_instance [<instance_list>]

[-max_levels <num_levels>]

[-hierarchy]

[-full]
```

Arguments

- **Default Argument:** [<instance_list>]: Use this option to specify the list of instances. By default, all instances in the current design or current instance are listed.
- `[ -max_levels <num_levels>]`: Specifies a limit to the number of levels of hierarchy that are listed.
- `[ -hierarchy]`: Lists all levels of instance hierarchy. By default, only the current level of hierarchy is listed.
- `[ -full]`: Displays the full hierarchy. By default, if there is a submodule in multiple locations in a hierarchy, its components are listed only once with ellipses (...) indicating the contents of a previously displayed module.

Examples

The following examples show the usage of the list_instance command:

```
vc_static_shell> list_instance
I_BNOT__SVAC_1_abort (BITWISE_NOT) I_BNOT__SVAC_1_disable_iff
(BITWISE_NOT) I_BNOT__SVAC_1_disable_iff_0 (BITWISE_NOT)
I_BNOT__SVAC_1_disable_iff_1 (BITWISE_NOT) _SVAC_1_ended_reg
(SEQ_FF) I_OR_N_36 (BITWISE_OR)
I_AND_N_1 (BITWISE_AND) I_BNOT_N_0 (BITWISE_NOT)
I_BUF__sva_topbit_0 (CONNECT)
```

list_libs

Description

list_libs command reports all the libs loaded and the details of these libs.

Syntax

```
list_libs [<patterns>]
```

Arguments

- **Default Argument** [<patterns>]: Use this option to specify the list of libs

Examples

The following examples show the usage of the list_libs command:

```
vc_static_shell> list_libs
-------------------------------------------------
Library File Path
------- ---- ----
ATL25_25_cell_wcmil ATL25_25_cell_wcmil.db /remote/dtdata1/
testdata/libraries/syn/
lsi_10k lsi_10k.db /remote/dtdata1/testdata/libraries/syn/
```

The following examples show the usage of the list_libs command with a query pattern

```
vc_static_shell> list_libs ATL25_25_cell_wcmil
-------------------------------------------------
Library File Path
------- ---- ----
ATL25_25_cell_wcmil ATL25_25_cell_wcmil.db /remote/dtdata1/
testdata/libraries/syn/
```

remove_attribute

Description

This command removes the specified attribute from the specified objects. For a complete list of attributes, see the attributes man page.

A returned empty string indicates that no object has been removed.

Syntax

```
remove_attribute <object_list>
```

Arguments

- **Default Argument** <object_list>: Specifies a list of objects from which the attribute is to be removed.

remove_buffer

Description

This command removes buffer cells and the specified connected net.

Syntax

```
remove_buffer <cell name>
[-from <start point>]
[-net]
[-to <pins>]
[-level <level>]
```

Arguments

- **Default Argument:** <cell name>: Specifies the buffer to be removed.
- `[ -from <start point>]`: Starting Driver Pin to begin buffer deletion.
- `[ -net]`: Net from which buffer need to be deleted.
- `[ -to <pins>]`: Load pins where to end buffer deletion.
- `[ -level <level>]`: Number of level(s) of buffers to be introduced.

Examples

The following example removes a buffer named buf1 in design top:

```
vc_static_shell> remove_buffer buf1
[Info] NET_DISCONNECTED: Disconnecting net 'n1' to object 'Z'
in design 'top'
[Info] NET_DISCONNECTED: Disconnecting net 'i' to object 'I' in
design 'top'
[Info] NET_DISCONNECTED: Disconnecting net 'n1' to object 'I'
in design 'top'
[Info] NET_CONNECTED: Connecting net 'i' to object 'I' in
design 'top'
[Info] REMOVE_CELL: Removing cell 'buf1' in design 'top'.
[Info] REMOVE_NET: Removing net 'n1' in design 'top'.
```

remove_bus

Description

This command removes port or net buses from a design. If port and net buses have the same name, both port buses and net buses are removed. Individual members of buses remain. The bus can be from the current design or one of its subdesigns. To delete a bus on a subdesign, it must be unique. Multicorner-Multimode Support This command has no dependency on scenario-specific information.

Syntax

```
remove_bus <bus_name_list>
```

Arguments

- **Default Argument** <bus_name_list>: Specifies a list of port or net buses to remove.

Examples

The following example removes the specified port buses:

```
vc_static_shell> remove_bus {AB AC}
```

In the following example, all buses with names that end in s are removed. If port and net buses have the same name, both buses are removed.

```
vc_static_shell> remove_bus *s
```

remove_cell

Description

The remove_cell command removes cells or cell instances from the current design. The command also removes pins owned by the specified cells. The design or library cell to which the cell refers is not removed. If a cell instance is used, the parent design must be unique. To create cells, use the create_cell command.

Syntax

```
remove_cell <name_list>

[-all]
```

Arguments

- **Default Argument:** <name_list>: Specifies a list of cells to be removed from the current design. Each cell name must exist in the current design.
- `[ -all]`: Removes all cells in the current design.

Examples

The following example uses remove_cell to remove the specified cells in the current design:

```
vc_static_shell> remove_cell {U1 U2}
Removing cell 'U1' in design 'example'.
Removing cell 'U2' in design 'example'.
```

The following example removes all cells remaining in the current design:

```
vc_static_shell> remove_cell -all
Removing cell 'U3' in design 'example'.
Removing cell 'U4' in design 'example'.
Removing cell 'U5' in design 'example'.
Removing cell 'U6' in design 'example'.
Removing cell 'U7' in design 'example'.
Removing cell 'U8' in design 'example'.
```

remove_net

Description

This command removes nets or net instances from the current design. Net connections to pins or ports are disconnected. You cannot remove bused nets with the remove_net command. Use the remove_bus command to remove bused nets. Scalar (single bit) nets that are components of a bused net cannot be removed. You must remove bused nets first. To create nets, use the create_net command. Multicorner-Multimode Support This command has no dependency on scenario-specific information.

Syntax

```
remove_net <name_list>
```

```
[-only_physical]
```

```
[-all]
```

Arguments

- **Default Argument:** <name_list>: Specifies a list of nets to remove from the current design. Each net name must exist in the current design. You must specify either net_list or -all
- `[ -only_physical]`: Removes only physical nets in the current design. The physical nets are those nets that do not connect to a logical netlist.
- `[ -all]`: Removes all nets in the current design. You must specify either -all or net_list.

Examples

The following example removes the nets in the current design:

```
vc_static_shell> get_nets *
{"w2", "out", "w", "out1", "w3", "clk1", "w10", "clk2", "sel",
"in"}
vc_static_shell> remove_net w
[Info] REMOVE_NET: Removing net 'w' in design 'test'.
1
```

The following example removes the physical nets:

```
vc_static_shell>remove_net -only_physical {physnet1 physnet2}
```

remove_port

Description

This command removes ports from the current design or its subdesign. Bused ports cannot be removed with remove_port; use remove_bus to remove bused ports. Also, scalar (single bit) ports that are components of a bused port cannot be removed. In this case, you must first remove the bused port. To create ports, use the create_port command. Multicorner-Multimode Support This command has no dependency on scenario-specific information.

Syntax

```
remove_port <name_list>
```

Arguments

- **Default Argument** <name_list>: Specifies a list of ports to be removed from the current design. Each port name must exist in the current design.

Examples

The following example uses remove_port to remove ports from the current design

```
vc_static_shell> remove_port "B*"
Removing port 'B1' in design 'my_design'.
Removing port 'B2' in design 'my_design'.
```

report_cell

Description

Reports cell information of the current design

Syntax

```
report_cell <cell_list>
```

```
[-no_split]
```

Arguments

- **Default Argument:** <cell_list>: Use this option to specify the list of cells
- `[ -no_split]`: Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

Examples

The following examples show the usage of the report_cell command:

```
vc_static_shell> report_cell
****************************************
```

```
Report : cell
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
****************************************
Attributes:
b - black-box (unknown)
h - hierarchical
n - noncombinational
Cell Reference Library Area Attributes
---------------------------------------------------------------
-----------------
u1 block 21.000000 h
u2 block 21.000000 h
u3 block 21.000000 h
---------------------------------------------------------------
-----------------
Total 3 Cells 63.000000

vc_static_shell> report_cell -no_split
****************************************
Report : cell
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
****************************************

Attributes:
b - black-box (unknown)
h - hierarchical
n - noncombinational
Cell Reference Library Area Attributes
---------------------------------------------------------------
-----------------
u12345678910123456789 block 21.000000 h
u2 block 21.000000 h
u3 block 21.000000 h
---------------------------------------------------------------
-----------------
Total 3 Cells 63.000000
```

report_link

Description

Report status of the design build using Simon/VNR (stub)

Syntax

```
report_link
[-sim_match]
```

- `[ -sim_match]`: Includes library cells where a simulation model is given

report_net

Description

Reports net information of the current design

Syntax

```
report_net <net_list>
[-no_split]
```

Arguments

- **Default Argument:** <net_list>: Use this option to specify the list of nets
- `[ -no_split]`: Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

Examples

The following examples show the usage of the report_net command:

```
vc_static_shell> report_net
****************************************
```

```
Report : net
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
****************************************
Net Fanout Fanin Resistance Pins Attributes
---------------------------------------------------------------
-----------------
Q 1 1 0.000000 2
S 1 1 0.000000 2
PQRSTUVWXYZABCDEFGHIJKLMNOPGRS
1 1 0.000000 2
```

```
vc_static_shell> report_net -no_split
****************************************
Report : net
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
****************************************
Net Fanout Fanin Resistance Pins Attributes
---------------------------------------------------------------
-----------------
Q 1 1 0.000000 2
S 1 1 0.000000 2
PQRSTUVWXYZABCDEFGHIJKLMNOPGRS1 1 0.000000 2
```

report_port

Description

Displays port information within the design

Syntax

```
report_port <port_list>

[-no_split]
```

Arguments

- **Default Argument:** <port_list>: Use this option to specify the list of ports
- `[ -no_split]`: Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

Examples

The following examples show the usage of the report_net command:

```
vc_static_shell> report_port
****************************************
Report : port
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
Port Dir Pin Cap(min/max) Wire Cap(min/max)
---------------------------------------------------------------
-----
A in 0.0000/0.0000 0.0000/0.0000
B out 0.0000/0.0000 0.0000/0.0000
C in 0.0000/0.0000 0.0000/0.0000
D out 0.0000/0.0000 0.0000/0.0000
```

```
vc_static_shell> report_port -no_split
****************************************
Report : port
Design : top
Version: P-2020.03-Alpha
Date : Wed Mar 20 20:16:37 2019
****************************************
Port Dir Pin Cap(min/max) Wire Cap(min/max)
---------------------------------------------------------------
-------
ABCDEFGHIJKLMNOPQRSTUVWXYZ in 0.0000/0.0000 0.0000/0.0000
B out 0.0000/0.0000 0.0000/0.0000
C in 0.0000/0.0000 0.0000/0.0000
D out 0.0000/0.0000 0.0000/0.0000
```

set_always_on_cell

Description

The set_always_on_cell command sets all cells in the library whose name matches the specified cell name as always-on cells. Only buffer or inverter cells in the library can be specified as always-on cells.

Syntax

```
set_always_on_cell
```

Examples

The following example uses the set_always_on_cell command to set the library cell named INV_AO as an always-on cell:

```
prompt> set_always_on_cell INV_AO
```

set_attribute

Description

This command sets the value of an attribute on an object. For a complete list of attributes, see the attributes man page.

This command returns a collection of objects that have the specified attribute value set. If the attribute is not set on any objects, the command returns an empty string.

Syntax

```
set_attribute <objects>

[-type <boolean | integer | float | string>]

[-quiet]
```

Arguments

- **Default Argument**: <objects>: Specifies the objects on which the attribute is to be set.

- `[ -type <boolean | integer | float | string>]`: Specifies the data type of the attribute_value argument. This argument is required when creating new attributes; otherwise, it is optional. If the attribute data type is not specified, the tool uses either the specified attribute_value argument or the data type of the existing attribute.
- `[ -quiet]`: Turns off the warning message that would otherwise be issued if the attribute or objects are not found.

Examples

The following example defines an integer cell attribute named X, and then sets this attribute to 30 on all cells in this level of the hierarchy:

```
prompt> define_user_attribute -type int -classes cell X
cell
prompt> set_attribute [get_cells *] X 30
{U1}
```

set_get_command_message_limit

Description

This command sets the limit number of get commands failures.

Syntax

```
set_get_command_message_limit <command name>

-no_limit

-number <number>
```

Arguments

- **Default Argument:** <command name>: Use this option to specify the Get command to be set. The values are all, get_cells, get_designs, get_lib_cells, get_lib_pins, get_nets, get_pins and get_ports.
- `-no_limit`: Use this option to remove limit.
- `-number <number>`: Use this option to specify the limit number of failures. The value is >= 1.

set_isolation_cell

Description

The set_isolation_cell command sets the specified library cells as isolation cells. You can also specify the data and enable pin details for the isolation cell.

Syntax

```
set_isolation_cell
[-data_pin <data_pin_name>]
[-enable_pin <enable_pin_name>]
```

- `[ -data_pin <data_pin_name>]`: Specifies the name of the data pin of the isolation cell.
- `[ -enable_pin <enable_pin_name>]`: Specifies the name of the enable pin of the isolation cell.

Examples

In the following example, all cells in the library whose name starts with ISO are set as isolation cells. The EN pin of those library cells is considered the enable pin of the isolation cell.

```
prompt> set_isolation_cell ISO* -enable_pin EN
```

set_level_shifter_cell

Description

The set_level_shifter_cell command specifies the properties of the level-shifter cells.

Syntax

```
set_level_shifter_cell
[-cell_type <cell_type>]
[-cell_input_voltage_range <{lower_range upper_range}>]
```

```
[-cell_output_voltage_range <{lower_range upper_range}>]
[-std_cell_main_rail_pg_pin <pg_pin_name>]
[-data_pin <data_pin_name>]
[-input_voltage_range <{lower_range upper_range}>]
[-output_voltage_range <{lower_range upper_range}>]
[-input_signal_level <signal_level>]
[-enable_pin <enable_pin_name>]
[-enable_signal_level <signal_level>]
[-output_signal_level <signal_level>]
```

Arguments

- `[ -cell_type <cell_type>]`: Specifies the cell type of the level-shifter cell.
- `[ -cell_input_voltage_range <{lower_range upper_range}>]`: Specifies the cell input voltage range of the level-shifter cell.
- `[ -cell_output_voltage_range <{lower_range upper_range}>]`: Specifies the cell output voltage range of the level-shifter cell.
- `[ -std_cell_main_rail_pg_pin <pg_pin_name>]`: Specifies the standard cell P/G pin of the level-shifter cell.
- `[ -data_pin <data_pin_name>]`: Specifies the data pint of the level-shifter cell.
- `[ -input_voltage_range <{lower_range upper_range}>]`: Specifies the input voltage range of the level-shifter cell.
- `[ -output_voltage_range <{lower_range upper_range}>]`: Specifies the output voltage range of the level-shifter cell.
- `[ -input_signal_level <signal_level>]`: Specifies the input signal level of the level-shifter cell.
- `[ -enable_pin <enable_pin_name>]`: Specifies the enable pin of the level-shifter cell.
- `[ -enable_signal_level <signal_level>]`: Specifies the enable signal level of the level-shifter cell.

- [-output_signal_level <signal_level>]: Specifies the output signal level of the level-shifter cell.

Examples

In the following example, library cells whose names start with LVLLH are set as level-shifter cells of type low to high.

```
prompt> set_level_shifter_cell LVLLH* -cell_type LH -
std_cell_main_rail_pg_pin VDD \\
-enable_pin EN -input_signal_level VDDL -enable_signal_level
VDDH \\
-output_signal_level VDDH -cell_input_voltage_range {0.7 1.4}
\\
-cell_output_voltage_range {0.7 1.4}
```

set_pg_pin_model

Description

The set_pg_pin_model command defines the power and ground pins for a library cell. If the power or ground pin already exists, the new specification overrides the old one. Otherwise, new power and ground pins are created for the library cell.

Syntax

```
set_pg_pin_model
[-pg_pin_name <pin_name>]
[-pg_voltage_name <voltage_names>]
[-pg_pin_type <pin_types>]
[-pg_pin_direction <pin_directions>]
[-pg_physical_connection <physical_connections>]
[-pg_related_bias_pin <related_bias_pins>]
```

Arguments

- `[ -pg_pin_name <pin_name>]`: Specifies the names of the power and ground pins of the library cell.
- `[ -pg_voltage_name <voltage_names>]`: Specifies the voltage name of the corresponding power or ground pin of the library cell. The voltage names are defined in the library voltage map. There must be a one-to-one correspondence between the voltage names specified in this option and the pins specified in the -pg_pin_name option.
- `[ -pg_pin_type <pin_types>]`: Specifies the pin type for the each pin specified in the -pg_pin_name option. Valid values are primary_power, primary_ground, backup_power, backup_ground, internal_power, internal_ground, pwell, nwell, deeppwell, and deepnwell. There must be a one-to-one correspondence between the pin types specified in this option and the pins specified in the -pg_pin_name option.
- `[ -pg_pin_direction <pin_directions>]`: Specifies the direction for each pin specified in the -pg_pin_name option. Valid values are input, output, inout, and internal. There must be a one-to-one correspondence between the pin directions specified in this option and the pins specified in the -pg_pin_name option.
- `[ -pg_physical_connection <physical_connections>]`: Specifies the type of physical connection used for each pin specified in the -pg_pin_name option. Valid values are device_layer and routing_pin. There must be a one-to-one correspondence between the connection types specified in this option and the pins specified in the -pg_pin_name option.
- `[ -pg_related_bias_pin <related_bias_pins>]`: Specifies the related bias pin for each pin specified in the -pg_pin_name option. There must be a one-to-one correspondence between the related bias pins specified in this option and the pins specified in the -pg_pin_name option.

Examples

The following example defines the power and ground pins for the library cells whose name starts with LVLH.

```
prompt> set_pg_pin_model LVLH* -pg_pin_name {VDD VSS} \\
-pg_voltage_name {VDDL VSS} \\
-pg_pin_type {primary_power primary_ground}
```

set_pin_model

Description

The set_pin_model command defines the related power pin, related ground pin, and related bias pin for the specified library cell pins. If the related pin is already defined, the new specification overrides the old one.

Syntax

```
set_pin_model
[-pins <pin_names>]
[-related_power_pin <pin_names>]
[-related_ground_pin <pin_names>]
[-related_bias_pin <pin_names>]
[-power_down_function <functions>]
```

Arguments

- `[ -pins <pin_names>]`: Specifies the names of the pins for which you are defining the related power and ground pins.
- `[ -related_power_pin <pin_names>]`: Specifies the related power pin for each pin specified in the -pins option. There must be a one-to-one correspondence between the power pins specified in this option and the pins specified in the -pins option.
- `[ -related_ground_pin <pin_names>]`: Specifies the related ground pin for each pin specified in the -pins option. There must be a one-to-one correspondence between the ground pins specified in this option and the pins specified in the -pins option.
- `[ -related_bias_pin <pin_names>]`: Specifies the related bias pin for each pin specified in the -pins option. There must be a one-to-one correspondence between the bias pins specified in this option and the pins specified in the -pins option.
- `[ -power_down_function <functions>]`: Specifies the power-down function for each pin specified in the -pins option. There must be a one-to-one correspondence between the function specified in this option and the pins specified in the -pins option.

Examples

The following example defines the pin model for the RETN and RETNOUT

```
prompt> set_pin_model *DRFF* -pins {RETN RETNOUT} \\
-related_power_pin VDDG \\
-related_ground_pin VSSG
```

The following example defines the pin model for the SLEEPOUT pin of the library cells that match the pattern HEAD*.

```
prompt> set_pin_model HEAD* -pins {SLEEPOUT} \\
-power_down_function "!VDDG + VSS"
```

set_power_switch_cell

Description

The set_power_switch_cell command sets the specified library cells as power-switch cells.

The switch_cell_type attribute must be either coarse_grain or fine_grain .

The switch_function attribute must be defined to identify the control logic of its switch pins.

It can be defined at either controlled output power or ground pins which is virtual VDD or vitural VSS pg_pin.

The cell must have at least one switch pin. And switch pin cannot be output pin.

The cell must have at least one controlled power or ground pin, and one regular power and ground pin.

The output power pin must have pg_function Boolean expression containing input power or ground pin.

The pg_function attribute must contain only the input power or ground pin.

Syntax

```
set_power_switch_cell

[-cell_type <course_grain | fine_grain>]

[-switch_pin <pin_name>]
```

```
[-pg_pin <{pin_name switch_function pg_function}>]
```

Arguments

- `[ -cell_type <course_grain | fine_grain>]`: Specifies the type of the power-switch cell. The type can be either coarse_grain or fine_grain.
- `[ -switch_pin <pin_name>]`: Specifies the switch pin of the power-switch cell.
- `[ -pg_pin <{pin_name switch_function pg_function}>]`: Specifies the power or ground pin of the power-switch cell.

Examples

In the following example, the library cells whose names starts with FOOT are set as course-grain power-switch cells.

```
prompt> set_power_switch_cell FOOT* -cell_type coarse_grain \\
-switch_pin SLEEPN -pg_pin {VSS !SLEEPN VSS}
```

set_retention_cell

Description

The set_retention_cell command sets the specified library cells as retention cells. You can also specify the retention pin and retention cell type for the retention cell.

Syntax

```
set_retention_cell

[-cell_type <retention_type>]

[-retention_pin <{pin_name pin_type disable_value}>]
```

Arguments

- `[ -cell_type <retention_type>]`: Specifies the type of the retention cell.
- `[ -retention_pin <{pin_name pin_type disable_value}>]`: Defines the retention pin of the library cell. The valid values for the pin_type

argument are restore, save, and save_restore. The value values for the disable_value argument are 0 and 1.

Examples

The following example sets all library cells whose name matches *DRFF* as DRFF retention cells. The retention pin, RETN, is a save-restore pin and is disabled by a logic 1 value.

```
prompt> set_retention_cell *DRFF* -cell_type DRFF \\
-retention_pin {RETN save_restore 1}
```

set_top_module

Description

This command changes the top instance.

Syntax

```
set_top_module

-topInst <string>
```

Arguments

- `-topInst <string>`: Use this option to specify the top instance name.

Common Commands

add_tag_field

Description

This command adds a new field to a tag. The field can be an existing field, which was not previously used for that tag; or a new user field. After adding the field to the tag, use set_violation_field to fill in the string for each violation.

Syntax

```
add_tag_field <tag> [<tag>]
[-type <value>]
```

Arguments

- **Default Argument:** `<tag>:` Use this option to specify the tag name.
- **Default Argument:** `<field>`: Use this option to specify the field name. Use colon separated name if field needs to be added to complex user defined field.
- **Default Argument:** `[<app>]`: Use this option to specify the application name.
- `[ -type <value>]`: Use this option to specify the field type. Possible option values are string and list. Default value is string.

analyze

Description

This command analyzes the specified HDL source files and stores the design templates they define into the specified library in a format ready to specialize and elaborate to form linkable cells of a full design.

Using this command, the user can specify multiple source files in single language in one command. Upon completion of the command all specified files are analyzed and ready to be elaborated. The command returns 1 on success and 0 on failure.

When the -vcs option is specified, other options cannot be specified, specify the entire vlogan/vhdlan command line within '{' and '}'

Syntax

```
analyze <file_list>
-format <verilog | sverilog | vhdl | sysc | spi>
-library <library_name>
-work <library_name>
-define <define_macros>
[-vcs <vcs_cmd>]
[-netlist]
```

Arguments

- **Default Argument:** <file_list>: Use this option to specify the list of files to be analyzed. When specifying more than one file, separate the names with a space and enclose the list of names in braces ({}) or within quotation (" ").
- `-format <verilog | sverilog | vhdl | sysc | spi>`: Use this option to specify the input file format type. The supported types are verilog: IEEE Standard Verilog format, VHDL: IEEE Standard VHDL, sverilog: IEEE Standard System Verilog.
- `-library <library_name>`: Use this option to remap the work library in the same way as the -library option. It specifies the library name to which work must be mapped. The -work option references the library where the top-level unit of the elaboration hierarchy is compiled. If the elaboration starts from a module, then it is the library name where the module is compiled. If the elaboration starts from a configuration (in case of -v2kconfig), it is the library name where the configuration is compiled.
- `-work <library_name>`: Use this option to remap the work library in the same way as the -library option. It is an alias for the -library option for VHDL only. It specifies the library name to which work must be mapped. The -work option references the library where the top-level unit of the elaboration hierarchy is compiled. If the elaboration starts from a module, then it is the library name where the module is compiled. If the

elaboration starts from a configuration (in case of -v2kconfig), it is the library name where the configuration is compiled.

- `-define <define_macros>`: Use this option to specify a list of top-level macros. This option can only be used with the Verilog and System Verilog formats.
- `[ -vcs <vcs_cmd>]`: Use this option to analyze the design files using VCS analyze command. This option specifies all VCS-specific command-line options. Options specified by -vcs follow the VCS command-line syntax. The vlogan and vhdlan arguments and switches used in this switch must be enclosed in curly braces.

  The [-sverilog|-verilog] argument specifies the format of the files to be analyzed.

  The [-y directory_path] argument specifies directories that contain the library files to be searched for unresolved module instantiations in the design.

  The [+libext+.extension1+...] argument specifies the extension to consider during a files search in the -y library directories. The default is no extension.

  The [-v library_file] holds several module definitions, to be used during the search for unresolved modules.

  The [-f command_file] argument specifies a command file.

  The [+define+macro_name+...] argument defines a macro.

  The [+incdir+dir1+...] argument includes directories in the search list.
- `[ -netlist]`: Enables VC Static to invoke different design readers for RTL file and Netlist file present in the tcl script. When RTL and netlist files are present in the same tcl script, then you must read the netlist designs using the -netlist option. This improves the performance of the analyze command.

Examples

The following example analyzes a VHDL file named top.vhd. The library in which this will be analyzed is my_work.

```
analyze -format vhdl -work my_work top.vhd
```

The following example analyzes a VHDL file named my_cfg.v using the VCS analyze command.

```
analyze -vcs {my_cfg.v -sverilog}
```

change_names

Description

The change_names command changes the names of ports, cells (including physical-only cells), and nets in a design to conform to specified name rules. This command cannot be used to change the name of library cells or bus ports.To change the name of bus ports, you must first use the undefine_bus command to remove the bus property from the port. If an object name does not conform to the specified rules, the tool changes the name and ensures that the new name is unique within the design.

To show the effects of running this command without actually making the changes, use the report_names command.

There are two primary reasons for using the change_names command: It enables you to modify design object names in the tool so that the names match those that are ultimately created for a saved design. The names the tool displays in reports and in other information match those used in your target system. It enables you to define naming rules specific to your target system. For example, you might be using VHDL as a design transfer mechanism, but the naming rules of your system might be more restrictive than those supported by the true VHDL format.

When you run the change_names command with no options, it operates on the ports, cells, and nets in the current design. When you specify the -hierarchy option, changes are expanded to include all design objects within the current design hierarchy. For runtime reasons, it's best to use the -instance option only when you have a small list of instances to be renamed. If you have a big list of instances to be renamed, you must use the -rule option with change_names or use the -skip_inactive_constraints option and apply the constraints after running change_names -instance.

Syntax

```
change_names

[-hierarchy]

[-rules <name_rules>]
```

Arguments

- [-hierarchy]: Use this option to apply change_names to the hierarchy.
- [-rules <name_rules>]: Use this option to specify rules to be used for change_names.

check_hdl_lib

Description

This command enables language check on VCS library files and library modules. If enabled, VC Lint performs rule checking on the library files passed with "-v" and "-y" options.

Syntax

```
check_hdl_lib
[-all]
[-lib <libFile list>]
[-module <libModule list>]
[-lib_file <libListFile>]
[-module_file <libModuleListFile>]
-disable
```

Arguments

- [-all]: Use this option to check all the library files and modules.
- [-lib <libFile list>]: Use this option to check the specified library files.
- [-module <libModule list>]: Use this option to check the specified library modules.
- [-lib_file <libListFile>]: Use this option to check the library files in the specified file.
- [-module_file <libModuleListFile>]: Use this option to check the library modules in the specified module.

- -disable: Use this option to disable language check violations on libcells.

Examples

The following example enables language check on all the library files and modules.

```
check_hdl_lib -all
```

checkpoint_session

Description

Use the checkpoint_session command to save a session. This command saves the session in the <sessionName>_rtdb /checkpoints directory.You can add multiple checkpoints during a single run at different stages, but these checkpoints must have different sessionName specified to avoid over writing of the files. If two checkpoint_session commands are run one after another with the same session_name, then the output of the second command will overwrite the first command output.

Syntax

```
checkpoint_session
        -session <session_name>
        [-full]
        [-incremental]
```

Arguments

- -session <session_name>:Use this option to checkpoint this session under a specified name.
- [-full]: Use this option to create a full checkpoint.
- [-incremental]: Use this option to create an incremental checkpoint.

configure_module_synthesis

Description

This command configures module level synthesis configurations.

Syntax

```
configure_module_synthesis
[-enable]
[-disable]
[-config <configuration>]
[-module_list <module_list>]
```

Arguments

- `[ -enable]`: Use this option to enable module configuration.
- `[ -disable]`: Use this option to disable module configuration.
- `[ -config <configuration>]`: Use this option to specify module configuration.
- `[ -module_list <module_list>]`: Use this option to specify list of modules.

configure_libcell_uniquification

Description

Configures libcell uniquification.

Arguments

- `[-skip_sequential]`: Skips uniquification of all sequential elements.
- `[-skip_module <libcell-list>]`: Skips uniquification of all instances of the specified libcell.
- `[-skip_instance <libcell-instance-list>]`: Skips uniquification of the specified instances.

configure_tcl_command

Description

This command disables tcl commands. The commands can be enabled again with -enable option.

Syntax

```
configure_tcl_command <cmd>

-enable

-disable
```

Arguments

- **Default Argument:** <cmd>: Use this option to specify the command to be disabled.
- `-enable`: Use this option to enable the given command.
- `-disable`: Use this option to disable the given command.

configure_unobservable_logic_identification

Description

This command configures unobservable logic identification.

Syntax

```
configure_unobservable_logic_identification

-blackbox_endpoints yes/no/auto

-hierpin_endpoints yes/no/auto

-none/sequential/combinational
```

Arguments

- `-blackbox_endpoints yes/no/auto`: Use this option to treat black-box inputs as primary outputs. The values are auto, no, and yes.
- `-hierpin_endpoints yes/no/auto`: Use this option to treat hierarchical output pins as primary outputs. The values are auto, no, and yes.

- `-none/sequential/combinational`: Use this option to identify no, sequential, or combinational unobservability. The values are combinational, none, and sequential.

configure_waiver_filter_field

Description

This command configures filter field of a tag. Using this command, you can customize filter fields for certain tags in waiver window. If this filter command is not set, all fields are enabled in waiver window for each tag.

Syntax

```
configure_waiver_filter_field <tagname>
```

```
-enable
```

```
-disable
```

Arguments

- **Default Argument:** <tagname>: Use this option to specify the tag name.
- **Default Argument:** <fieldname>: Use this option to specify the field of a tag.
- `-enable`: Use this option to enable field in waiver filter template in GUI.
- `-disable`: Use this option to disable field in waiver filter template in GUI.

Examples

Specify the tag and field names, and use the -enable or -disable options to enable or disable tags.

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field
{Source:PinName} -enable
```

All the fields must include subfields till leaf level as below:

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field
{SourceInfo:PowerNet:NetName} -enable
```

```
configure_waiver_filter_field -tag ISO_INST_MISSING -field
{SourceInfo:PowerNet:NetName} -enable
configure_waiver_filter_field -tag ISO_INST_MISSING -field
{SinkInfo:GroundNet:NetName} -disable
configure_waiver_filter_field -tag PG_SUPPLY_NOPORT -field
{SupplyPort:PortName} -enable
```

create_clock

Description

This command creates a clock object in the current design and defines the specified source_objects as clock sources in the current design. A pin or port can be a source for a single clock. If source_objects is not specified, but a clock_name is given, a virtual clock is created. A virtual clock can be created to represent an off-chip clock for input or output delay specification. For more information about input and output delay, refer to the set_input_delay and set_output_delay command man pages.

To show information about all clock sources in a design, use the report_clock command. To get a list of clock sources, use the get_clocks command. To return sequential cells related to a given clock, use the all_registers command. To undo create_clock, use the remove_clock command.

Syntax

```
create_clock [<value>]

[-period <period_value>]

[-name <clock_name>]

[-waveform <edge_list>]

[-add]

[-comment <string>]
```

Arguments

- **Default Argument:** [<>]: Specify this option to use clock as reference clock [formal].

- **Default Argument:** [<value>]: Use this option to specify clock initial value (0/1 - default 0) at time 0 [formal].
- **Default Argument:** [<>]: Use this option to specify a list of pins or ports on which to apply this clock. If you do not use this option, you must use -name clock_name, which creates a virtual clock not associated with a port or pin. If you specify a clock on a pin that already has a clock, the new clock replaces the old clock unless you use the -add option.
- `[ -period <period_value>]`: Use this option to specify the period of the clock waveform in library time units.
- `[ -name <clock_name>]`: Use this option to specify clock name.If you do not use this option, the clock is given the same name as the first clock source specified in source_objects. If you do not use source_objects, you must use this option, which creates a virtual clock not associated with a port or pin. Use this option along with source_objects to give the clock a more descriptive name than that of the pin or port where it is applied. If you specify the -add option, you must use the -name option and the clocks with the same source must have different names.
- `[ -waveform <edge_list>]`: Use this option to specify the rise and fall edge times, in library time units, of the clock over an entire clock period. If the first time in the list is a rising transition, typically it is the first rising transition after time zero. There must be an even number of increasing times, and they are assumed to be alternating rise and fall times. The numbers must represent one full clock period. If -waveform edge_list is not specified, but -period period_value is, a default waveform with a rise edge of 0.0 and a fall edge of period_value/2 is assumed.
- `[ -add]`: Use this option to either add the clock to the existing clock or to overwrite the existing clock. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the -name option. Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the synthesis timing engine must explore all possible combinations of launch and capture clocks.
- `[ -comment <string>]`: Use this option to specify comment. It allows the command to accept a comment string. The tool honors the

annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the write_sdc or write_script command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

create_generated_clock

Description

This command creates a generated clock object. Creates a generated clock in the current design. If successful, the command returns a collection containing the new or modified generated clock. You can specify a pin or a port as a generated clock object. The command also specifies the clock source from which it is generated. The advantage of using this command is that whenever the master clock changes, the generated clock automatically changes. .If you use this command on an existing generated_clock object, it overwrites its attributes. The generated_clock objects are expanded to real clocks at the time of analysis. To display information about generated clocks, use the report_clocks command.

Syntax

```
create_generated_clock <master_pin>

-name <clock_name>

-divide_by <divide_factor>

-multiply_by <multiply_factor>

-duty_cycle <high_pulse_percentage>

-edges <edge_list>

-edge_shift <edge_shift_list>

-invert

-preinvert

-add

-combinational

-comment <string>
```

```
-pll_output <output_pin>
-pll_feedback <feedback_pin>
-master_clock <clock_name>
```

Arguments

- **Default Argument:** <master_pin>: Use this option to specify source clock object from which this clock is derived (port or pin object).
- **Default Argument:** <>: Use this option to specify collection of objects.
- `-name <clock_name>`: Use this option to specify clock name.
- `-divide_by <divide_factor>`: Use this option to specify frequency division factor (integer).
- `-multiply_by <multiply_factor>`: Use this option to specify frequency multiplication factor (integer).
- `-duty_cycle <high_pulse_percentage>`: Use this option to specify duty cycle(of high pulse width), if frequency multiplication is used.
- `-edges <edge_list>`: Use this option to specify a list of positive integers that represents the edges from the source clock that are to form the edges of the generated clock.
- `-edge_shift <edge_shift_list>`: Use this option to specify a list of floating-point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform.
- `-invert`: Use this option to invert the generated clock signal.
- `-preinvert`: Use this option to create a generated clock based on the inverted clock signal.
- `-add`: Use this option to add new options to the original clock.
- `-combinational`: Use this option to specify combinational signal.
- `-comment <string>`: Use this option to specify comment.
- `-pll_output <output_pin>`: Use this option to specify the output pin of the PLL which is connected to the feedback pin.
- `-pll_feedback <feedback_pin>`: Use this option to specify the feedback pin of the PLL.
- `-master_clock <clock_name>`: Use this option to specify master clock.

create_interface_wrapper

Description

This command creates the wrapper that instantiates the interfaces specified using the add_top_interface commands, and connects the top module ports to the interfaces/modports specified via the connect_top_port commands. If the setup is incompletely specified, an error is issued and the command is ignored.

Syntax

```
create_interface_wrapper

-top <top_module_name>

[-name <wrapper_name>]

[-output <wrapper_file>]

[-donot_compile]
```

Arguments

- `-top <top_module_name>`: Use this option to specify top module that is instantiated in wrapper.
- `[ -name <wrapper_name>]`: Use this option to specify the name of wrapper (default top_<top_module_name>).
- `[ -output <wrapper_file>]`: Use this option to specify the output file for generated wrapper.
- `[ -donot_compile]`: Use this option to create wrapper but not compile.

Examples

```
vc_static_shell> add_top_interface â€“top dut â€“interface
atb_if_t -instance ifx_a0
vc_static_shell> connect_top_port â€“port pm0 -interface
ifx_a0.master
vc_static_shell> connect_top_port â€“port ps0 â€“interface
ifx_a0.slave
vc_static_shell> read_file â€“top dut â€“sva â€“vcs { dut.v }
vc_static_shell> create_clock dut.clk â€“period 100
vc_static_shell> create_clock ifx_a0.clk â€“period 100
vc_static_shell> create_reset dut.resetn â€“low
vc_static_shell> create_reset ifx_a0.resetn â€“low

The generated wrapper would look like the following:
// Automatically Generated File: <date>
module top_dut;
atb_if_t ifx_a0();
dut dut (.pm0(ifx_a0.master),.ps0(ifx_a0.slave));
endmodule
```

create_reset

Description

This command is convenient and defines simulation reset value as well as formal analysis value for a set of signals, all in one command. These signals can be internal nets, or primary inputs (ports) of the design. Use of create_reset is the same as pairs of sim_force and set_constant, but in more concise and less error prone form. Return value is a collection, nets or ports that are defined as resets with this command.

Syntax

```
create_reset [source_objects]

-sync

-async

-type {reset|set|both}

-name <reset_name>

-sense {low|high|any}

-add

-tdr
```

Arguments

- `[source_objects]`: List of nets, ports, and pins on which this reset is defined.

- -sync: Use this option to specify if the signal drives synchronous reset. If this argument is not specified, the reset is considered as async by default.
- -async: Use this option to specify if the signal drives asynchronous reset.
- -type {reset|set|both}: Use this option to specify the type of reset signal. The default value is reset.
- -name <reset_name>: Use this option to specify the name of reset signal.
- -sense {low|high|any}: These switches indicate the assertion polarity for the reset signal. -low means the reset is asserted low. -high indicates the reset is asserted high. any indicates the reset is asserted to low or high. One of these switches is required.
- -add: Use this option to add new resets in addition to existing reset specified in the default argument.
- -tdr: Use this option to specify that the specified reset is test and/or debug reset.

Examples

To create asynchronous reset on a pin a_RST_1:

```
create_reset a_RST_1 -type async
```

create_static

Description

Use this command to specify the design object (port/pin/net) which must be treated as Quasi-static signal. Hierarchical objects can be specified in this command. Quasi-static signals are the ones that are unlikely to change and remain static most of the times.

Syntax

```
create_static <signal_name>
        [-check_glitch]
        [-des_clock <clk_name>]
```

```
[-cycle <cycle value>]
[-rise_edge <signal_name>]
```

Arguments

- **Default Argument:** `<signal_name>`: Use this option to specify hierarchical pin or port or net name, which needs to be treated as quasi static.
- `<signal_name>`: Use this option to specify hierarchical pin, port, or net name that needs to be treated as quasi static.
- `[-check_glitch]`: Use this option to start Glitch analysis on the specified Quasi objects.
- `[-des_clock <clk_name>]`: Use this option to specify destination clocks.
- `[-cycle <cycle value>]`: Use this option to specify the cycle of the quasi static constraint. The value of this option must be an integer between '1' and '100'; both inclusive.
- `[-rise_edge <signal_name>]`: Use this option to specify rise edge signal of the quasi static constraint.

Examples

To make a EN signal quasi static:

```
create_static -name EN
```

To make a hierarchical pin quasi static:

```
create_static -name [get_pins -hierarchical Top/MUX/Sel
```

define_design_lib

Description

This command defines hdl design library.You can use this command multiple times, once for each logical library name that is to be mapped to a physical library. You can run the commands at any time, but they are valid only before the first read into that logical library. When multiple logical libraries point to the same physical library specified by using the library_name argument, the tool uses the first logical library name that is defined. The path specified by using the -path option is used for proper mapping. All logical libraries that point to the same path, must share the same physical library.

A logical library cannot be mapped to two physical libraries in the same container. But they can be mapped to two physical libraries in different containers.

This command returns 0 to indicate failure and 1 to indicate success.

Syntax

```
define_design_lib <library_name>

-path <directory>
```

Arguments

- **Default Argument:** <library_name>: Use this option to specify the design library to be mapped.
- `-path <directory>`: Use this option to specify the directory to map the library.

define_name_rules

Description

This command defines a set of rules for naming design objects. Name rules are used by the change_names and report_names commands. The report_name_rules command displays a listing of the name rules currently defined in the shell.Name rules can be defined in multiple calls to the define_name_rules command.

The -type option enables you to define rules that apply to a specific object type, such as port, cell, or net). Each call to define_name_rules is additive or overrides previous calls for a specific name rules.

Syntax

```
define_name_rules

[-allowed <char>]

[-first_restricted <string>]
```

```
[-last_restricted <string>]
[-prefix <prefix>]
[-map <map_string>]
[-type port|cell|net]
[-restricted <string>]
[-replacement_char <char>]
[-remove_char]
[-rule_name <rule_name>]
```

Arguments

- `[ -allowed <char>]`: Use this option to specify a set of allowed chars in names.
- `[ -first_restricted <string>]`: Use this option to specify set of restricted first chars in name.
- `[ -last_restricted <string>]`: Use this option to specify set of restricted last chars in name.
- `[ -prefix <prefix>]`: Use this option to specify prefix to be used when creating names.
- `[ -map <map_string>]`: Use this option to name mapping and substitution.
- `[ -type port|cell|net]`: Use this option to apply rules to port or cell or net.
- `[ -restricted <string>]`: Use this option to specify a set of restricted chars in names.
- `[ -replacement_char <char>]`: Use this option to specify replacement char for name changes.
- `[ -remove_char]`: Use this option to remove chars rather than replacing illegal chars.
- `[ -rule_name <rule_name>]`: Use this option to specify rule name.

diff_database

Description

Given a previous run with violations on disk, this command reads the violations from the previous run. For each violation which is common to both runs, it either deletes the violation from the current run, or adds a waiver for the violation in the current run. The result is a new, smaller database where the current run contains only new violations, not found in the previous run.

Either the option -remove or -waive must be given, to tell what operation should be performed on the violations in both databases.

The previous run will normally have been saved by checkpoint_session in a directory such as myrun_cpdb. Specify the session name with "-checkpoint myrun". Since the database is not normally saved to disk, make sure to save it before checkpointing by executing "save_db".

The older save/restore capability is also supported; specify the session name with "-session myrun" to read from myrun_rtdb. In less common flows, a database file may be directly available on disk; specify a filename with "-raw mypath/report.db".

Syntax

```
diff_database

[-remove]

[-waive]

[-checkpoint <session>]

[-session <session>]

[-raw <filename>]

[-app <Application>]
```

Arguments

- `[ -remove]`: Remove violations found in previous database
- `[ -waive]`: Waive violations found in previous database
- `[ -checkpoint <session>]`: Checkpoint session name
- `[ -session <session>]`: Saved session name
- `[ -raw <filename>]`: Raw database filename (less common)

- `[ -app <Application>]`: Use this option to specify the application name.

disable_tag_field

Description

This command prevents a field from being printed in the report_* output for a tag.

Syntax

```
disable_tag_field <tag>
```

Arguments

- **Default Argument:** `<tag>`: Use this option to specify the tag name.
- **Default Argument:** `<field>`: Use this option to specify the field that must not appear in the report_* output.

elaborate

Description

This command builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration.

Note: Using this command, the user can elaborate design from pre-analyzed design files, from a specified top module. At the completion, this command returns 1 on success and 0 on failure.

Note: When the -vcs option is specified, other options cant be specified.

Syntax

```
elaborate <design_name>

[-library <library_name>]

[-work <library_name>]
```

```
[-architecture <arch_name>]
[-parameters <param_list>]
[-file_parameters <file_list>]
[-vcs <vcs_command>]
[-sva]
[-sva2005]
[-v2kconfig <configuration-name>]
[-buildTop <dut name>]
[-j <number_of_processes>]
```

Arguments

- **Default Argument:** <design_name>: Use this option to specify the design name which needs to be elaborated. The design can be Verilog module of VHDL entity.
- `[ -library <library_name>]`: Use this option to remap the work library to library_name. This option can only be used with the VHDL format. By default, the analyze command stores all output in the work library. To store design elements in libraries other than library specified by using the -work option, use the -library option.
- `[ -work <library_name>]`: Use this option to remap the work library in the same way as the -library option. It is an alias for the -library option for VHDL only. Specifies the library name to which work must be mapped. The -work option references the library where the top-level unit of the elaboration hierarchy is compiled. If the elaboration starts from a module, then it is the library name where the module is compiled. If the elaboration starts from a configuration (in case of -v2kconfig), it is the library name where the configuration is compiled.
- `[ -architecture <arch_name>]`: Use this option to specify the name of the architecture. In VHDL, the default architecture is the most recently analyzed architecture.
- `[ -parameters <param_list>]`: Use this option to specify a list of design parameters enclosed in quotes for Verilog. For VHDL use -vcs {-gv ...} option. Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, 8,7,5) or

on parameter names (for example, N=>8,M=>6). It is acceptable to mix ordered and named parameter specifications, as long as the ordered parameters are listed first..

- `[ -file_parameters <file_list>]`: Use this option to specify a list of files that contain parameter specifications. The specifications are appended to the parameters listed in the -parameters option (if present). The syntax of the parameter file is identical to the syntax between the parameter-delimiters (#) in the -parameters option, except that the backslashes (\\) at the end of lines and the hashes (#) must be omitted. The specified files are searched for in the search_path.
- `[ -vcs <vcs_command>]`: Use this option to read the design using VCS command arguments and switches. Options specified by -vcs follow the VCS command-line syntax. When specifying the VCS commands, you must include them either in â€˜{ }â€™ or double quotes. Note: (1)If there is one design top, it should not be passed using vcs arguments. that is, elaborate -vcs {designtop}. It should be passed as follows: %elaborate designtop (2) For a model with several top modules (in following example: dut_top and tb_top), you must pass the arguments as follows: %elaborate dut_top -vcs tb_top.

  The [-sverilog|-verilog] argument specifies the format of the files to be analyzed.

  The [-y directory_path] argument specifies directories that contain the library files to be searched for unresolved module instantiations in the design.

  The [+libext+.extension1+...] argument specifies the extension to consider during a files search in the -y library directories. The default is no extension.

  The [-v library_file] holds several module definitions, to be used during the search for unresolved modules.

  The [-f command_file] argument specifies a command file.

  The [+define+macro_name+...] argument defines a macro.

  The [+incdir+dir1+...] argument includes directories in the search list.
- `[ -sva]`: Use this boolean switch to compile any discovered SVA or PSL properties into formally analyzable form when the design is read in. Properties declared in the source code will be initially enabled as they are declared in the code (assume, assert, cover). Use this option to process SVA/PSL during compilation using 2009 semantics.

- `[ -sva2005]`: Use this option to process SVA/PSL during compilation using 2005 semantics.
- `[ -v2kconfig <configuration-name>]`: Use this option to specify the v2k configuration. v2k configuration cannot be directly specified as a design top.
- `[ -buildTop <dut name>]`: Use this option to specify the DUT down from which synthesis model is generated.
- `[ -j <number_of_processes>]`: Use this option to specify the number of processes that can be used for parallel compilation in the RTL flow. Value >= 1

Examples

The following example elaborates a design named top from a library name my_work.

```
elaborate top -work my_work
```

generate_waiver_commands

Description

This command can be used to generate field-based waiver(s) corresponding to violations.

Syntax

```
generate_waiver_commands
        -app <app list>
        [-status]
        [-viol <viol list>]
```

Arguments

- `-app <app-list>`: Waivers will be generated for the specified applications.
- `[-status]`: Waivers are generated with the specified state. Valid values include: `Acknowledged`, `NeedsInfo`, `Open`, `Waived`, `Waived_Temp`, and `Ignore`.

- `[-viol <viol list>]`: Field-based waivers are generated for the specified violation id(s).

Examples

The following command generates filtered report:

```
generate_waiver_commands -app CDC -viol [ report_cdc -tag
SYNCCDC_CTRLPATH_FULL -id_list ]
```

In this case, VC SpyGlass CDC generates the following data:

```
waive_cdc  -add SYNCCDC_CTRLPATH_FULL_98 -tag
SYNCCDC_CTRLPATH_FULL -filter { ContainerInstance ==
"a_syncInfo" && DestClockInfoList:DestClockInfo:ClockName ==
"c2" && DestClockInfoList:DestClockInfo:ClockObject == "clk2"
&& DestObject == "a_syncInfo/dest/out/Q[0]" && DestObjectType
== "flop" && DstFileLine:FileName == "../test.v" &&
DstFileLine:LineNumber == "295" && Module == "MultiSrcNFF" &&
ReasonInfoList:ReasonInfo:ReasonCode == "SYNC_BY_NFF" &&
ReasonInfoList:ReasonInfo:ReasonCodeMsg == "[INFO] Multi Flop
Synchronizer detected" &&
SrcClockInfoList:SrcClockInfo:ClockDomainId == "1.1" &&
SrcClockInfoList:SrcClockInfo:ClockName == "c1" &&
SrcClockInfoList:SrcClockInfo:ClockObject == "clk1" &&
SrcFileLine:FileName == "../test.v" && SrcFileLine:LineNumber
== "295" && SrcObject == "a_syncInfo/b_src/out/Q[0]" &&
SrcObjectType == "flop" && SyncDepth == "2" && SyncObject ==
"a_syncInfo/syncI/out/Q[0]"}
```

The following command generates the complete report:

```
generate_waiver_commands -app CDC
```

The `generate_waiver_commands` command honors the *configure_waiver_filter_field* command. If you have defined some fields for a tag by using the *configure_waiver_filter_field* command or if it is predefined by the application, waivers are generated using the defined fields. Else, all fields are used.

For example, the `configure_waiver_filter_field -tag SYNCCDC_CTRLPATH_FULL -disable -field SrcObjectType` command can be used to disable generation for the Source object type.

get_blackbox

Description

This command returns the collection of black-boxed elements in the design.

Syntax

```
get_blackbox

-designs

-automatic

-unresolved
```

Arguments

- `-designs`: Use this option to create a collection of designs which are black-boxed.
- `-automatic`: Use this option to create a collection of cells that are automatically black-boxed by the tool, while loading the design. This usually happens if an instance is present in the design, but its definition is missing in the design. This can also happen if a module has non-synthesizable constructs, hence the hardware inference tool has black-boxed that module because that module cannot be synthesized.
- `-unresolved`: Use this option to return unresolved modules in the design.

Examples

The following example shows the usage of the get_blackbox command:

```
vc_static_shell> get_blackbox -designs
{"module1"}
```

get_clock_relationship

Description

This command reports the relationship between any given number of clocks based on the status of current clock groups. The relationship can be asynchronous,logically_exclusive or physically_exclusive. Multiple types of relationships or no relationship can exist between clocks.

Syntax

```
get_clock_relationship

-clocks

[-type <type>]

[-csv]

[-file <file name>]

[-gui]
```

Arguments

- `-clocks`: Use this option to specifiy the clocks for which the relationship needs to be reported.
- `[ -type <type>]`: Use this option to check if a particualar type of relationship exists between the given clocks.
- `[ -csv]`: Use this option to report the output in comma separated format.
- `[ -file <file name>]`: Use this option to dump the output to a text file.
- `[ -gui]`: Use this option to report the output in GUI.

Examples

The following example queries the relationship between clocks c1 and c2

```
prompt> get_clock_relationship {c1 c2}
asynchronous
```

The following example queries the if the clocks c1 cand c2 have a logically exclusive relationship.

```
prompt> get_clock_relationship {c1 c2} -type
logically_exclusive
true
```

The following example reports the relationship between clocks c1 and c2 in

CSV format. -csv option is avaliable only in non-PT mode.

```
prompt> get_clock_relationship {c1 c2} -csv
'Clock1','Clock2','Relationship'
c1,c2,asynchronous
c2,c1,asynchronous
```

get_constant_sources

Description

Reports the RTL/SCA constant sources propagating to a signal. The command calls a fanin traversal from the given constant signal, until it reaches RTL constants (supply/ground) or set_case_analysis. Then it reports the constant sources along with their types.

Syntax

```
get_constant_sources <>
```

Arguments

- **Default Argument:** <>: Object to which constant(s) propagate

Examples

```
> get_constant_sources FF/CLK
Type Object
RTL scan_en
SCA sel1
SCA sel2
```

get_exception

Description

This command creates a collection of exception objects which are used to store constraints information of any design object.

The command returns a collection, if the given pattern matches any design object and constraints are defined on that object. If no object matches, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as get_attribute. In addition, you can assign the result to a variable.

Syntax

```
get_exception <design_object_name>
```

```
[-type]
```

Arguments

- **Default Argument:** <design_object_name>: Search for design objects which matches this pattern and collects constraints on those objects.
- `[ -type]`: Searches for constraints in the given type only. This can be one of set_case_analysis, set_cdc_ignore_path, set_clock_sense or set_disable_timing.

Examples

The following example queries the constraints defined on the pin RG1/Q.

```
prompt> get_attribute [get_exceptions -type set_case_analysis
RG1/Q] type
set_case_analysis
```

The following example queries the file line information of the constraints defined on the pin RG1/Q.

```
prompt> get_attribute [get_exceptions RG1/Q] source
{File: ./test.sdc Line: 7} {File: ./test.sdc Line: 8} {File: ./
test.sdc Line: 9}
```

The following example queries the commands of the constraints defined on the pin RG1/Q.

```
prompt> get_attribute [get_exceptions RG1/Q] command
{set_case_analysis 1 [get_pins RG1/Q]} {set_clock_sense
[get_pins {RG1/Q}]} {set_disable_timing [get_pins { RG1/Q }]}
```

get_field_subfield

Description

This command is used to query the qualified fields of violation which belongs to record type.

Syntax

```
get_field_subfield <tag>

[-leaf]

[-quiet]
```

Arguments

- **Default Argument:** <tag>: Use this option to specify the tag name you want to query.
- **Default Argument:** <field>: Use this option to specify the field name in the tag you want to query.
- `[ -leaf]`: Use this option to recursively query the subfields.
- `[ -quiet]`: Use this option if you want to suppress the error messages.

Examples

```
%vc_static_shell> get_field_subfield DESIGN_PIN_SCMR
InstanceInfo
PowerNet GroundNet PowerMethod GroundMethod
```

```
%vc_static_shell> get_field_subfield DESIGN_PIN_SCMR
InstanceInfo -leaf
PowerNet:NetName PowerNet:NetType GroundNet:NetName
GroundNet:NetType PowerMethod
GroundMethod
```

```
%vc_static_shell> get_field_subfield DESIGN_PIN_SCMR
wrong_instance
Error: undefined field 'wrong_instance'
```

```
%vc_static_shell> get_field_subfield DESIGN_PIN_SCMR
wrong_instance -quiet
```

get_glassbox

Description

This command returns list of objects which are listed as glassbox.

get_license

Description

Use this command to check-out the application specific license keys explicitly. The command reserves all keys required by a particular script at the beginning of a batch script. This command returns the number of application license keys successfully obtained. For more details on the application license keys, see the VC Static Platform Product Installation Notes.

Syntax

```
get_license <feature-list>

-quantity <num>

[-licwait <minutes>]
```

Arguments

- **Default Argument:** <feature-list>: Specifies the list of application specific keys to be reserved.
- `-quantity <num>`: Specifies the total number of licenses to be checked out: num >= 1
- `[ -licwait <minutes>]`: Specifies the time period for which the license server must wait (in minutes) before terminating the run.

get_no_msg_reporting_tags

Description

This command returns list of tags which doesn't report any violation

It will check all the tags which are enabled for the required applcation.

A tag may not report violation due to either it has not been run or it has not found any violation scenario which needs to reported.

Syntax

```
get_no_msg_reporting_tags [<app>]
```

Arguments

- **Default Argument:** [<app>]: Use this option to specify the application name , default Lint.

Examples

To get information about the tags which doesn't report any violation, use:

```
vc_static_shell> get_no_msg_reporting_tags Lint
```

get_pi_drive_clock

Description

This command returns all reset/clock pairs in the database.

get_readmsg_attribute

Description

This command returns specific attribute for a design read message.

Syntax

```
get_readmsg_attribute <>
```

Arguments

- **Default Argument:** <>: Use this option to specify message name.
- **Default Argument:** <>: Use this option to specify attribute name.

Examples

The following example retrieves the count attribute in the message

UPF_OBJECT_NOT_FOUND_ERROR.

```
get_readmsg_attribute UPF_OBJECT_NOT_FOUND_ERROR count
```

get_readmsg_field

Description

This command returns string data for one qualified field of the design read message.

Syntax

```
get_readmsg_field <id>
```

Arguments

- **Default Argument:** <id>: Use this option to identify integer violation.
- **Default Argument:** <field>: Use this option to specify qualified field name.

Examples

The following example returns the 'file' field of the message with id given by $id.

```
get_readmsg_field $id file
```

get_readmsg_ids

Description

This command returns the list of ids of violations for the given design read message.

Syntax

```
get_readmsg_ids <name>
```

Arguments

- **Default Argument:** <name>: Use this option to specify existing design read message name.

Examples

The following example returns the list of message ids corresponding to the message, UPF_OBJECT_NOT_FOUND_ERROR.

```
get_readmsg_ids UPF_OBJECT_NOT_FOUND_ERROR
```

get_readmsg_names

Description

This command returns list of design read message names.

Syntax

```
get_readmsg_names [<type>]
```

Arguments

- **Default Argument:** [<type>]: Use this option to specify design read type and default is UPF.

Examples

The following example retrieves all upf read messages.

```
get_readmsg_names upf
```

get_readmsg_names

Description

This command returns list of design read message names.

Syntax

```
get_readmsg_names [<type>]
```

Arguments

- **Default Argument:** [<type>]: Use this option to specify design read type and default is UPF.

Examples

The following example retrieves all upf read messages.

```
get_readmsg_names upf
```

get_supported_tags

Description

This command returns supported rule list for a given RCA.

Syntax

```
get_supported_tags -rca <ML_RCA_TAG>
```

Arguments

`-rca <ML_RCA_TAG>`

Specified RCA tag.

Examples

```
get_supported_tags -rca DEBUG_LINT_PORT_STRUCT_SIGNAL
```

get_violation_waiver

Description

This command returns the name of the waiver covering a single violation, or the empty string if it is not waived. For example, to get the name of the waiver covering violation 123, use:

vc_static_shell> set violation_name [get_violation_waiver 123]

To get a list of all the tags with violations in the current run, use get_violation_tags. To get a list of all the violation IDs for an individual tag, use get_violation_ids.

There are also commands that operate on tag definitions; for an overview, see get_tags.

Syntax

```
get_violation_waiver <id> [<id>]
```

Arguments

- **Default Argument:** <id>: Use this option to identify integer violation.
- **Default Argument:** [<app>]: Use this option to specify application name.

get_waiver_attribute

Description

This command returns specific attribute for waiver. This command helps to identify specific attributes in the filter database and can be used to identify waivers which possess a specific quality. One main usage of this command is to identify waivers with no violations waived (zero violation count) and correct/delete them.

Syntax

```
get_waiver_attribute <waiver>
```

Arguments

- **Default Argument:** <waiver>: Use this option to specify waiver name.
- **Default Argument:** <attribute>: Use this option to specify the attribute name.

get_waivers

Description

This command is used to query all the waivers which matches the specified pattern.

Syntax

```
get_waivers <pattern>
```

Arguments

- **Default Argument:** <pattern>

Use this option to specify pattern of waivers to be returned.

Examples

```
%vc_static_shell> get_waivers *ISO*
FLT-ISO_STRATEGY_MISSING
```

index_database

Description

This command creates an index to speed up multiple waivers. It speeds up subsequent waiver/filter operations by creating a cache for commonly requested fields. If only one or two simple waivers are executed, this command will occupy more time than simply executing the waivers. However, if many complex waivers are being executed, creating this cache saves time.

infer_clock_roots

Description

This command infers clock roots automatically without the need of defining the clock information using SDC.

This command does not provide any output. Use report_clock_roots -file file_name to dump inferred clock roots to a file. This file can be read using read_sdc file_name.

Syntax

```
infer_clock_roots
-exclude_latch
-sequentials <list_of_Qpins_or_cells>
```

Arguments

- `-exclude_latch`: Use this option to trace latch enables for clock root identification.
- `-sequentials <list_of_Qpins_or_cells>`: Use this option to specify sequential elements for which clock roots will be identified.

infer_reset_roots

Description

This command infers reset roots.

Syntax

```
infer_reset_roots
-include_latch
-report_reg <file_name>
```

Arguments

- `-include_latch`: Use this option to trace latch enables for reset root identification.
- `-report_reg <file_name>`: Use this option to dump register and its reset root to a specified file.

infer_setup

Description

This command infers clock roots/reset roots automatically without the need of defining the clock or reset information using SDC.

This command does not generate any output. Use the `write_inferred_setup -type (clock/reset) -file file_name` to include inferred clock roots/reset roots to a file.

Syntax

```
infer_setup
        -type <clock/reset>
        [-incremental]
        [-full]
        [-apply]
        [-infer_all_potentials]
        [-generated_clocks <true/false>]
        [-generated_resets <true/false>]
        [-sync_resets <true/false>]
        [-infer_gated <true/false>]
        [-include_hanging]
        [-check_vcs_clock]
        [-infer_latch_out]
        [-filter_names <clock_name|reset_name>]
```

Arguments

- `-type <clock/reset>`: Use this option to specify the type of root (clock/reset) to be inferred.
- `[ -incremental]`: Use this option to specify that user-defined commands are propagated first and then inference is done only from elements that are not receiving anything.
- `[ -full]`: Use this option to specify that all the roots are inferred.
- `[ -apply]`: Use this option to use inferred clocks directly.
- `[ -infer_all_potentials]`: Use this option to enable inference to all roots in auto detection.
- `[ -generated_clocks <true/false>]`: Use this option to dump the generated clocks.

- [-generated_resets <true/false>]: Use this option to dump the generated resets.
- [-sync_resets <true/false>]: Use this option to enable inference for sync resets.
- [-infer_gated <true/false>]: Use this option to enable inference for gated roots.
- [-include_hanging]: Use this option to enable inference of clocks on hanging nets.
- [-check_vcs_clock]: Use this option to infer paths with the vcs_clock attribute.
- [-infer_latch_out]: Use this option to enable inference of latch outputs as generated clock.
- [-filter_names <clock_name|reset_name>]: Use this option to not infer clock/reset which have the specified string in its name.

Examples

To infer clock root/reset root based upon type requested.

```
infer_setup -type clock

write_inferred_setup -type clock -file infer.sdc
```

link

Description

This command is used to link current design.This command uses the link_library and search_path variables to resolve design references. A "*" entry in the value of the link_library variable indicates that link should search all the designs already loaded in memory. If the link_library variable has no "*" entry, the already-loaded designs are not searched. The default value for the link_library variable is "* your_library.db". The search_path variable specifies a list of directory names that the link command uses to search for link_library files.

For simple file names (names that contain no '/' character), the link command looks for the files in the directories specified by the search_path variable. For absolute or relative path names, the search_path variable is not used. If the referenced designs are not found in any of the specified files, the link command looks in the search path directories for any .ddc or .db files that have the same name as the referenced design (for example, adder.ddc or adder.db). In this case, the .ddc files are searched first.

The order of directories in the search path and of the files in the link libraries is important. Search order during a link is (1)Local link library files (2)Link library files (3)Search path directories The first occurrence of a design reference is used.

Syntax

```
link

-cov <metric_type>

-llk <llk_type>

-aep <aep_type>

-inject_fault <fault_type>
```

Arguments

- `-cov <metric_type>`: Use this option to enable coverage instrumentation during compilation.
- `-llk <llk_type>`: Use this option to create livelock goals during compilation.
- `-aep <aep_type>`: Use this option to specify the types of AEP checks that should be instrumented as the design is compiled.
- `-inject_fault <fault_type>`: Use this option to inject behavioral faults in the design for doing sign-off with formal.

link_design

Description

The link_design locates all designs and library components that are referenced by the current design and links them to the current design. During linking, the tool loads all files specified by the link_path variable if they are not already in memory. Successful linking results in a fully instantiated design on which you can perform analysis.

Syntax

```
link_design
[-design_name <design>]
[-quiet]
```

Arguments

- `[ -design_name <design>]`: Specifies the design to be linked. The default is the current design.
- `[ -quiet]`: Does not print any information with this option.

Examples

The following example shows the usage of the link_design command:

```
vc_static_shell> read_verilog top.v
vc_static_shell> link_design top
```

list_all_waiver_files

Description

A waiver command can now be linked to a file when created from GUI or given through manage_waiver_file command. This command displays a complete list of files that are associated with such waivers.

llib

Description

This command lists information within library.This command lists the information of VHDL objects (entity, architecture, configuration, package and package body) and Verilog objects (modules) in specified design library. It supports for pure design i.e. Verilog, VHDL, *.sv and mix design as well. Before using this command, you must load design using analyze command and then use this command.

Syntax

```
llib
[-l]
[-r]
[-lib <library_name>]
[-all]
[-design_unit_name <design_unit_name>]
```

Arguments

- `[ -l]`: Use this option to show VCS version, timestamp, dependencies, package references, etc.
- `[ -r]`: Use this option to print architecture name for each entity and print package body name for each package.
- `[ -lib <library_name>]`: Use this option to search the logical name of the library.
- `[ -all]`: Use this option to search all logical libraries in setup file.
- `[ -design_unit_name <design_unit_name>]`: Use this option to specify design unit name (Configuration, Package, Entity or Module).

man

Description

This command shows the man page for the given command or message. This command is used to view the runtime documentation for Monet messages and commands in the Monet shell.

Syntax

```
man
-command_name
```

Arguments

- `-command_name`: Use this option to specify the man page name.

merge_database

Description

This command merges report database from a saved run into the current run.This command helps you to read a report database on the disk from a previous run, and merge it into the report database for the current run.

Given a previous run with violations on disk, this command reads the violations from the previous run. For each violation which is unique to the previous run, it adds a copy of the violation into the current violation database. The result is a new, larger database which is the union of both databases.

The previous run will normally have been saved by checkpoint_session in a directory such as myrun_cpdb. Specify the session name with "-checkpoint myrun". Since the database is not normally saved to disk, make sure to save it before checkpointing by executing "save_db".

The older save/restore capability is also supported; specify the session name with "-session myrun" to read from myrun_rtdb. In less common flows, a database file may be directly available on disk; specify a filename with "-raw mypath/report.db".

Syntax

```
merge_database

[-session <session_name>]

[-input_filename <input_filename>]

[-checkpoint <session>]
```

Arguments

- `[ -session <session_name>]`: Use this option to specify the name of saved session to merge messages from database.
- `[ -input_filename <input_filename>]`: Use this option to specify raw message file.
- `[ -checkpoint <session>]`: Checkpoint session name

read_file

Description

This command helps you to read in design source files, and link design in memory. This command can be used to load design in single language (Verilog/SV or VHDL)

Note: Using this command, the user can specify all source files in one command in a single language environment. The files get analyzed and then elaborated. Upon completion of the command the complete design has been loaded and is ready to be used. The command returns 1 on success and 0 on failure.

Note: When the -vcs option is specified, other options cant be specified.

Syntax

```
read_file

- <top>

-library <library name>

-define <list of verilog defines>

-work <library name>

-netlist

-parameters <string containing ordered or named parameters
separated by comma>

-vcs <vcs command line>

-vcs_elab <vcs elaborate command line>

-sva

-sva2005

-v2kconfig <configuration-name>

-buildTop <dut name>

-multi_step

-cov <metric_type>

-llk type <llk_type>

-aep <aep_type>

[-inject_fault <fault_type>]
```

```
-j <number_of_processes>
-slist
```

Arguments

- `-<top>`: Use this option to specify the name of the top design.
- `-library <library name>`: Use this option to remap the work library to library_name. This option can only be used with the VHDL format. By default, the analyze command stores all output in the work library. Use the -library option to store design elements in libraries other than library specified by using the -work option.
- `-define <list of verilog defines>`: Use this option to specify a list of top-level macros. This option can only be used with the Verilog and System Verilog formats.
- `-work <library name>`: Use this option to remap the work library in the same way as the -library option. It is an alias for the -library option for VHDL only. Specifies the library name to which work must be mapped. The -work option references the library where the top-level unit of the elaboration hierarchy is compiled. If the elaboration starts from a module, then it is the library name where the module is compiled. If the elaboration starts from a configuration (in case of -v2kconfig), it is the library name where the configuration is compiled.
- `-netlist`: Use this option to invoke the Verilog netlist reader. This option should only be specified if the design is in Verilog netlist format only, no behavioral syntax is present. This option cannot be used in conjunction with -vcs option.
- `-parameters <string containing ordered or named parameters separated by comma>`: Use this option to specify a list of design parameters enclosed in quotes. Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, 8,7,5) or on parameter names (for example, N=>8,M=>6). It is acceptable to mix ordered and named parameter specifications, as long as the ordered parameters are listed first.
- `-vcs <vcs command line>`: Use this option to read the design using VCS command arguments and switches. Options specified by -vcs follow the VCS command-line syntax. When specifying the VCS commands, you must include them either in â€˜{ }â€™ or double quotes.

  The [-sverilog|-verilog] argument specifies the format of the files to be analyzed.

The [-y directory_path] argument specifies directories that contain the library files to be searched for unresolved module instantiations in the design.

The [+libext+.extension1+...] argument specifies the extension to consider during a files search in the -y library directories. The default is no extension.

The [-v library_file] holds several module definitions, to be used during the search for unresolved modules.

The [-f command_file] argument specifies a command file.

The [+define+macro_name+...] argument defines a macro.

The [+incdir+dir1+...] argument includes directories in the search list.

- `-vcs_elab <vcs elaborate command line>`: Use this option to elaborate the design using VCS command arguments and switches enclosed in curly braces. Specifies all VCS-specific command-line options. Options specified by -vcs follow the VCS command-line syntax.
- `-sva`: Use this option to process SVA/PSL during compilation using 2009 semantics.
- `-sva2005`: Use this option to process SVA/PSL during compilation using 2005 semantics.
- `-v2kconfig <configuration-name>`: Use this option to specify the v2k configuration. V2k configuration cannot be directly specified as a design top.
- `-buildTop <dut name>`: Use this option to specify the DUT down from which synthesis model is generated.
- `-multi_step`: Use this option to load design in multi step mode.
- `-cov <metric_type>`: Use this option to specify types of structural coverage properties that should be instrumented as the design is compiled. These coverage properties match equivalent coverage objects in VCS in senatics and locations. The keyword all is supported to prepare for all structural coverage property types. Valid coverage property types are the following and are separated by the plus + delimiter: line, cond, toggle, fsm_state, fsm_transition. In order to control the shapes of coverage instrumentation, the standard VCS switches such as like -cm_cond, -cm_tgl etcetera can be used. These switches are passed through the -vcs switch of read_file command.

- `-llk type <llk_type>`: Use this option to create livelock goals during compilation.
- `-aep <aep_type>`: Use this option to specify types of AEP properties that should be instrumented as the design is compiled. The keyword all is supported to prepare for all structural property types. Valid structural property types are the following and are separated by the plus + delimiter:
    - ❒ bounds_check: Create a property every place in the source design where an array is indexed with a variable. The property will fail if the index can be assigned a value that is outside the bounds of the declared array. The type attribute is set to bounds_check.
    - ❒ multi_driver: Create a check for signals that are driven by multiple drivers. The property will fail if the signal is simultaneously driven by more than 1 signal. The type attribute is set to multi_driver.
    - ❒ conflict_driver: Create a check for signals that are driven by multiple drivers. The property will fail if the signal is simultaneously driven by more than 1 signal to opposite logic values. The type attribute is set to conflict_driver.
    - ❒ floating_bus: Create a check for signals that are driven by multiple drivers. The property will fail if the signal is not driven by at least 1 driver at all times. The type attribute is set to floating_bus.
    - ❒ x_assign: Create a check for every statement in the rtl where an explicit logic X assignment is made. The property will fail if the assignment is ever activated. The type attribute is set to x_assign.
    - ❒ arith_oflow: Find all occurrences of arithmetic operators in the design (+, -, *, /). The property will fail if the expression ever over- or underflows the legal ranges for the operation being executed. The type attribute is set to arith_oflow.
    - ❒ set_reset: Find all sequentials in the design which use both set and reset. The property fails if the set and reset signals are ever asserted at the same time. The type attribute is set to set_reset.
    - ❒ parallel_case: Find all occurrences of the parallel case directives in the code. These can be either: // synopsys parallel_case pragma, // synthesis parallel_case pragma, or SystemVerilog unique keyword case statements. For each of those scenarios, the property will fail if the design can behave in a way such that the pragma does not hold. The type attribute is set to parallel_case.

- `[ -inject_fault <fault_type>]`: Use this option to inject behavioral faults in the design for formal testbench analyzer flow.
- `-j <number_of_processes>`: Use this option to specify the number of processes that can be used for parallel compilation in the RTL flow. Value >= 1.
- `-slist`: Use this option to specify list of input files.

Examples

The following example reads in a verilog file named top.v in verilog netlist format. The design top is named top.

```
read_file -format verilog -top top top.v -netlist
```

read_sdc

Description

This command helps you to read an already existing SDC file as an input. SDC commands can be specified as Monet shell commands using this command. (Currently SDC commands are read in Monet shell). Also, this command is used to read User specified SDC file to populate SDC data model.

Syntax

```
read_sdc

-version_sdc_version

-module <list_of_modules>

-instance <list_of_instances>

-spec_file
```

Arguments

- `-version_sdc_version`: Use this option to specify the SDC version value.
- `-module <list_of_modules>`: Use this option to specify a list of modules for which the SDC needs to be applied.

- -instance <list_of_instances>: Use this option to specify a list of instances for which the SDC needs to be applied.
- -spec_file: Use this option to specify the SDC file to be read in.

remove_case_analysis

Description

This command removes the effect of setting value constraints from the specified list of ports/pins/nets. Upon completion, it returns 1 for success and 0 otherwise.

Syntax

```
remove_case_analysis
-all
-list
```

Arguments

- -all: Use this option to remove all case_analysis.
- -list: Use this option to specify a list of port/pin/net objects.

Examples

Here effect of constant value specified on â€œin1â€?port will get removed. While for â€œin2â€?it will remains effective. If remove_case_analysis â€“all is used then effect on all ports will get removed. i.e in this case on both â€œÃ¯n1â€?and â€œin2â€?

```
vc_static_shell>set_case_analysis 1â€™b1 {in1 in2}
vc_static_shell>remove_case_analysis in1
```

remove_clock

Description

Removes one or more clocks from the current design.

To display information about clocks and generated clocks in the design, use the report_clock command.

Syntax

```
remove_clock
```

```
-all
```

```
-clock_list
```

Arguments

- `-all`: Specifies to remove all clocks in the current design.
- `-clock_list`: Specifies a list of collections containing clocks or patterns matching the clock names.

Examples

The following example removes clock CLK1.

```
vc_static_shell>remove_clock CLK1
```

The following example removes all clocks from current design.

```
vc_static_shell>remove_clock -all
```

remove_clock_groups

Description

Removes specific exclusive or asynchronous clock groups from the current design.

Syntax

```
remove_clock_groups
```

```
[-logically_exclusive]
```

```
[-physically_exclusive]
```

```
[-exclusive]
```

```
[-asynchronous]
```

```
-name
```

```
-all
```

Arguments

- `[ -logically_exclusive]`: Specifies that the groups set for logically exclusive clocks are to be removed. The -physically_exclusive, -logically_exclusive, and -asynchronous options are mutually exclusive. You must choose only one.
- `[ -physically_exclusive]`: Specifies that the groups set for physically exclusive clocks are to be removed. The -physically_exclusive, -logically_exclusive, and -asynchronous options are mutually exclusive. You must choose only one.
- `[ -exclusive]`: Specifies that the groups set for logically exclusive clocks are to be removed.
- `[ -asynchronous]`: Specifies that groups set for asynchronous clocks are to be removed. The -physically_exclusive, -logically_exclusive, and -asynchronous options are mutually exclusive. You must choose only one.
- `-name`: Specifies a list of clock groups to be removed, which matches the groups in the given names. These clock groups are predefined by the set_clock_groups command. The -name and -all options are mutually exclusive.
- `-all`: Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The -name and -all options are mutually exclusive.

Examples

The following example removes logically_exclusive clock group named GRP1.

```
vc_static_shell>remove_clock_groups -logically_exclusive -name
GRP1
```

The following example removes all asynchronous clock groups from current design.

```
vc_static_shell>remove_clock_groups -asynchronous -all
```

remove_generated_clocks

Description

Removes generated clock objects from the current design.

To display information about clocks and generated clocks in the design, use the report_clock command.

Syntax

```
remove_generated_clocks
-all
-generated_clk_name
```

Arguments

- `-all`: Indicates that all generated clocks are to be removed.
- `-generated_clk_name`: Specifies a list of names of generated clocks to be removed.

Examples

The following example removes generated clock GEN1.

```
vc_static_shell>remove_generated_clocks GEN1
```

The following example removes all generated clocks from current design.

```
vc_static_shell>remove_generated_clocks -all
```

rename_tag

Description

This command renames a violation tag.

Syntax

```
rename_tag <tag> <alias>
```

Arguments

<tag>

Use this option to specify existing tag name.

<alias>

Use this option to specify new tag name and replace the existing tag name.

Examples

```
vcst_shell> rename_tag ISO_INST_MISSING MY_NEW_NAME
vcst_shell> report_lp
----------------------------------------------
Tree Summary
----------------------------------------------
Severity Stage   Tag                   Count
-------- ----- --------------------- -----
error    Design  MY_NEW_NAME             3
warning  UPF     ISO_STRATEGY_REDUND     1
warning  Design  ISO_STRATEGY_UNUSED     4
-------- ----- --------------------- -----
Total                                    8
```

report_mode

Description

This command defines active/inactive modes of dbcell instances.This command reports which cells have modes and for each mode the current status and condition causing current status. Also, the report reflects the mode specifications of set_mode command and set_case_analysis.

Syntax

```
report_mode
```

```
-instance_objects
```

Arguments

- `-instance_objects`: Use this option to specify a list of dbcell instances. Also, it specifies that the mode report is to include only the specified list of cell instances. By default, all cell instances that have nodes are reported.

report_names

Description

This command reports potential name changes of ports, cells, and nets in a design.

Syntax

```
report_names
[-hierarchy]
[-rules <name_rules>]
```

Arguments

- `[ -hierarchy]`: Use this option to apply change_names to the hierarchy.
- `[ -rules <name_rules>]`: Use this option to specify rules to be used for report_names.

report_properties

Description

This command reports properties for selected object.

Syntax

```
report_properties
[-instance <instance>]
[-port <port>]
[-pin <pin>]
[-net <net>]
```

Arguments

- `[ -instance <instance>]`: Use this option to specify design instance name.

- `[ -port <port>]`: Use this option to specify design port name.
- `[ -pin <pin>]`: Use this option to specify design pin name.
- `[ -net <net>]`: Use this option to specify design net name.

report_read

Description

This command reports the messages issued in reading the design. This command is used to dump the messages issued in design load. The command report_read (â€“family hdl) will work on the messages issued during VCS flow, such as parsing, design resolution, and hardware inference.

Syntax

```
report_read
[-family <{design_read_family_list}>]
[-list]
[-verbose]
[-no_summary]
[-tag_type <builtin | vcst | legacy>]
```

Arguments

- `[ -family <{design_read_family_list}>]`: Use this option to provide a list of allowable family: Values: all, hdl, netlist, sdc, upf.
- `[ -list]`: Use this option to list all messages in simple form.
- `[ -verbose]`: Use this option to display short description for each message.
- `[ -no_summary]`: Use this option to suppress summary information.
- `[ -tag_type <builtin | vcst | legacy>]`: Use this option to specify the tag naming convention: Values: builtin, legacy, vcst.

report_read_violations

Description

Use this command to get reports of the design read, upf parsing, sdc read, and Tcl command violations which were identified. By default, it writes a summary of the messages. The `-verbose` option is required to write the details of each message. By default, only the first 100 messages of each tag are printed. To print more, use the `-limit` flag.

Syntax

```
report_read_violations
[-no_summary]
[-list]
[-verbose]
[-limit <count>]
[-include_waived]
[-only_waived]
[-all_tags]
[-tag <list of tag names>]
[-waived <list of waiver names>]
[-id <list of message identifiers>]
[-family <list of family name>]
[-severity <list of message severities>]
[-filter <expression>]
[-regexp]
[-nocase]
[-file <file name>]
[-append]
```

Arguments

- `[ -no_summary]`: Suppresses the two summary tables which list the number of violations in each family and stage.

- `[ -list]`: In addition to the summary tables, print a one sentence description of each violation with the design data fields filled in. Useful for generating a file with one line per violation.
- `[ -verbose]`: In addition to the summary tables, print a number of lines of detail about each violation. This verbose format includes the description, basic design detail fields for the violation, and also detailed debugging fields for the violation. Useful for getting all details of the violation.
- `[ -limit <count>]`: When used with list or verbose mode, print only this number of violations for each tag. Useful to limit the file size for designs with a large violation count.
- `[ -include_waived]`: By default, any violation which is waived is not included in the report. Use this switch to include the waived messages in the report.
- `[ -only_waived]`: By default, any violation which is waived is not included in the report. Use this switch to invert the display so that only waived messages are included in the report.
- `[ -all_tags]`: Included all the tags checked for in the report.
- `[ -tag <list of tag names>]`: To focus on only certain tags, use this switch with a list of tag names. Only violations whose tag is on this list will be displayed
- `[ -waived <list of waiver names>]`: To focus on only certain waivers, use this switch with a list of waiver names. Only violations which are waived by a waiver on this list will be displayed.
- `[ -id <list of message identifiers>]`: To focus on only one message or a short list of messages, use this switch with a list of message identifiers such as SDC:123, UPF:23. Only these violations will be displayed.
- `[ -family <list of family name>]`: To focus on only messages from certain families, use this switch with a list of families. The valid families are: all, design, sdc, upf.
- `[ -severity <list of message severities>]`: To focus on only messages with a certain severity, use this switch with a list of severities. The valid severities are: error, info, warning.
- `[ -filter <expression>]`: This switch allows you to specify complex criteria based on pattern matching. Only violations matching the filter

expression will be shown. An expression may contain several terms separated with a double ampersand. Each term has a field name, a comparison operator, and a target string. The field name may be any field name shown in the verbose report; for a field inside a record, use a colon to separate the record path components. The comparison operator is any of the standard operators such as == != =~. See the examples section for several examples.

- `[ -regexp]`: Use this switch to indicate that the filter expression type is a regular expression. The default is glob-style.
- `[ -nocase]`: Use this switch to indicate that the filter expression ignoring the case when matching string values.
- `[ -file <file name>]`: Write the results to the designated file.
- `[ -append]`: Append results to the designated file.

report_session_data

Description

This command displays session-specific information for the current and restored runtime database.Upon completion, this command returns 1 for success, 0 otherwise.

Syntax

```
report_session_data
[-commands]
```

Arguments

- `[ -commands]`: Use this option to display the log file generated during the saved session.

report_tag

Description

This command prints a report about a set of tags. For example, to get a report of all the isolation tags and information about them, use:

```
vc_static_shell> report_tag -family isolation
```

The command has several arguments to control the set of tags which are printed, and also allows control over which attributes are printed. If requested, the report can be printed in "comma separated value" format for easy import into a spreadsheet.

Syntax

```
report_tag
[-app <app>]
[-severity <severity>]
[-tag <pattern>]
[-stage <pattern>]
[-family <pattern>]
[-enabled]
[-disabled]
[-builtin]
[-user]
[-csv]
[-order <attributes>]
```

Arguments

- `[ -app <app>]`: Use this option to specify application name.
- `[ -severity <severity>]`: Use this option to print only tags with this severity.
- `[ -tag <pattern>]`: Use this option to print only tags with name matching pattern.
- `[ -stage <pattern>]`: Use this option to print only tags with stage matching pattern.
- `[ -family <pattern>]`: Use this option to print only tags with family matching pattern.
- `[ -enabled]`: Use this option to print only enabled tags.

- `[ -disabled]`: Use this option to print only disabled tags.
- `[ -builtin]`: Use this option to print only builtin tags.
- `[ -user]`: Use this option to print only user defined tags.
- `[ -csv]`: Use this option to print in "comma separated value" format.
- `[ -order <attributes>]`: Use this option to specify Tcl list of attributes in desired print order.

Examples

Report all the tags, and all the information about them to file tag.txt

```
report_tag > tag.txt
```

Report all the disabled, built-in tags

```
report_tag -disabled -builtin
```

Report only the severity attribute, followed by description, and use comma separated value format

```
report_tag -csv -order {description severity}
```

reset_mode

Description

This command resets active modes of dbcell instances to default. This is the default behavior when no modes are specified with the set_mode command.

Syntax

```
reset_mode
```

```
-instance_objects
```

Arguments

- `-instance_objects`: Use this option to specify a list of dbcell instances.

set_case_analysis

Description

This command is used to specify a logic value (treat it as temporary constant) on pins or ports. This command performs analysis assuming this constant value at this port/pin. You can use case analysis settings to place the design into a given operating mode without altering the netlist. Then these values are propegated based on the requirements of applications.

You can use remove_case_analysis to remove added case values.

Syntax

```
set_case_analysis

-value

-object_list
```

Arguments

- `-value`: Use this option to specify a constant value to be set. Either 1 or 0.
- `-object_list`: Use this option to specify a list of port/pin/net objects.

Examples

The following command sets the IN1 port to constant logic 0.

```
set_case_analysis 0 IN1
```

waive_read

Description

Provides ability to waive based on Tags, Family, Severity, Filter rules using debug fields, Wildcards and expressions and add comment for waivers as well.

NOTE: *Use waive_read command before report_read_violations command.*

Syntax

```
waive_read
```

```
[-add <name>]
[-append <name>]
[-comment <comment>]
[-delete <name(s)>]
[-delete_all]
[-tcl]
[-force]
[-tag <tag>]
[-id <tag>]
[-stage <stage>]
[-family <family>]
[-severity <list>]
[-filter <expression>]
[-regexp]
[-nocase]
```

Arguments

- `[ -add <name>]`: Add waiver
- `[ -append <name>]`: Append additional filter parameters to an existing waiver
- `[ -comment <comment>]`: Waiver Comment.
- `[ -delete <name(s)>]`: Delete waiver
- `[ -delete_all]`: Delete all waivers
- `[ -tcl]`: Display the waiver list in TCL command format
- `[ -force]`: Create a container for waive_read append operations.
- `[ -tag <tag>]`: Waive violations based on tag
- `[ -id <tag>]`: Waive violations based on IDs
- `[ -stage <stage>]`: Waive violations based on stage: Values: design, pg, upf

- `[ -family <family>]`: Waive violations based on family: Values: all, sdc, upf, design
- `[ -severity <list>]`: Waive violations based on severity: Values: all, error, info, warning
- `[ -filter <expression>]`: Waive violations based on expression
- `[ -regexp]`: Indicates filter expression type to be regular expression (default glob-style).
- `[ -nocase]`: Filter expressions ignore case when matching string value.

Examples

The following example shows some usage of waive_read command

```
vc_static_shell> waive_read -family upf -add waiver1 -comment
{ask hemang} -tag SM*
```

Command Sanity Checks

This section describes the VC SpyGlass sanity checks on Lint Tcl commands.

VC SpyGlass Lint provides the following tags that report sanity errors in the Tcl commands. Use the *report_read_violations* command to report the violations of these tags.

Errors Generated by Tcl Commands

VC SpyGlass reports error generated by Tcl commands in the `session.log` file as well as the GUI.

For example, if you have specified a pattern in the `get_cells` command and VC SpyGlass cannot find the corresponding cell in the design, it reports the following DB_QUERY_PATTERN_NO_MATCH violation in the session log as well as the GUI:

```
get_cells Cell*
```

```
case1.tcl:40: [Warning] DB_QUERY_PATTERN_NO_MATCH:
No matching cell to the given pattern 'Cell*'
```

TCL_COMMAND_INVALID

Severity

Error

Short Help

Reports invalid values of Tcl commands

Description

Reports invalid values specified with a Tcl command argument. In this case, VC SpyGlass CDC ignores the specified command. The tag reports the following message:

```
Value <OptionValue> specified for option <OptionName> of
<CommandName> command is invalid. Command is ignored. [Reason:
<Reason>.].
```

Example

Consider the following specification:

```
set_constraints_scope -module M1
```

In this case, the TCL_COMMAND_INVALID reports a violation if the `M1` module does not exist in the design.

What Next

Review the invalid value of the argument and specify the correct value.

TCL_PREREQUISITE_COMMAND_NOT_FOUND

Severity

Error

Short Help

Reports if pre-requisite commands do not exist

Description

Reports a violation if any mandatory pre-requisite command for the specified command does not exist. In this case, VC SpyGlass CDC ignores the specified command. The tag reports the following message:

```
Prerequisite command(s) <CommandName> does not exist for
command <CommandName>. Command is ignored.
```

Example

Consider the following specification:

```
define_attribute -name ATTR1
```

In this case, the `set_constraints_scope` command, which is a pre-requisite for the `define_attribute` command, is not specified. Therefore, the TCL_PREREQUISITE_COMMAND_NOT_FOUND tag reports a violation because the pre-requisite command is not specified.

What Next

Review the violation and ensure that the pre-requisite commands are specified.

TCL_COMMAND_OPTION_VALUE_INVALID

Severity

Warning

Short Help

Reports invalid values specified with an argument of a Tcl command

Description

This tag reports a violation if a value specified with an argument of a Tcl command is invalid. In this case, VC SpyGlass considers only the other valid values and ignores the invalid value.

The tag reports the following message:

```
Value <s> specified for option <s> of command <s> is invalid.
Option is ignored. [Reason: <Reason>].
```

Example

Consider the following specification:

```
apply_attribute ATTR1 -objects { OUT_PORT_1  IN_PORT_1} -start
```

In this case, `OUT_PORT_1` is an invalid value of the `-objects` argument if the `-start` argument is specified in the `apply_attribute` command. In this case, the TCL_COMMAND_OPTION_VALUE_INVALID tag reports a warning and considers only the `IN_PORT_1` value and ignores the `OUT_PORT_1` value.

What Next

Review the invalid value of the argument and specify the correct value.

TCL_OBJECT_NOT_FOUND

Severity

Error

Short Help

Reports invalid objects specified in Tcl commands

Description

Reports invalid objects specified with a Tcl command argument. In this case, VC SpyGlass CDC ignores the specified command. The tag reports the following message:

```
No matching object found for <object-name> in <OptionName>
option of <CommandName> command
```

Example

Consider the following specification:

```
configure_unconstrained_ports -module M1
```

In this case, the TCL_OBJECT_NOT_FOUND reports a violation if the `M1` module does not exist in the design.

What Next

Review the invalid value of the argument and specify the correct value.

TCL_COMMAND_CONFLICTING

Severity

Error

Short Help

Reports conflicting Tcl commands

Description

Reports a violation if conflicting commands are specified. In this case, VC SpyGlass CDC ignores the command that is specified later. The tag reports the following message:

```
Command <CommandName2> conflicts with command <CommandName1>.
Command is ignored
```

Example

Consider the following specifications:

```
set_constraints_scope -module M1

define_attribute -name ATTR1

set_connectivity_attribute ATTR1 -related_ports {P1}

set_clock_attribute ATTR1 -clock_objects {C1}

end_constraints_scope
```

In this case, the TCL_COMMAND_CONFLICTING tag reports a violation because both the `set_connectivity_attribute` and the `set_clock_attribute` commands are specified with the same attribute name and in the same scope. Therefore, the `set_clock_attribute` command is ignored.

What Next

Review and correct the conflicting commands.

Migrating Waivers

This section describes migrating waivers in VC SpyGlass Lint and includes the following sections:

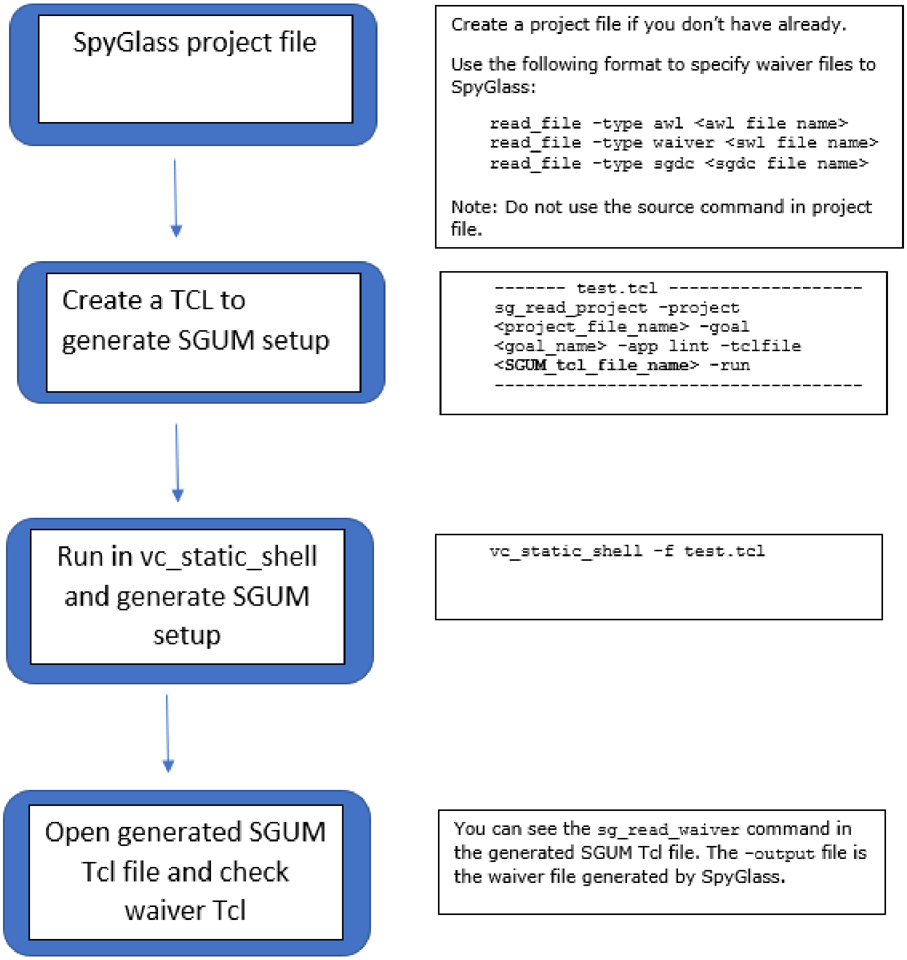
- *Working with Waivers*
- *Migrating Waivers*
- *Waiving Using GUI*
- *Reporting Waivers*
- *Waiver-related Commands and Application Variables*

Working with Waivers

You can migrate waivers from SpyGlass to the SGUM and VCUM in VC SpyGlass.

SpyGlass to SGUM Migration Flow

The following describes the SpyGlass to SGUM migration flow.


SpyGlass project file
Create a TCL to generate SGUM setup
Run in vc_static_shell and generate SGUM setup
Open generated SGUM Tcl file and check waiver Tcl

Create a project file if you don't have already.

Use the following format to specify waiver files to SpyGlass:

```
    read_file -type awl <awl file name>
    read_file -type waiver <swl file name>
    read_file -type sgdc <sgdc file name>
```

Note: Do not use the source command in project file.

```
     ------- test.tcl -------------------
     sg_read_project -project
     <project_file_name> -goal
     <goal_name> -app lint -tclfile
     <SGUM_tcl_file_name> -run
     ------------------------------------
```

```
    vc_static_shell -f test.tcl
```

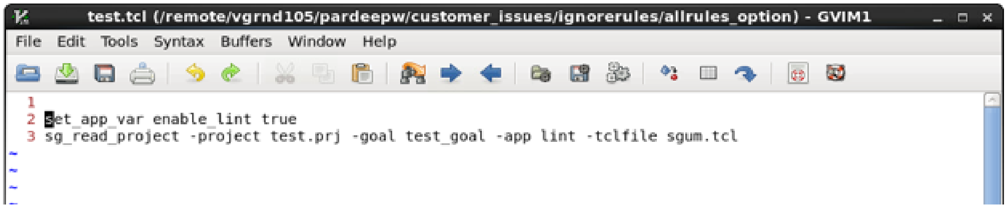
You can see the sg_read_waiver command in the generated SGUM Tcl file. The -output file is the waiver file generated by SpyGlass.

Keep the following guidelines in mind while migrating SpyGlass waivers to SGUM:

- If you are sg_shell user, you need to create a spyglass project file.
- Use the following format to specify waiver files to SpyGlass.

```
read_file -type awl <awl file name>

read_file -type waiver <swl file name>

read_file -type sgdc <sgdc file name>
```

The following shows a sample test.tcl file.


test.tcl (/remote/vgrnd105/pardeepw/customer_issues/ignorerules/allrules_option) - GVIM1
File
Edit
Tools
Syntax
Buffers
Window
Help

```

set app_var enable_lint true
sg_read_project -project test.prj -goal test_goal -app lint -tclfile sgum.tcl
```

NOTE: *Do not use source command in project file. The source waiver.tcl command results in an error.*

- Create a tcl file by using the following command to generate SGUM setup for the SpyGlass project file & goal.

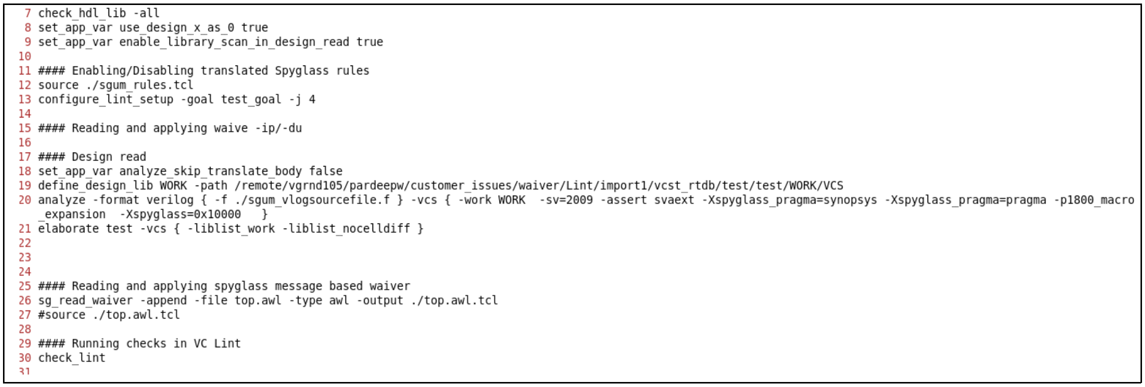
------------------------- test.tcl ---

```
sg_read_project -project <project_file_name> -goal <goal_name>
-app lint -tclfile <SGUM_tcl_file_name> -run -no_rule_gen -
rename_path -tag_path -post_process_report
```

NOTE: *If you do not use the -run command, you will not be able to generate full setup for SGUM run. Without the -run option, waiver tcl will be generated but it is not complete one. Waivers with -msg (message based waivers) will not be translated into tcl file because it is needed to run the design one time. Therefore, you need to perform a full SGUM run to generate the complete waiver tcl file. Once you generated the complete waiver file, you can delete or comment out the sg_read_waivers line & use for next time.*

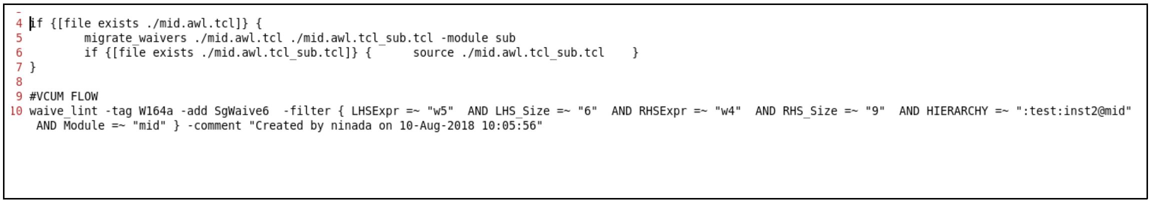
If you ave specified a waiver file in the .prj (`read_file -type awl/ waiver`), the awl or swl command will automatically convert to a tcl file by the `sg_read_waiver` command which you can see in the standard output:

```
sg_read_waiver -append -file top.awl -type awl -output ./
top.awl.tcl
```



```
check_hdl_lib -all
set_app_var use_design_x_as_0 true
set_app_var enable_library_scan_in_design_read true

#### Enabling/Disabling translated Spyglass rules
source ./sgum_rules.tcl
configure_lint_setup -goal test_goal -j 4

#### Reading and applying waive -ip/-du

#### Design read
set_app_var analyze_skip_translate_body false
define_design_lib WORK -path /remote/vgrnd105/pardeepw/customer_issues/waiver/Lint/import1/vcst_rtdb/test/test/WORK/VCS
analyze -format verilog { -f ./sgum_vlogsourcefile.f } -vcs { -work WORK  -sv=2009 -assert svaext -Xspyglass_pragma=synopsys -Xspyglass_pragma=pragma -p1800_macro
_expansion  -Xspyglass=0x10000   }
elaborate test -vcs { -liblist_work -liblist_nocelldiff }



#### Reading and applying spyglass message based waiver
sg_read_waiver -append -file top.awl -type awl -output ./top.awl.tcl
#source ./top.awl.tcl

#### Running checks in VC Lint
check_lint

```

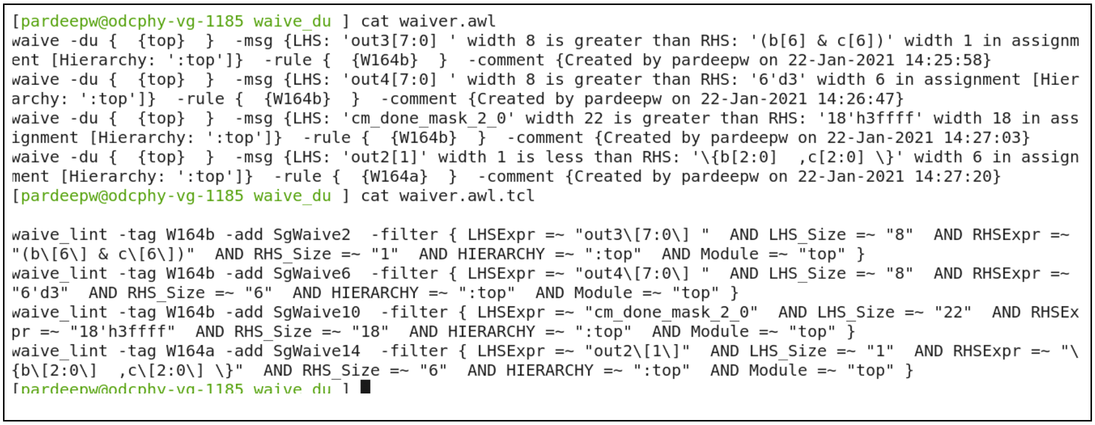
- If you have hierarchical waivers (`waive -import`), these waivers are converted to `migrate_waivers` command that you can see in the output file of waiver tcl as shown in the following sample.



```
if {[file exists ./mid.awl.tcl]} {
        migrate_waivers ./mid.awl.tcl ./mid.awl.tcl_sub.tcl -module sub
        if {[file exists ./mid.awl.tcl_sub.tcl]} {      source ./mid.awl.tcl_sub.tcl    }
}

#VCUM FLOW
waive_lint -tag W164a -add SgWaive6  -filter { LHSExpr =~ "w5"  AND LHS_Size =~ "6"  AND RHSExpr =~ "w4"  AND RHS_Size =~ "9"  AND HIERARCHY =~ ":test:inst2@mid"
 AND Module =~ "mid" } -comment "Created by ninada on 10-Aug-2018 10:05:56"
```

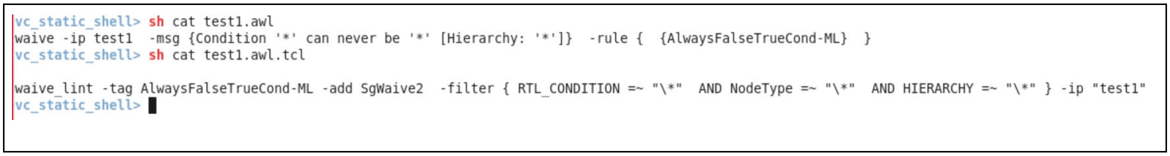
In this case, the `./mid.awl.tcl` file is the block-level waiver file and the `./mid.awl.tcl_sub.tcl` file is the top-level waiver file.

- If you have any waivers with `waive -du`, it will convert to `waive_lint -filter {module =~ }` as shown in the following sample file:



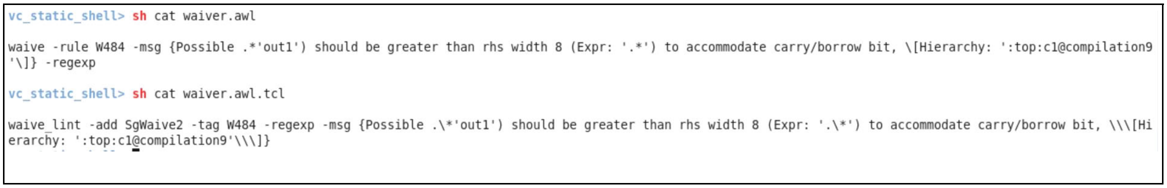
```
[pardeepw@odcphy-vg-1185 waive_du ] cat waiver.awl
waive -du {  {top}  }  -msg {LHS: 'out3[7:0] ' width 8 is greater than RHS: '(b[6] & c[6])' width 1 in assignm
ent [Hierarchy: ':top']}  -rule {  {W164b}  }  -comment {Created by pardeepw on 22-Jan-2021 14:25:58}
waive -du {  {top}  }  -msg {LHS: 'out4[7:0] ' width 8 is greater than RHS: '6'd3' width 6 in assignment [Hier
archy: ':top']}  -rule {  {W164b}  }  -comment {Created by pardeepw on 22-Jan-2021 14:26:47}
waive -du {  {top}  }  -msg {LHS: 'cm_done_mask_2_0' width 22 is greater than RHS: '18'h3ffff' width 18 in ass
ignment [Hierarchy: ':top']}  -rule {  {W164b}  }  -comment {Created by pardeepw on 22-Jan-2021 14:27:03}
waive -du {  {top}  }  -msg {LHS: 'out2[1]' width 1 is less than RHS: '\{b[2:0]  ,c[2:0] \}' width 6 in assign
ment [Hierarchy: ':top']}  -rule {  {W164a}  }  -comment {Created by pardeepw on 22-Jan-2021 14:27:20}
[pardeepw@odcphy-vg-1185 waive_du ] cat waiver.awl.tcl

waive_lint -tag W164b -add SgWaive2  -filter { LHSExpr =~ "out3\[7:0\] "  AND LHS_Size =~ "8"  AND RHSExpr =~
"(b\[6\] & c\[6\])"  AND RHS_Size =~ "1"  AND HIERARCHY =~ ":top"  AND Module =~ "top" }
waive_lint -tag W164b -add SgWaive6  -filter { LHSExpr =~ "out4\[7:0\] "  AND LHS_Size =~ "8"  AND RHSExpr =~
"6'd3"  AND RHS_Size =~ "6"  AND HIERARCHY =~ ":top"  AND Module =~ "top" }
waive_lint -tag W164b -add SgWaive10  -filter { LHSExpr =~ "cm_done_mask_2_0"  AND LHS_Size =~ "22"  AND RHSEx
pr =~ "18'h3ffff"  AND RHS_Size =~ "18"  AND HIERARCHY =~ ":top"  AND Module =~ "top" }
waive_lint -tag W164a -add SgWaive14  -filter { LHSExpr =~ "out2\[1\]"  AND LHS_Size =~ "1"  AND RHSExpr =~ "\
{b\[2:0\]  ,c\[2:0\] \}"  AND RHS_Size =~ "6"  AND HIERARCHY =~ ":top"  AND Module =~ "top" }
[pardeepw@odcphy-vg-1185 waive_du ] █
```

- If you have any waivers with `waive -ip`, it will convert to `waive_lint -ip`, as shown in the following sample file:



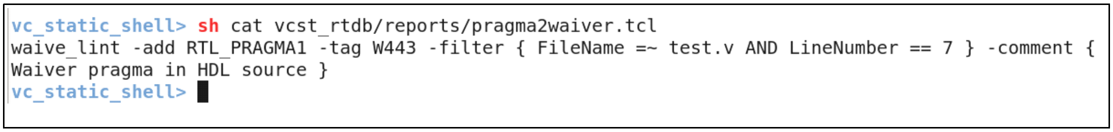
```
vc_static_shell> sh cat test1.awl
waive -ip test1  -msg {Condition '*' can never be '*' [Hierarchy: '*']}  -rule {  {AlwaysFalseTrueCond-ML}  }
vc_static_shell> sh cat test1.awl.tcl

waive_lint -tag AlwaysFalseTrueCond-ML -add SgWaive2  -filter { RTL_CONDITION =~ "\*"  AND NodeType =~ "\*"  AND HIERARCHY =~ "\*" } -ip "test1"
vc_static_shell> █
```

- Waiver with `-regexp` (Partial static waivers)



```
vc_static_shell> sh cat waiver.awl

waive -rule W484 -msg {Possible .*'out1') should be greater than rhs width 8 (Expr: '.*') to accommodate carry/borrow bit, \[Hierarchy: ':top:c1@compilation9
'\]} -regexp

vc_static_shell> sh cat waiver.awl.tcl

waive_lint -add SgWaive2 -tag W484 -regexp -msg {Possible .\*'out1') should be greater than rhs width 8 (Expr: '.\*') to accommodate carry/borrow bit, \\\[Hi
erarchy: ':top:c1@compilation9'\\\]}
```

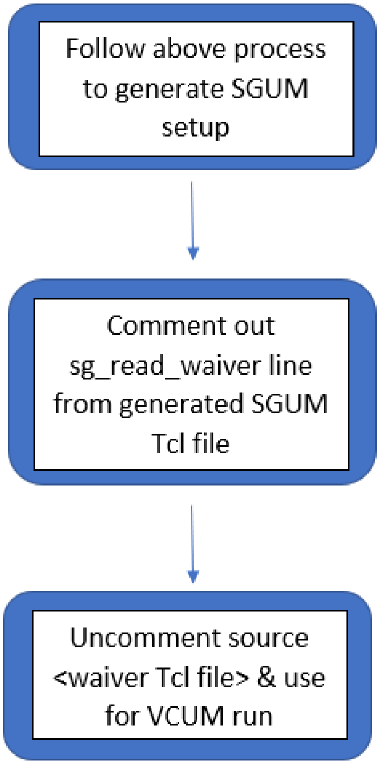
- Pragma-based waivers are generated and applied automatically by tool. The `enable_pragma_based_waiver` application variable is set to enabled by default. The waiver file is generated in the `vcst_rtdb/reports/pragma2waiver.tcl` directory.



```
vc_static_shell> sh cat vcst_rtdb/reports/pragma2waiver.tcl
waive_lint -add RTL_PRAGMA1 -tag W443 -filter { FileName =~ test.v AND LineNumber == 7 } -comment {
Waiver pragma in HDL source }
vc_static_shell> █
```

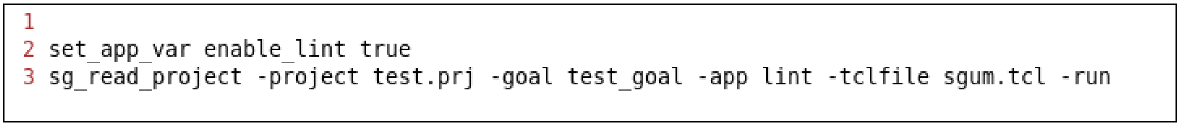
SpyGlass to VCUM Migration Flow

The following describes the SpyGlass to VCUM migration flow.


Follow above process to generate SGUM setup
Comment out sg_read_waiver line from generated SGUM Tcl file
Uncomment source <waiver Tcl file> & use for VCUM run

You need to first run SGUM and generate the setup and invoke the setup into VC SpyGlass.

After the SGUM run completes, the sgum.tcl file is created as shown in the following figure.



```

set_app_var enable_lint true
sg_read_project -project test.prj -goal test_goal -app lint -tclfile sgum.tcl -run
```

Next, you can invoke `sgum.tcl` in vc_static_shell.

Migrating Waivers

In VC SpyGlass, the `migrate_waivers` Tcl command enables you to migrate block-level waivers to the top level.

The command uses the following syntax.

```
migrate_waivers <block-level-waiver-file> <top-level-waiver
file-name-to-be-created> -module <mod-name>/-instance <inst-
name>

-module <module_name> Specify the block module.

-instance <instance_name> Specify the instance of module.
```

For example, consider the following commands:

```
vc_static_shell> migrate_waivers block_waive.tcl top_waive.tcl
-instance ublock

source top_waive.tcl


vc_static_shell> migrate_waivers block_waive.tcl top_waive.tcl
-module block

source top_waive.tcl
```

If you need to migrate waivers of multiple blocks to the top level, specify separate `migrate_waiver` command for each block as shown in the following sample top-level waiver file.

```
migrate_waiver block1.tcl top1.tcl -module block1

migrate_waiver block2.tcl top2.tcl -module block2

migrate_waiver block3.tcl top3.tcl -module block3

source top1.tcl

source top2.tcl

source top3.tcl
```

Therefore, if there are N number of blocks, an equivalent number of top-level waiver files need to be generated.

Applying Waivers

You can apply the waivers by using any of the following two methods:

Method #1

By using the `manage_waiver_file` command. The command uses the following syntax:

```
manage_waiver_file <file_name>

-add

-remove
```

Arguments

- `<file_name>`: Use this option to specify the waiver file name.
- `-add`: Use this option to add waiver file.
- `-remove`: Use this option to remove waiver file.

Examples

Specify the following commands in the Tcl script to add three files and define the `new1.tcl` file as default waiver file.

```
manage_waiver_file -add waiver2.tcl

manage_waiver_file -add waiver4.tcl

manage_waiver_file -add new1.tcl

set_app_var default_waiver_file new1.tcl
```

You can also specify multiple files in a single command as follows:

```
manage_waiver_file -add { waiver1.tcl waiver2.tcl }

manage_waiver_file -remove { waiver1.tcl waiver2.tcl }
```

Method #2

You can source the waiver.tcl file as follows:

```
Source waiver.tcl
```

Migrating Pragma Waivers

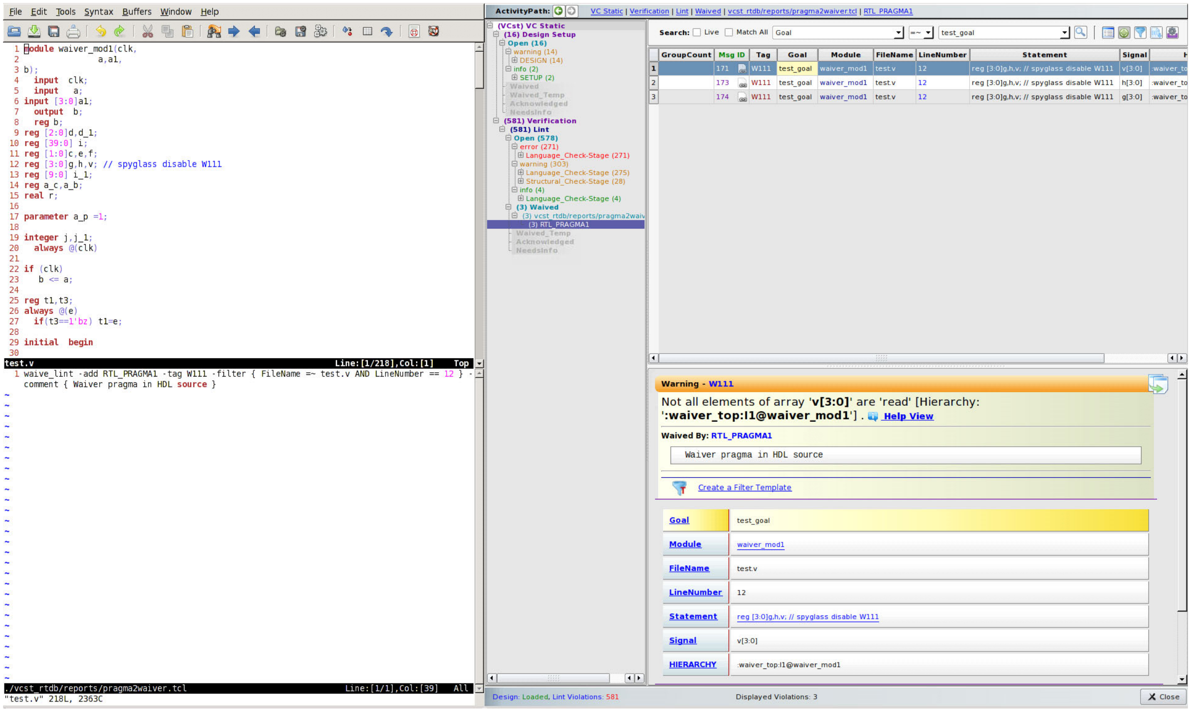
VC SpyGlass enables you to specify waivers in the RTL file itself by using inline waiver or pragma waivers.

Pragma waivers can be of the following types:

- Line pragma waivers: The scope of these waivers is restricted to the line in which the pragma waiver is specified.

  Syntax: `// spyglass disable <Rule Name>`

  The following figure shows an line pragma waiver.



```
module waiver_mod1(clk,
               a,a1,
b);
  input  clk;
  input   a;
input [3:0]a1;
  output  b;
  reg b;
reg [2:0]d,d_1;
reg [39:0] i;
reg [1:0]c,e,f;
reg [3:0]g,h,v; // spyglass disable W111
reg [9:0] i_1;
reg a_c,a_b;
real r;

parameter a_p =1;

integer j,j_1;
  always @(clk)

if (clk)
   b <= a;

reg t1,t3;
always @(e)
  if(t3==1'bz) t1=e;

initial  begin

test.v                                          Line:[1/218],Col:[1]     Top
```

```
waive_lint -add RTL_PRAGMA1 -tag W111 -filter { FileName =~ test.v AND LineNumber == 12 } -
comment { Waiver pragma in HDL source }
./vcst_rtdb/reports/pragma2waiver.tcl                Line:[1/1],Col:[39]   All
"test.v" 218L, 2363C
```

	GroupCount	Msg ID	Tag	Goal	Module	FileName	LineNumber	Statement	Signal	H
1		171	W111	test_goal	waiver_mod1	test.v	12	reg [3:0]g,h,v; // spyglass disable W111	v[3:0]	:waiver_to
2		173	W111	test_goal	waiver_mod1	test.v	12	reg [3:0]g,h,v; // spyglass disable W111	h[3:0]	:waiver_to
3		174	W111	test_goal	waiver_mod1	test.v	12	reg [3:0]g,h,v; // spyglass disable W111	g[3:0]	:waiver_to

Warning - W111

Not all elements of array '**v[3:0]**' are 'read' [Hierarchy: '**:waiver_top:I1@waiver_mod1**'] . Help View

Waived By: RTL_PRAGMA1

Waiver pragma in HDL source

Goal	test_goal
Module	waiver_mod1
FileName	test.v
LineNumber	12
Statement	reg [3:0]g,h,v; // spyglass disable W111
Signal	v[3:0]
HIERARCHY	:waiver_top:I1@waiver_mod1

Design: Loaded, Lint Violations: 581

Displayed Violations: 3

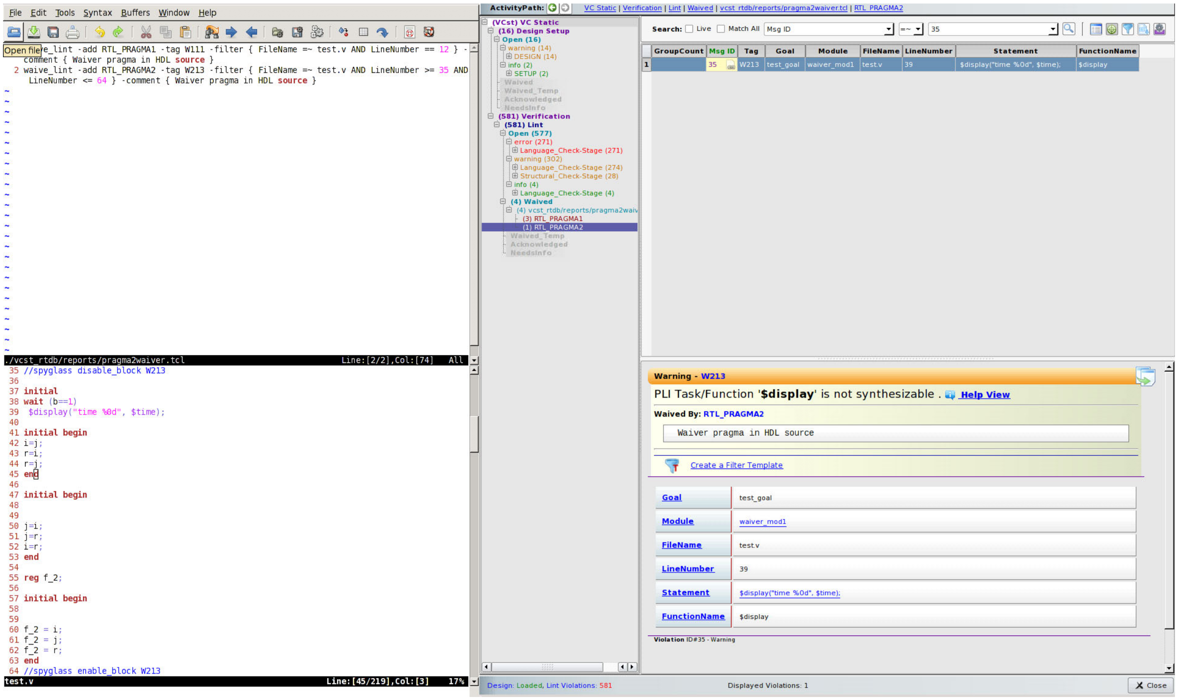
- Block pragma waivers: The scope of these waivers is restricted to the block of RTL surrounded by the pragma waivers.

  Syntax:

  ```
  Line:X :: // spyglass disble_block <Rule Name>
  Line:Y :: // spyglass enable_block <Rule Name>
  ```

`Where Y > X`

The following figure shows an line pragma waiver.



```
  waive_lint -add RTL_PRAGMA1 -tag W111 -filter { FileName =~ test.v AND LineNumber == 12 } -
  comment { Waiver pragma in HDL source }
2 waive_lint -add RTL_PRAGMA2 -tag W213 -filter { FileName =~ test.v AND LineNumber >= 35 AND
   LineNumber <= 64 } -comment { Waiver pragma in HDL source }
```

./vcst_rtdb/reports/pragma2waiver.tcl Line:[2/2],Col:[74] All

```
35 //spyglass disable_block W213
36
37 initial
38 wait (b==1)
39  $display("time %0d", $time);
40
41 initial begin
42 i=j;
43 r=i;
44 r=j;
45 end
46
47 initial begin
48
49
50 j=i;
51 j=r;
52 i=r;
53 end
54
55 reg f_2;
56
57 initial begin
58
59
60 f_2 = i;
61 f_2 = j;
62 f_2 = r;
63 end
64 //spyglass enable_block W213
```

test.v Line:[45/219],Col:[3] 17%

ActivityPath: VC Static | Verification | Lint | Waived | vcst_rtdb/reports/pragma2waiver.tcl | RTL_PRAGMA2

```
(VCst) VC Static
  (16) Design Setup
    Open (16)
      warning (14)
        DESIGN (14)
      info (2)
        SETUP (2)
    Waived
    Waived_Temp
    Acknowledged
    NeedsInfo
  (581) Verification
    (581) Lint
      Open (577)
        error (271)
          Language_Check-Stage (271)
        warning (302)
          Language_Check-Stage (274)
          Structural_Check-Stage (28)
        info (4)
          Language_Check-Stage (4)
      (4) Waived
        (4) vcst_rtdb/reports/pragma2waiv
          (3) RTL_PRAGMA1
          (1) RTL_PRAGMA2
      Waived_Temp
      Acknowledged
      NeedsInfo
```

Search: □ Live □ Match All Msg ID =~ 35

	GroupCount	Msg ID	Tag	Goal	Module	FileName	LineNumber	Statement	FunctionName
1		35	W213	test_goal	waiver_mod1	test.v	39	$display("time %0d", $time);	$display

Warning - W213

PLI Task/Function '**$display**' is not synthesizable . Help View

Waived By: RTL_PRAGMA2

Waiver pragma in HDL source

Create a Filter Template

Goal	test_goal
Module	waiver_mod1
FileName	test.v
LineNumber	39
Statement	$display("time %0d", $time);
FunctionName	$display

Violation ID#35 - Warning

Design: Loaded, Lint Violations: 581

Displayed Violations: 1

The scope of the pragma waiver starts from line X and ends at line Y. Any violation of rule <rule-name> from line X till line Y are waived off.

In case `// spyglass enable_block <Rule Name>` is not specified, the scope to the waiver is then from line X till end of file.

Use the `enable_pragma_based_waiver` application variable to manage the functioning of the pragma waiver consumption. The app var can have the following values:

- `enabled`: Pragma-based waivers will be generated and applied automatically by tool.
- `disabled`: Pragma-based waivers will not be generated by the tool.
- `create_invalid_rule_pragmas`: Pragma-based waivers for invalid rules are created and generated in a commented format by tool.

The pragma waivers created by the tool are available in the `vcst_rtdb/reports/pragma2waiver.tcl` file.

Waiving Using GUI

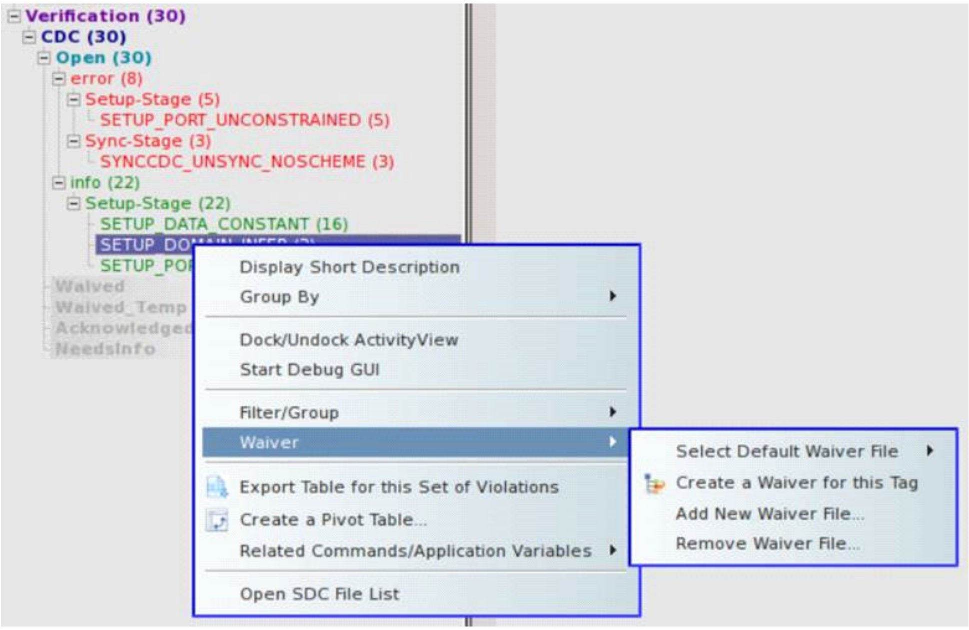
This section describes the following by using the VC SpyGlass GUI:

Managing Waivers

Creating a Waiver

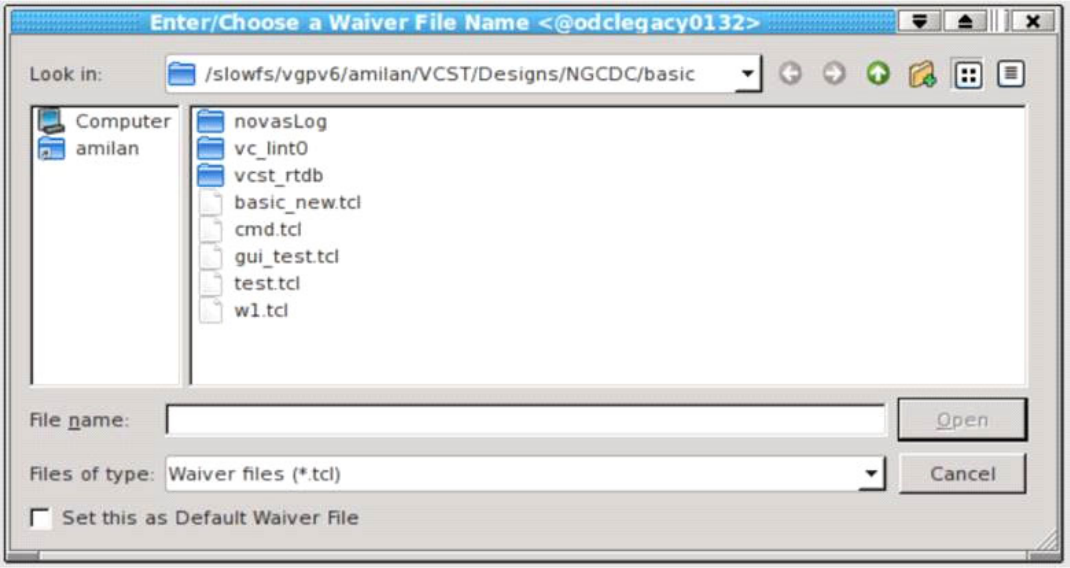
Managing Waivers

VC SpyGlass GUI provides the following waiver management options in the `Waiver` sub-menu item of the right-click menu on the `activity tree` as shown in the following figure.



The following options are available:

- **Add New Waiver File**: Use this menu item to add a waiver file. This command opens the following dialog where you can add a waiver file as well as specify the default waiver file:



Enter/Choose a Waiver File Name <@odclegacy0132>

Look in: /slowfs/vgpv6/amilan/VCST/Designs/NGCDC/basic

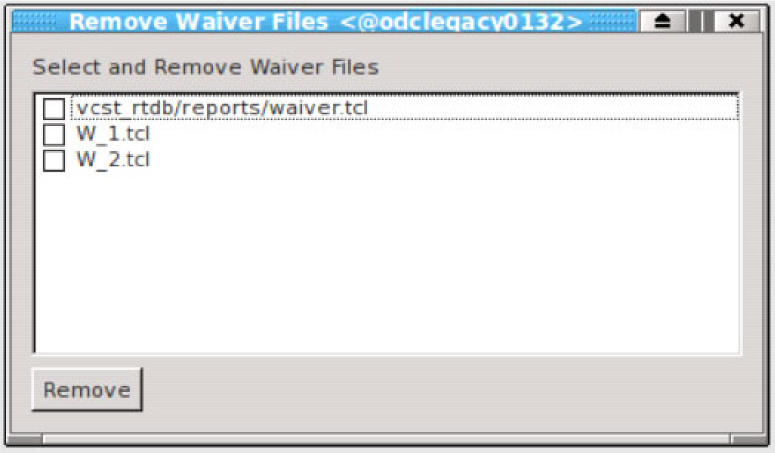
File name:

Files of type: Waiver files (*.tcl)

□ Set this as Default Waiver File

Check the `Set this as Default Waiver File` option to make the newly added waiver file as the default waiver file.

- **Remove Waiver File**: This menu item opens the following dialog and is used to remove a waiver file. Deselect the check box for the waiver file name and then click the `Remove` button.



Remove Waiver Files <@odclegacy0132>

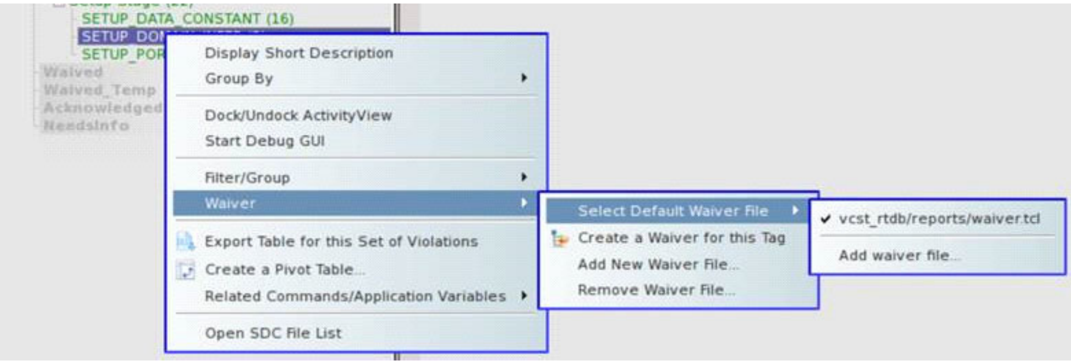
Select and Remove Waiver Files

□ vcst_rtdb/reports/waiver.tcl
□ W_1.tcl
□ W_2.tcl

Remove

- **Select Default Waiver File**: By default, the waivers are added to the default waiver file `<Working_directory>/vcst_rtdb/reports/waiver.tcl`. You can change the default waiver file by using this menu item.

  As shown in the following figure, the default waiver file mentioned above is displayed with a check sign denoting that the file is set as the default waiver file.



When more waiver files are added, the files are also listed to be selected as the default waiver file as required as shown in below capture.

You can use the `Add waiver file` menu item to add a waiver file. When you use this menu item to add a waiver file, the tool makes this added waiver file as the default waiver file. The same does not apply to a waiver file created by using the `Add New Waiver file` menu item in the parent sub menu.

Creating a Waiver

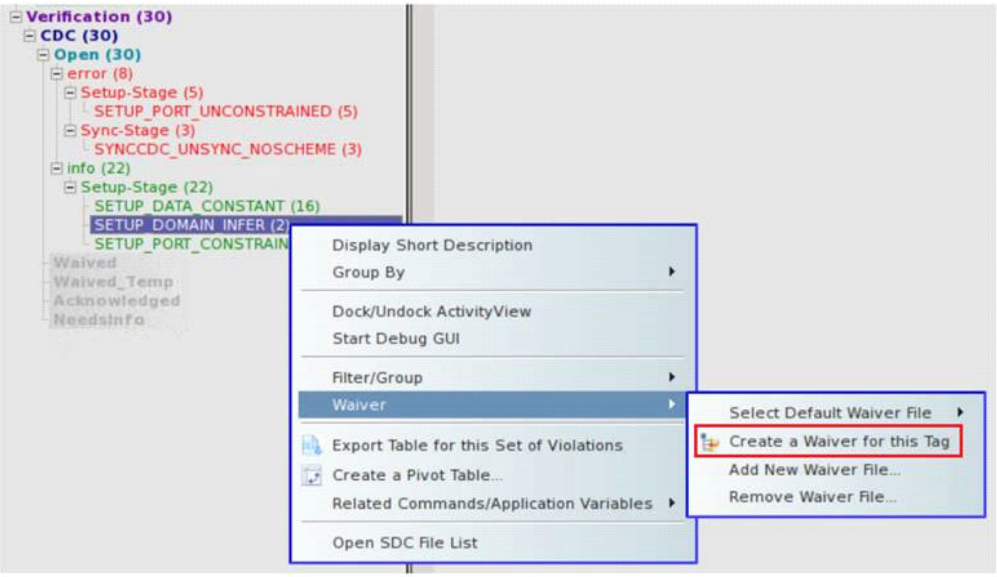
You can create waivers to:

- *Waive all Messages of a Tag*
- *Waive Messages Selectively*

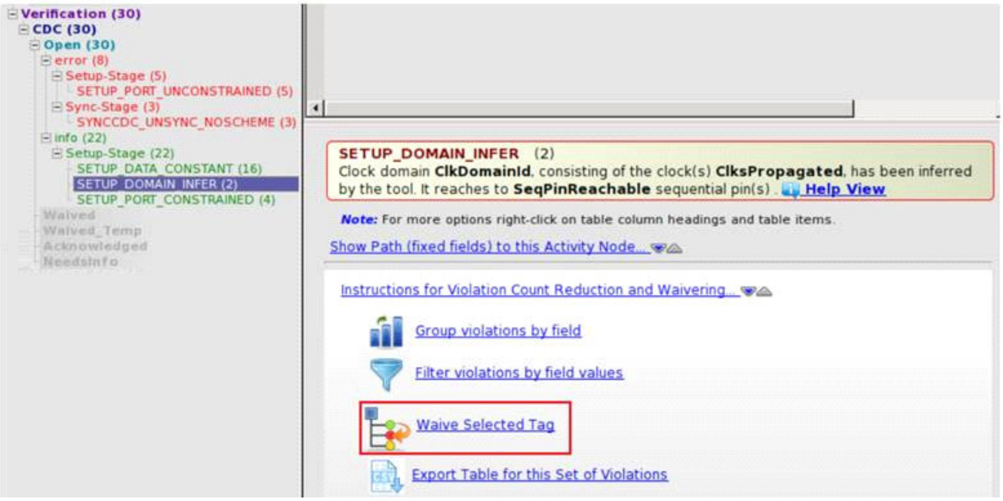
Waive all Messages of a Tag

You can waive all violations of a tag by using either of the following menu items:

- The `Create a Waiver for this Tag` menu item in the sub menu of the `Waiver` menu of the activity tree right-click menu as shown in the following figure. This menu item is available only when a node for a tag is selected and right-click on it.



- The `Waive Selected Tag` menu item from the information view as shown in the following figure.

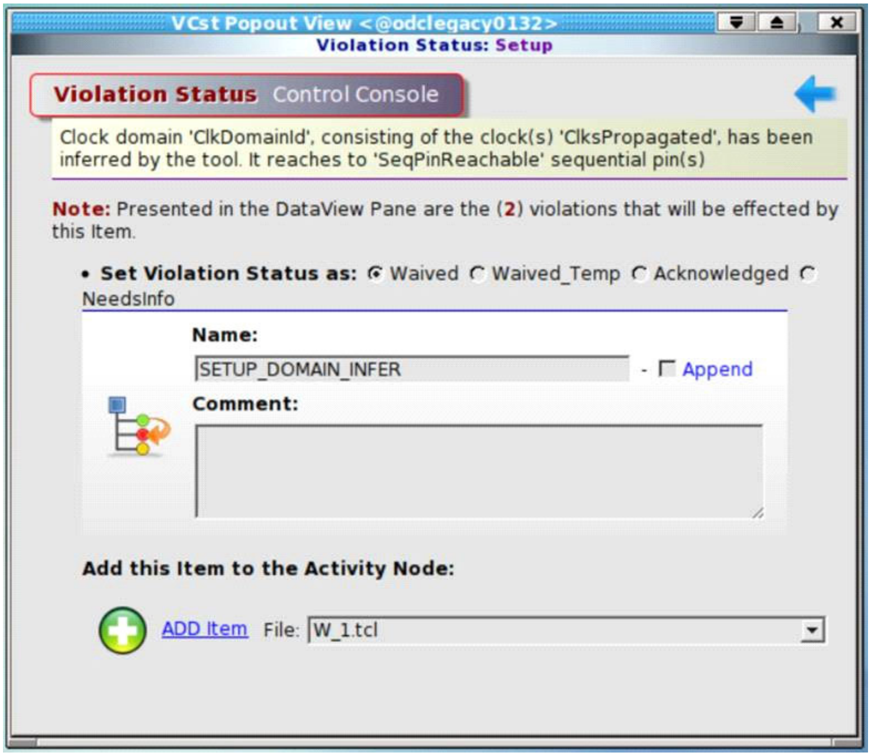

Verification (30)
CDC (30)
Open (30)
error (8)
Setup-Stage (5)
SETUP_PORT_UNCONSTRAINED (5)
Sync-Stage (3)
SYNCCDC_UNSYNC_NOSCHEME (3)
info (22)
Setup-Stage (22)
SETUP_DATA_CONSTANT (16)
SETUP_DOMAIN_INFER (2)
SETUP_PORT_CONSTRAINED (4)
Waived
Waived_Temp
Acknowledged
NeedsInfo

SETUP_DOMAIN_INFER (2)
Clock domain **ClkDomainId**, consisting of the clock(s) **ClksPropagated**, has been inferred by the tool. It reaches to **SeqPinReachable** sequential pin(s) . **Help View**

Note: For more options right-click on table column headings and table items.

Show Path (fixed fields) to this Activity Node...
Instructions for Violation Count Reduction and Waivering...
Group violations by field
Filter violations by field values
Waive Selected Tag
Export Table for this Set of Violations

For both the above menu items, the following waiver dialog is launched to waive all violations of the tag.



VCst Popout View <@odclegacy0132>

Violation Status: Setup

Violation Status Control Console

Clock domain 'ClkDomainId', consisting of the clock(s) 'ClksPropagated', has been inferred by the tool. It reaches to 'SeqPinReachable' sequential pin(s)

Note: Presented in the DataView Pane are the (**2**) violations that will be effected by this Item.

- **Set Violation Status as:** ● Waived ○ Waived_Temp ○ Acknowledged ○ NeedsInfo

Name:

SETUP_DOMAIN_INFER - □ Append

Comment:

Add this Item to the Activity Node:

ADD Item File: W_1.tcl

The dialog box provides the following options:

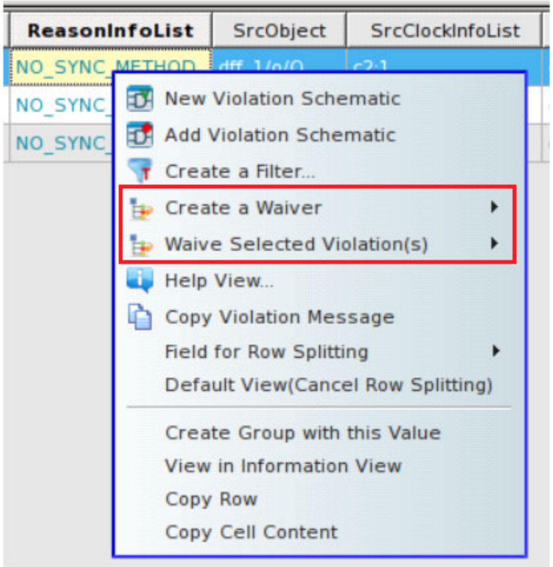
- Name: Name of the waiver.
- Comments: Any comments to be added to the waiver.
- File: The default waiver file is displayed in this drop down list. To change the waiver file, select a different file from the drop down list.

  A new waiver file also can be added using the `Add waiver file` option from the drop down list.
- ADD Item: Use this button to create the waiver.

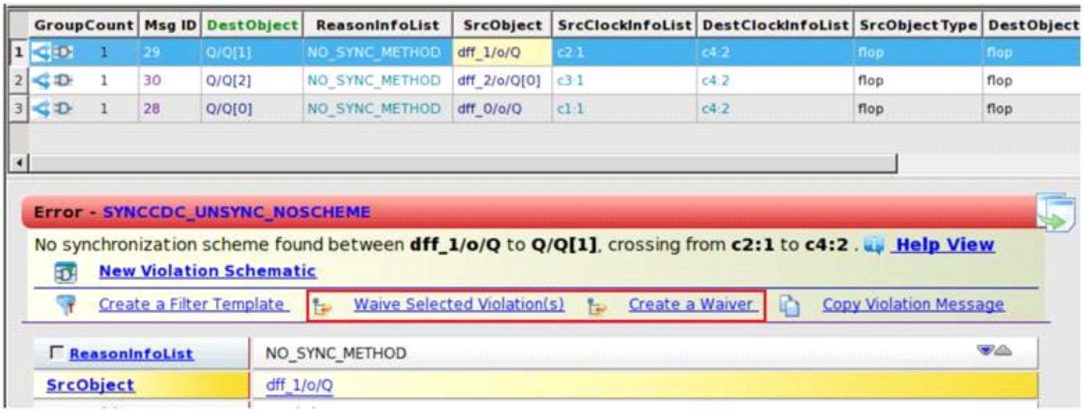
Waive Messages Selectively

You can waive selective violations of a tag by using either of the following menu items:

- The `Waive Selected Violation(s)` and `Create a Waiver` menu items from the data view as shown in the following figure:



- The `Waive Selected Violation(s)` and the `Create a Waiver` menu items from the information view.



	GroupCount	Msg ID	DestObject	ReasonInfoList	SrcObject	SrcClockInfoList	DestClockInfoList	SrcObjectType	DestObject
1	1	29	Q/Q[1]	NO_SYNC_METHOD	dff_1/o/Q	c2:1	c4:2	flop	flop
2	1	30	Q/Q[2]	NO_SYNC_METHOD	dff_2/o/Q[0]	c3:1	c4:2	flop	flop
3	1	28	Q/Q[0]	NO_SYNC_METHOD	dff_0/o/Q	c1:1	c4:2	flop	flop

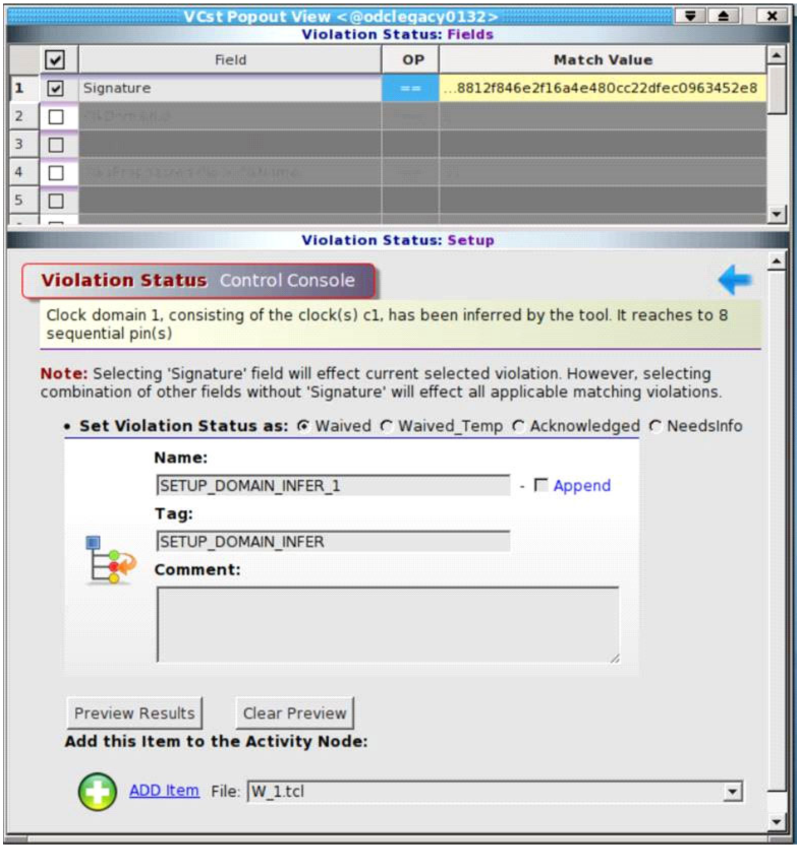
Error - SYNCCDC_UNSYNC_NOSCHEME

No synchronization scheme found between **dff_1/o/Q** to **Q/Q[1]**, crossing from **c2:1** to **c4:2** . Help View

□ ReasonInfoList	NO_SYNC_METHOD
SrcObject	dff_1/o/Q

You can click the `Waive Selected Violation(s)` menu item and the selected violation is waived to the default waiver file. This method can be considered as a quick method of waiving.

If you click the `Create a Waiver` menu item, the following dialog box is displayed to waive the violation.



VCst Popout View <@odclegacy0132>

Violation Status: Fields

	☑	Field	OP	Match Value
1	☑	Signature	==	...8812f846e2f16a4e480cc22dfec0963452e8
2	□	[illegible]	[illegible]	[illegible]
3	□			
4	□	[illegible]	[illegible]	[illegible]
5	□			

Violation Status: Setup

Violation Status Control Console

Clock domain 1, consisting of the clock(s) c1, has been inferred by the tool. It reaches to 8 sequential pin(s)

Note: Selecting 'Signature' field will effect current selected violation. However, selecting combination of other fields without 'Signature' will effect all applicable matching violations.

- **Set Violation Status as:** ● Waived ○ Waived_Temp ○ Acknowledged ○ NeedsInfo

Name: SETUP_DOMAIN_INFER_1 - □ Append

Tag: SETUP_DOMAIN_INFER

Comment:

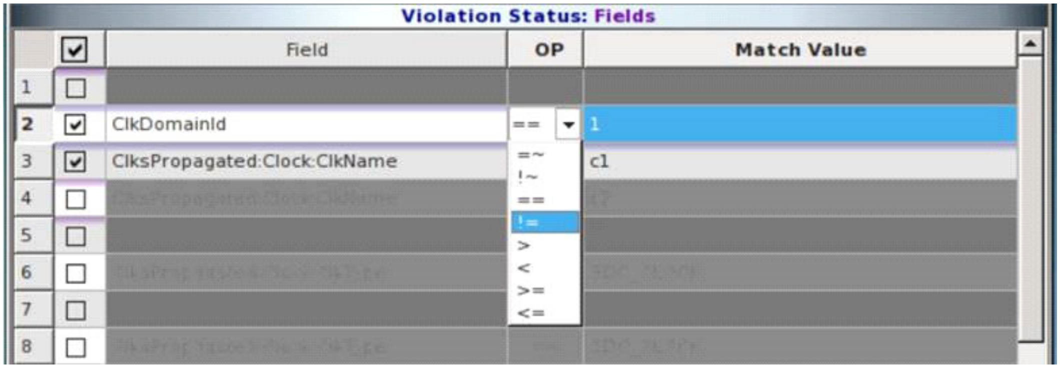
Add this Item to the Activity Node:

File: W_1.tcl

This dialog provides the following options:

- ❒ Name: Name of the waiver.
- ❒ Comments: Any comments to be added to the waiver.
- ❒ File: The default waiver file is displayed in this list. To change the waiver file, select a different file from the drop down list.

  A new waiver file also can be added using the `Add waiver file` option from the drop down list. the following dialog box is displayed:



Violation Status: Fields

	☑	Field	OP	Match Value
1	☐			
2	☑	ClkDomainId	==	1
3	☑	ClksPropagated:Clock:ClkName		c1
4	☐	ClksPropagated:Clock:ClkName		c2
5	☐			
6	☐	[illegible]		[illegible]
7	☐			
8	☐	[illegible]	==	[illegible]

- Fields table: By default, only the signature of violation is selected. You can select the check box of a field to enable the required fields to consider for waiving.
- OP (Operator of a field): Double-click on a cell under the `OP` column to see the drop down list icon. Required operator can be selected from the drop down list.

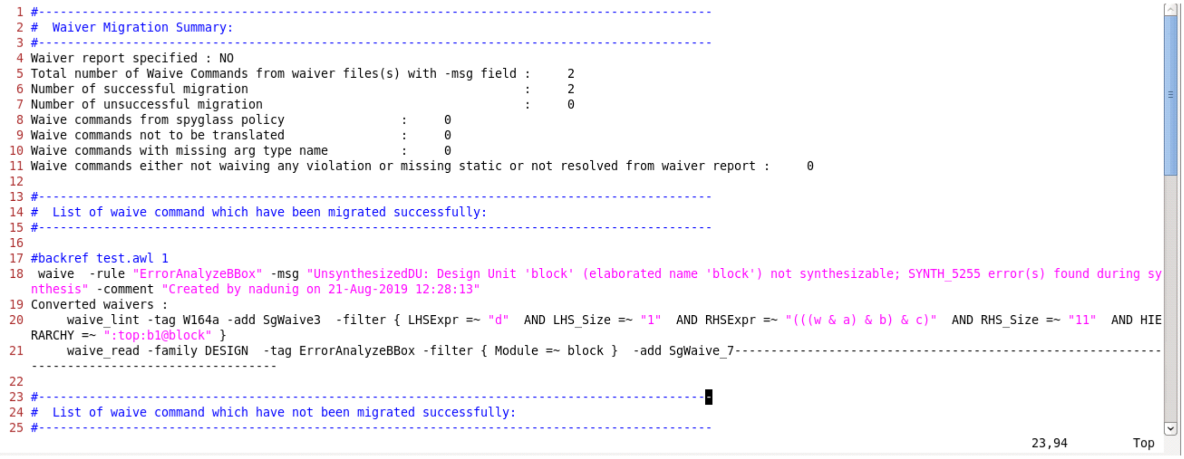
❒ ADD Item: Use this button to create the waiver

Reporting Waivers

VC SpyGlass generates the following reports:

- **merge_waiver_migration.rpt**: Use the `merge_waiver_migration` report to check the Waiver Migration Summary. This report can be used to check if the waiver commands have been migrated successfully or not. This report is generated in the `vcst_rtdb/reports/` directory.

  If some waivers are migrated successfully, it is included in the `List of waive command which have been migrated successfully` section. Any waiver that has an issue in migration is included in `List of waive command which have not been migrated successfully` section as shown in the following figure.



```
#-------------------------------------------------------------------------------------
#  Waiver Migration Summary:
#-------------------------------------------------------------------------------------
Waiver report specified : NO
Total number of Waive Commands from waiver files(s) with -msg field :     2
Number of successful migration                                      :     2
Number of unsuccessful migration                                    :     0
Waive commands from spyglass policy                  :     0
Waive commands not to be translated                  :     0
Waive commands with missing arg type name            :     0
Waive commands either not waiving any violation or missing static or not resolved from waiver report :     0

#-------------------------------------------------------------------------------------
#  List of waive command which have been migrated successfully:
#-------------------------------------------------------------------------------------

#backref test.awl 1
 waive  -rule "ErrorAnalyzeBBox" -msg "UnsynthesizedDU: Design Unit 'block' (elaborated name 'block') not synthesizable; SYNTH_5255 error(s) found during sy
nthesis" -comment "Created by nadunig on 21-Aug-2019 12:28:13"
Converted waivers :
     waive_lint -tag W164a -add SgWaive3  -filter { LHSExpr =~ "d"  AND LHS_Size =~ "1"  AND RHSExpr =~ "(((w & a) & b) & c)"  AND RHS_Size =~ "11"  AND HIE
RARCHY =~ ":top:b1@block" }
     waive_read -family DESIGN  -tag ErrorAnalyzeBBox -filter { Module =~ block }  -add SgWaive_7--------------------------------------------------------
---------------------------------

#-------------------------------------------------------------------------------------
#  List of waive command which have not been migrated successfully:
#-------------------------------------------------------------------------------------
                                                                                              23,94          Top
```

- **Waiver Report**: In SGUM, SpyGlass-like waiver report (waiver.rpt) is generated. You can use this report to check the waived violations.
- **pragma2waiver.tcl**: Pragma-based waivers are generated and applied automatically by tool. The waiver file is available at `vcst_rtdb/reports/pragma2waiver.tcl`.

Debugging Aids

VC SpyGlass provides the following aids that you can use to debug waiver related issues:

- **vcst_rtdb/reports/merge_waiver_migration.rpt**: Use the merge_waiver_migration report to check the Waiver Migration Summary. This report can be used to check if the waiver commands have been migrated successfully or not.

  If some waivers are migrated successfully, it is included in the `List of waive command which have been migrated successfully` section. Any waiver that has an issue in migration is included in `List of waive command which have not been migrated successfully` section.

- **Waiver Report**: In SGUM, SpyGlass-like waiver report (waiver.rpt) is generated. You can use this report to check the waived violations.
- **TCL Commands**: VC SpyGlas provides the following Tcl commands that can be used to debug waiver-related issues:
  - ❒ **waive_lint**: The `waive_lint` command can be used to obtain the waiver summary.
  - ❒ **waive_lint -tcl**: The `waive_lint -tcl` command can be used to get list of all the applied waiver commands.
  - ❒ **waive_lint -not_applied**: The `waive_lint -not_applied` command can be used to obtain the waiver summary of the waiver commands that do not waive any violations.
  - ❒ **waive_lint -tcl -not_applied**: The `waive_lint -tcl -not_applied` can be used to get list of all the applied waiver commands that do not waive any violations.

Waiver-related Commands and Application Variables

VC SpyGlass provides the following waiver-relates commands and application variables:

- *Application Variables*
- *Commands*

Application Variables

The following application variables are available:

- configure_waiver_file_action: Use this application variable to delete a waiver command or move a waiver command from one file to another file. The application variable can take three values; `no_change |comment |delete`. By default, the application variable is set to `no_change`. Refer the man page for more details.
- enable_waiver_opt: This application variable is enabled by default. Disabling this application variable impacts the run time.
- enable_pragma_based_waiver: Use this application variable to decide if pragma-based waivers are generated or not. This application variable can take any one of three values; `disabled| enabled |create_invalid_rule_pragmas`. By default, this application variable is set to enabled. Refer the man page for more details.
- ignore_module_instances_outside_ip: Use this application variable to perform the SpyGlass default waive -ip behavior in VC SpyGlass. In this case, even if one instance of a module is inside an IP, the violation is waived. Refer the man page for more details.
- default_waiver_file: Use this application variable to specify the default waiver file where waivers are generated automatically.

Commands

The following waiver-related commands are available:

- *Waiver Configuration Commands*
- *Waiver TCL Commands*

Waiver Configuration Commands

The following waiver configuration commands are available:

- configure_waiver_filter_field: Use this command to configure filter fields of a tag. You can customize filter fields for certain tags in waiver window. If this filter command is not set, all fields are enabled in the waiver window for each tag. Refer the man page for more details.
- configure_waiver: Use this command to configure the following for a waiver:
  - ❐ Disable the waiver with the specified name.
  - ❐ Disable waiver field processing.
  - ❐ Disable multi-threading in waiver.
  - ❐ Waiver will not be applied on given tags/severity.

NOTE: *The configure_waiver_filter_field command does not support the above configurations.*

Waiver TCL Commands

The following Tcl commands are available:

- waive_lint: Use this command to waive lint violations. Refer the man page for more details.
- manage_waiver_file: Use this command to add/remove a waiver file from the tool. Refer the man page for more details.
- migrate_waivers: Use this command to migrate block level waiver into top level. Refer the man page for more details.
- sg_read_waiver: Use this command to convert SpyGlass waivers into VC SpyGlass waivers. Refer the man page for more details.
- waive_read: Use this command to waive design read violations including built-in messages. Refer the man page for more details.
- get_violation_waiver: Use this command to get waiver name which is related to violation ID.
- get_waivers: Use this command to query all the waivers that match the specified pattern.

- list_all_waiver_files: Use this command to display a complete list of waiver files which are associated with such waivers.
- set_file_for_waiver: Use this command to set waiver file to generate waiver command.
- get_waiver_attribute: Use this command to get different attribute values, such as severity, tag etc of each waiver command.
- set_violation_state: Use this command to set state of violation based on Tags, Severity, Filter rules using debug fields, Wild cards and expressions, and add comments.