VCS®/VCSi™ Design Checker User Guide

Version C-2009.06
June 2009

Comments?
E-mail your comments about this manual to:
vcs_support@synopsys.com.

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of
>
> _____ and its employees. This is copy number_____."

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.

Registered Trademarks (®)

Synopsys, AMPS, Cadabra, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, iN-Phase, in-Sync, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PrimeTime, SiVL, SNUG, SolvNet, System Compiler, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, Columbia, Columbia-CE, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, Direct Silicon Access, Discovery, Encore, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, HSIMplus, HSPICE-Link, iN-Tandem, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Raphael-NES, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, Taurus, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

1

Design Checker in VCS

This chapter provides an overview of Design Checker available with VCS and includes the following sections:

Introduction

Design Checker is a static code purification core for checking assertions, Verilog, and SystemVerilog code. The tool contains approximately 260 rules that perform semantic analysis and check for coding styles that may cause problems in the simulation and synthesis flow. Design Checker helps you identify common and subtle coding mistakes.

Setting up VCS Design Checker

Before running VCS Design Checker, you must setup the environment variables. For more information about how to setup the environment variables, see the *VCS User Guide*.

2

Design Checker in Batch Mode

You can run the Design Checker in batch mode by specifying switches in the command line when you invoke the tool. This way, the tool runs to completion unattended, which can be handy for script-driven test environments. This chapter explains how to use the Design Checker in batch mode and includes the following sections:

Running the Design Checker in Batch Mode

Design Checker lets you specify a list of rules to verify your design. It provides the following two ways to specify the list of rules:

1. List down the rules that you want to perform on your design in a Checker configuration file, and then specify the Checker configuration file at the VCS command with appropriate command options. This way is the preferred mode of running VCS Design Checker. For example:

   ```
   % vcs design_files -check_config config_file.tcl
   ```

2. Specify the rules at the VCS command prompt as follows:

   ```
   % vcs design_files -check_rule ASSERT002
   ```

Note:

If you use `-check_rule` or `-nocheck_rule` with `-check_config` in the command line, `-check_config` will always get higher priority. For example, if the command line is `-check_config config.tcl -check_rule ASSERT001`, where config.tcl has command `unset_rule -rule ASSERT001`, then Design Checker will not produce violations related to rule `ASSERT001`.

Using the Checker Configuration File

A Checker configuration file is a Tcl file in which you specify the rules you want to validate in the design. The Checker configuration file may also contain built-in Tcl commands. You need to use the `-check_config` option in the batch mode to specify the Checker configuration file to the Design Checker. For example:

```
% vcs design_files -check_config config_file.tcl
```

A sample Checker configuration file is shown below.

```
set_rule -rule VMM_ENV001
set_rule -rule {VMM_ENV001 VMM_INT006}
set_message -rule VMM_ENV001
\\"NEW MESSAGE FOR THE RULE"
```

Note:

Please see section "Using Tcl Commands" to review built-in Tcl commands provided for the Checker configuration file.

Using Command-Line Options

This section describes the command-line options associated with Design Checker and illustrates how to use them with examples:

`-check_config` *argument*

Use this option to specify your Checker configuration file. For example;

```
% vcs design_files -check_config config_file.tcl
```

-check_rule *argument*

Use this option to enable rules verification. It takes an argument through which you can control the range of rule selection. For example:

```
% vcs design_files -check_rule VMM_ENV004
```

To check the design for multiple rules, you need to specify the rule names separated by the + operator. For example:

```
% vcs design_files -check_rule ASSERT001+PROPERTY002
```

Note:

> Do not separate the '+' operator with a blank space when using it with the -check_rule option.

-check_module *argument*

Use this option to perform rule checks on a specific module to the entire sub-hierarchy below. You can specify the module as an argument. For example:

```
% vcs design_files -check_module sub_module_1
```

The above command enables rule checking on module sub_module_1 and its entire sub hierarchy.

You can also specify multiple modules with the `-check_module` option using the + operator. For example:

```
% vcs design_files -check_module s1+i1
```

The above command enables rule checking on modules s1 and i1 and its sub hierarchies.

Note:

The -check_module option is not applicable to VMM rules.

-check_log *argument*

Use this option to specify a file name to log the error messages being displayed while the Design Checker performs rule checks on your design. For example:

```
% vcs design_files -check_config cfg.tcl -check_log
result.log
```

Note:

When you use the -check_log option, the violation messages (result) are stored in the check.log file.

-check_design_top *argument*

Use this option to specify the top module of the design. The argument should be the design top module. This option is required only if Structural rules of VCS Design Checker is enabled. For example:

```
% vcs design_files -check_design_top top
```

Note:

Do not specify a testbench top module name with `-check_design_top` option.

-nocheck_rule *argument*

Use this option to disable rule checking. You can control the range of rule selection with the help of an argument. For example:

```
% vcs design_files -nocheck_rule SEQUENCE004
```

-nocheck_module *argument*

Use this option to omit a specific module from rule check analysis. You can specify the module as an argument with the command option, as shown in the following example:

```
% vcs design_files -nocheck_module s1
```

If you specify both `-check_module` and `-nocheck_module` options, the option which comes later in the command line gets precedence. For example,

```
% vcs design_files -nocheck_module s1 -check_module s2
```

Here, `-check_module s2` gets precedence since it appears later.

The behavior of `-check_* and -nocheck_*` with any valid command-line options are as follows:

- The argument list of command-line options used for disabling rules (`-nocheck_*`) are same as their corresponding options for enabling rules (`-check_*`).
- Specifying command-line options `-check_rule`, `-check_module`, `-nocheck_rule`, `-nocheck_module`, `-check_log`, and `-check_design_top` without an argument is an error.

- If you specify `-nocheck_rule VMM_ENV001 -check_rule VMM_ENV001` (check_rule comes later in the command line), `check_rule` gets higher priority being the latest one in the command line.
- You cannot combine multiple command-line options using the + operator. Each command-line option should be specified separately.

Using Tcl Commands

The following is a detailed description of the Tcl commands.

set_rule

Use the command set_rule to perform a specific rule check or a list of rule checks that were previously omitted from the analysis.

Syntax:

set_rule [-rule *rule_name*]\
`[-rule {rule_list_separated_by_space}]`

The option `-rule` *`rule_name`* enables the specified rule. The option `-rule {rule_list_separated_by_space}` enables a list of rules that are specified within the { } braces.

For example:

set_rule -rule VMM_INT007

set_rule -rule {ASSERT001 ASSERT002}

unset_rule

Use the command unset_rule to disable a specific rule or a list of rules from rule check analysis.

Syntax:

unset_rule [-rule *rule_name*] [-all] \

`[-rule {rule_list_separated_by_space}]`

The option `-all` disables all the rules. The option `-rule` *`rule_name`* disables the specified rule. The option `-rule {rule_list_separated_by_space}` disables a list of rules that are specified within the { } braces.

For example:

unset_rule -rule ASSERT001

unset_rule [-all]

unset_rule -rule {PROPERTY001 PROPERTY002}

set_message

You can use the command set_message for the following:

- To set the limits on the number of messages issued for a rule.

Syntax:

set_message [-rule *rule_name*]\

`[-rule {rule_list_separated_by_space}]\`

[-limit *number*] [-cumulative_limit *number*] [rule_parameter]

Use the option `-limit` to limit the number of messages issued for a rule. You can use this with only one of the following options:

For example:

set_message -rule ASSERT001 -limit 20

- `-rule {rule_list_separated_by_space} -` sets the limit of all the rules listed within the braces `{}` to *`number.`*

For example:

set_message -rule {ASSERT001 ASSERT002 ASSERT003} -limit 10

Use the option `-cumulative_limit` to limit the total number of messages issued by the tool. The Design Checker stops issuing messages once the cumulative limit is reached.

For example:

set_message -cumulative_limit 200

Note:

By default, there is no limit to issue the number of messages for a rule.

If you specify rule group name with set_message, Design Checker gives the following warning and continues validation or reporting and ignores set_message command.

```
Warning-[WRONG_RULE_SETMESSG] Rule category used with
set_message
```

Rule category does not work with set_message. You must use valid rule ID.

set_rule_parameter

Use the set_rule_parameter command to change the rule parameter value.

Syntax:

set_rule_parameter -rule *rule_name* \
-parameter *parameter_name* -value {*value*}

For example:

```
set_rule_parameter -rule VMM_ENV004 -parameter
NB_MAX_CLOCKS -value {0}
```

Using the Log Files

When you run the Design Checker in batch mode or Tcl mode, log files are generated. The violation messages or the results are stored in the Log files. Among all the output files (log files), `check.log`, `smartlog.check.log`, and `smartlog.check.sml` files are important for you.

- **check.log:** This is a text file that shows the violations summary along with the rule details. You can open this file and view the results of the Design Checker.
- **smartlog.check.sml:** This file has the rule labels. DVE uses this file to show the results in the GUI mode.

- **smartlog.check.log:** This is also a text file that contains the violation messages. This file is specifically used by the DVE to match the rule details with the rule labels (present in the smartlog.check.log file).

The log files are generated by default. The smartlog.check.log file is generated only for DVE purpose. To view the log in the batch or Tcl mode, use the check.log file.

If you specify an argument with the -check_log option, then the argument will be considered while naming the log files. For example:

```
% vcs design_files -check_config cfg.tcl -check_log
result.log
```

Here, the log files that are generated will have x in their name, such as result.log, smartlog.result.log, and smartlog.result.sml.

Using Black-Box

Designers intentionally leave some part of design description empty for checks with specific message. You may not want to run Design Checker on an IP or a large block of a design that is already used in the design. You can specify these IPs or blocks of a design as black-boxes, so that Design Checker could omit them from reporting violations. This helps you in minimizing the number of violations reported on unwanted modules, entities, or db cells.

A black-box is either categorized as an automatic black-box or a user-defined black-box. For example, an unbound VHDL component can be treated as an automatic black-box. The purpose of having a user-defined black-box is also same. In addition, a user-defined black-box also allows to keep your design and makefiles unchanged.

Note:

Black-boxes are not applicable to VMM rules.

Automatic Black-Box

Design Checker treats the following as an automatic black-box:

- A Verilog module/udp that does not contain any statement (i.e. only port definitions and declaration or without initializations/ implicit assigns).
- A DB cell that does not contain any known attribute or functionality, or on which db reader cannot recognize the functionality for any reason.

User-Defined Black-Box

Design Checker treats the Verilog modules, VHDL entities, or db cells specified with the Tcl command set_black_box as user-defined black-box. For a user-defined black-box module, Design Checker does the following:

- Ignores çhecking a module and part of design hierarchy from this module to leaf (as it doesn't consider the instances inside it).
- Gives user-defined black-box violation ONLY for the module you specify.

The module which is instantiated in an user-defined black-box module and also instantiated in another module (normal, not black-box) is considered by Design Checker through the other hierarchy.

Note:

If you do not select any rule, the black-box message will not appear, as the Design Checker flow is NOT enabled. For example, if the Checker configuration tcl file has only "set_black_box {<mod_name>}" without any set_rule, you will NOT get any black-box violation.

Example:

```
module top1;
                mid1 inst1();
                mid2 inst2();
endmodule
module mid1();
                bot1 inst1();
                bot2 inst2();
endmodule
module mid2();
                bot2 inst1();
endmodule
module bot1();
endmodule
module bot2();
endmodule
```

Here, module mid1 is a user-defined black-box. In such scenario, Design Checker does the following:

- It flags the user-defined black-box violation only for module mid1.
- It skips checking if you select a rule between modules mid1 and bot1.
- It will NOT skip checking module bot2, as it is present through the design hierarchy path top1->mid2->bot2.

Black-Box Command

Automatic black-box & user-defined black-box for verilog modules applies to both language checks & structural checks. Automatic black-box & user-defined black-box for DB cell applies to structural check.

The command to specify user-specified black-box, in check_config file is as follows:

```
set_black_box <module_or_entity_name_list> -automatic
```

Where,

- -automatic: (Optional). If you specify this option, black-box will be detected automatically. By default, automatic detection is off.
- <module_or_entity_name_list>: (Optional). Identifies the module name.

Both option work independent of each other.

3

Design Checker in GUI Mode

This chapter provides information on using the Design Checker GUI mode in the following sections:

Overview

The Design Checker GUI (Graphical User Interface) provides an alternate method of creating a Checker configuration file and view the rule violation messages. You can simply navigate and select the

rules to create a Checker configuration file in the Design Checker GUI. You can then use this Checker configuration file to run in your source file.

Note:

You can run the Design Checker only in the batch mode. For more information about how to run the Design Checker in batch mode, see “Running the Design Checker in Batch Mode” .

After running the Design Checker in the batch mode, you can view, filter, and analyze the rule violation messages quickly in the Design Checker GUI. You can also open the source file to view the lines that have the violations. This section explains the Design Checker GUI and describes the tasks that you can perform in the GUI.

Design Checker GUI Workarea

The Design Checker GUI consists of the following windows:

- “Rules Setup Window”
- “Summary Window”
- “Source Viewer”

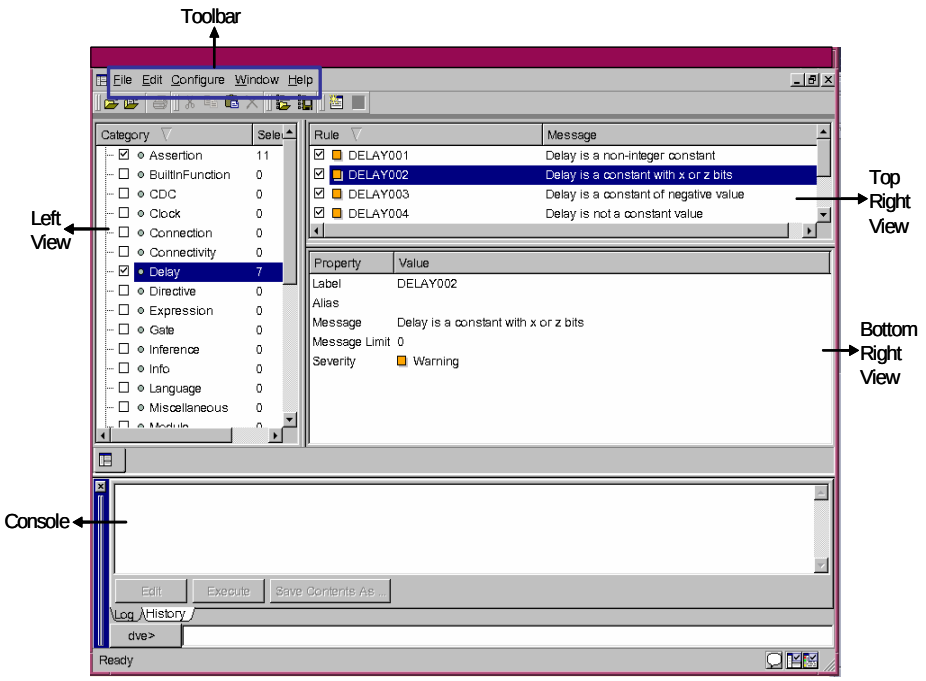
Rules Setup Window

You can create, open, and modify your Checker configuration file in the Rules Setup window. It consists of the following views:

- Left view: Displays rules category.
- Top right view: Displays rules under each category.

- Bottom right view: Displays the rule properties.
- Console: Contains Log and History tab. The Log tab displays the console messages and History tab displays the list of commands.
- Toolbar: Consists of File, Edit, Configure, Window, and Help menu.

The following illustration shows the Rules Setup window:



File Menu

The File menu contains the following options:

Options	Description
Open Database	Displays the Open Database dialog, which lets you select and open the smartlog file for viewing the Design Checker results.

Options	Description
Close Database	Displays the Close Database dialog, which lets you close the log file.
Open File	Displays the Open Source File dialog, which lets you select and open any source file in the editor.
Close File	Closes the source file.
Execute TCL scripts	Displays the Execute TCL Scripts dialog, which lets you open the Checker configuration or filter tcl file.
Load Session	Displays the Load Session dialog, which lets you load a saved session.
Save Session	Displays the Save Session dialog, which lets you save the current session.
Print	Prints the content of an active window.
Recent Database	Displays a list of recently opened databases.
Recent TCL Scripts	Displays a list of recently run scripts.
Recent Session	Displays a list of recently opened sessions.
Close Window	Closes the active window.
Exit	Exits the Design Checker GUI.

Edit Menu

The Edit menu contains the following options:

Options	Description
Cut/Copy/Paste/Delete	Cuts, copies, pastes, or deletes any text in the Source Viewer or Console.
Find/Find Next/Find Previous	Opens the Find dialog, which lets you search for matching text.
Filters	Displays the Message Filters dialog, which lets you work with filters.

Options	Description
Preferences	Displays the Application Preferences dialog. Note: In the Application Preferences dialog, you must select only the Static Check option under Categories in the left pane to change your preferences. Other options are not relevant to the Design Checker GUI. In the Application Preferences dialog you can select the severity level icons color, view or hide the grid, view or hide the source viewer, and set the maximum number of rule violation messages to sort in the Design Checker GUI.

Configure Menu

The Configure menu contains the following options:

Options	Description
Open Configuration	Displays the Open Rule Configurations File dialog, which lets you open the Checker configuration file. Note: You can also open the Checker configuration file from **File > Execute TCL Scripts** option.
Save Configuration	Lets you save the Checker configuration file.
Save Configuration As	Lets you save the Checker configuration file with a different name.

Window Menu

The Window menu contains functions that you can use to change the arrangement of the display windows in the Design Checker GUI. You can tile the windows vertically or horizontally, or cascade them. You can also dock the active window on the top, bottom, right or left, or undock it.

Help Menu

The Help menu provides access to the Design Checker documentation and lets you view the version of Design Checker.

History Tab

The History tab in the Console lets you view the list of commands that you have run in the console. This tab contains the following buttons:

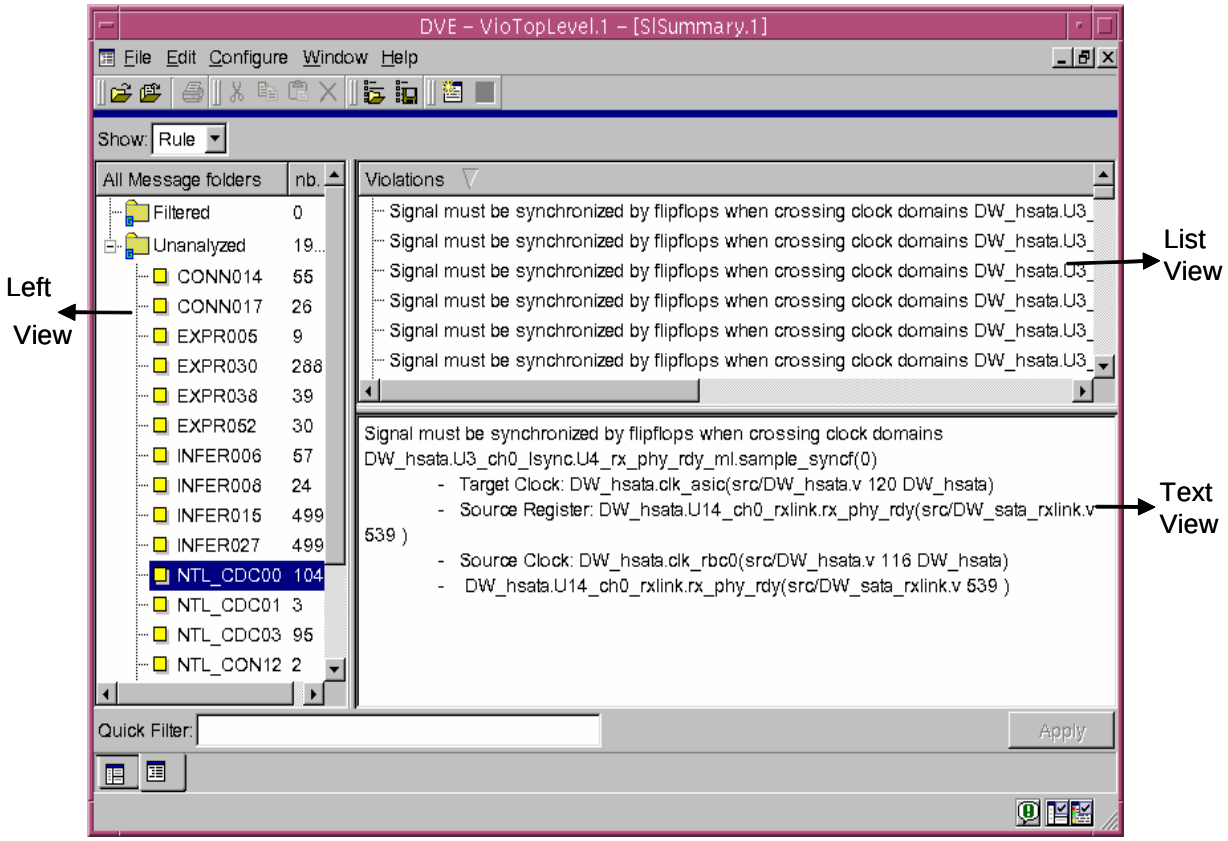
Options	Description
Edit	Lets you edit the command.
Execute	Lets you run the command.
Save Contents As	Lets you save the list of commands.

Summary Window

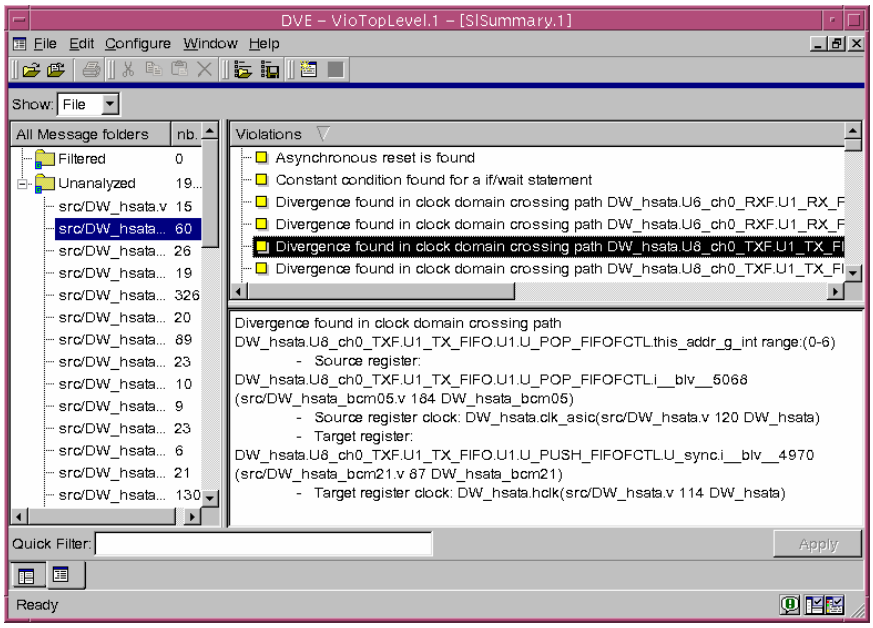
The Summary window is to display the checker results. It consists of the following views:

- Left View: Shows the All Message Folders. You can view the source file and open it in the editor or view the rules that are violated.

  The following illustration shows the results of the checker when violations are grouped by Rule. Colored icons represent the rule severity.



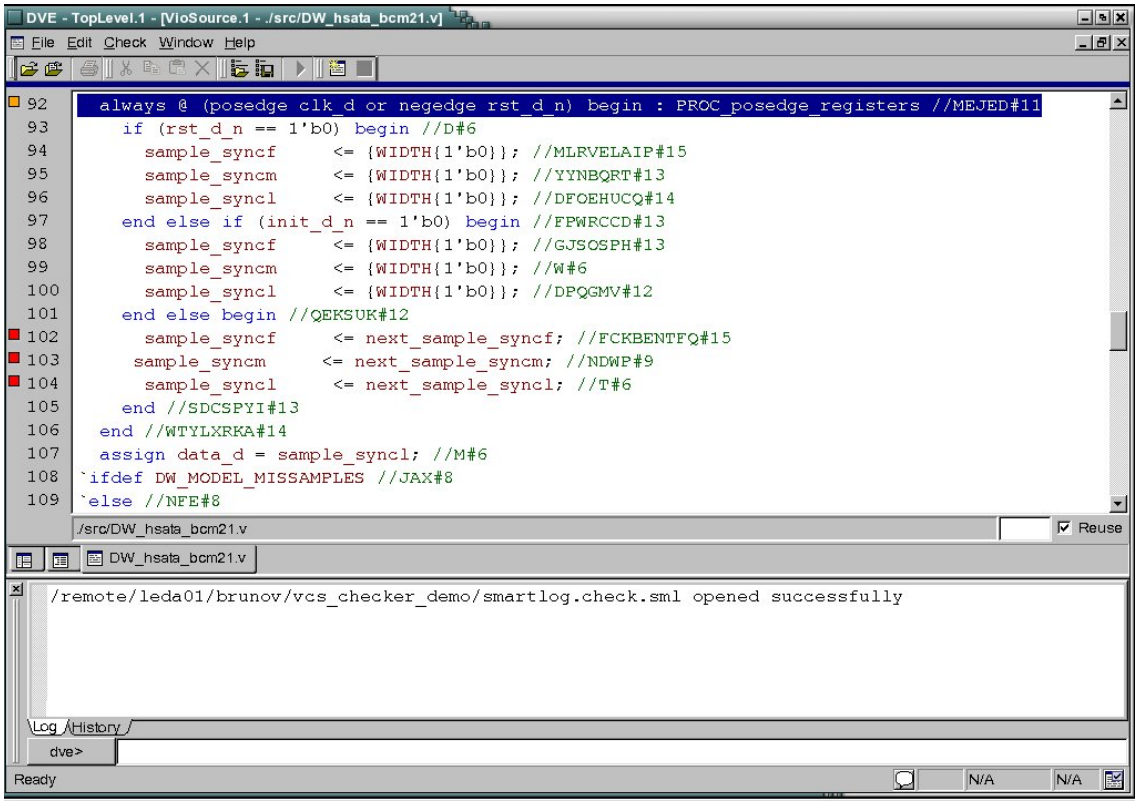
The following illustration shows the results of the checker when violations are grouped by Files. Colored icons in the Violations list represent the rule severity.



- List View: Shows all the violations for the selected rule. The List view on the top right shows the main message of each violation, the file name, and the line number of the violation.
- Text View: Shows the detailed message of the violation. The Text view is located at the bottom right.

Source Viewer

The Source Viewer is to view the violations in the source file. The lines that have the violations are annotated. The following illustration shows the Source Viewer:



Using the Design Checker GUI

The tasks that you can perform in the Design Checker GUI are as follows:

- “Creating a Checker Configuration File”

- “Opening and Modifying a Checker Configuration File”
- “Viewing the Violations”
- “Creating Filters”

Creating a Checker Configuration File

You can select the rules you want to run in your source file and create a Checker configuration file in the Rules Setup window. You can use this Checker configuration file when you are running the Design Checker in the batch mode.

Note:

You must run the Design Checker only in the batch mode. For more information about the Checker configuration file and how to run the Design Checker in batch mode, see “Running the Design Checker in Batch Mode”

To Create a Checker Configuration File:

1. Invoke the Design Checker GUI using the following command in the batch mode:

   ```
   dve -static_check
   ```

2. Select the rule categories in the Left view.

   The rules appear in the top right view.

3. Select the rules that you want to run on your source file.

   The rule properties appear in the bottom right view. You can alias the rule, modify the rule message, and set the maximum number of messages to be displayed for the rule.

4. Click **Configure > Save Configuration As**.

   The Save Rules Configuration As dialog appears.

5. Enter a file name, and click **Save**.

   The Checker configuration file is created and saved.

Opening and Modifying a Checker Configuration File

You can open the Checker configuration file and modify it in the Rules Setup window.

To Open and Modify the Checker Configuration File:

1. Invoke the Design Checker GUI using the following command in the batch mode:

   ```
   dve –static_check
   ```

2. Select **Configure > Open Configuration**.

   The Open Configuration dialog appears.

3. Locate and select the Checker configuration file that you created, and click **Open**.

   The Checker configuration file opens.

4. Modify the rule categories or rules as appropriate.

5. Click **Configure > Save Configuration**.

   The Checker configuration file is saved.

Viewing the Violations

You can view the rule violation messages in the Design Checker GUI. When you run the Design Checker in the batch mode, a smartlog file is generated in the current directory. You need to open the smartlog file in the Design Checker GUI to view the results. For more information about the smartlog files and other log files that are generated after running the Design Checker, see “Running the Design Checker in Batch Mode” .

To View the Violations

1. Invoke the Design Checker GUI using the following command in the batch mode:

   ```
   dve -static_check
   ```

2. Click **File > Open Database**.

   The Open Database dialog appears.

3. Locate and select the smartlog file, and click **Open**.

   The file opens in the Summary window. By default, the Rule option is selected in the Show drop-down list.

4. Select a rule in the left view.

   The violations under the selected rule appear in the top right view along with the file name and the line number.

5. Click a violation in the Violations list.

   The details appear in the bottom right view.

6. Select **File** in the **Show** drop-down list.

The source files are listed in the left view under All Message Folders.

7. Select the source file, and double-click to open it in the Source Viewer.

   The line number that has the violation is highlighted.

Creating Filters

You can sort the rule violation messages to view only the relevant messages using Filters. You need to specify a condition, such as matching text, and select an action you want to take on the rule violation messages. When you apply the filter on the rule violation messages, the messages are sorted based on the condition you specified and action is taken based on the action you specified when creating the filter.

For example, if there are several delay rule violation messages and clock rule violation messages and you want to view only the delay rule violation messages. Then, you can create a filter, specify a condition such as matching text `delay`, and select an action to move the delay rule violation messages in the Filtered folder.

To Create Filters

1. Invoke the Design Checker GUI using the following command in the batch mode:

   ```
   dve –static_check
   ```

2. Click **Edit > Filters**.

   The Message Filters dialog appears.

3. Click **Add**.

The Filter Editor dialog appears.

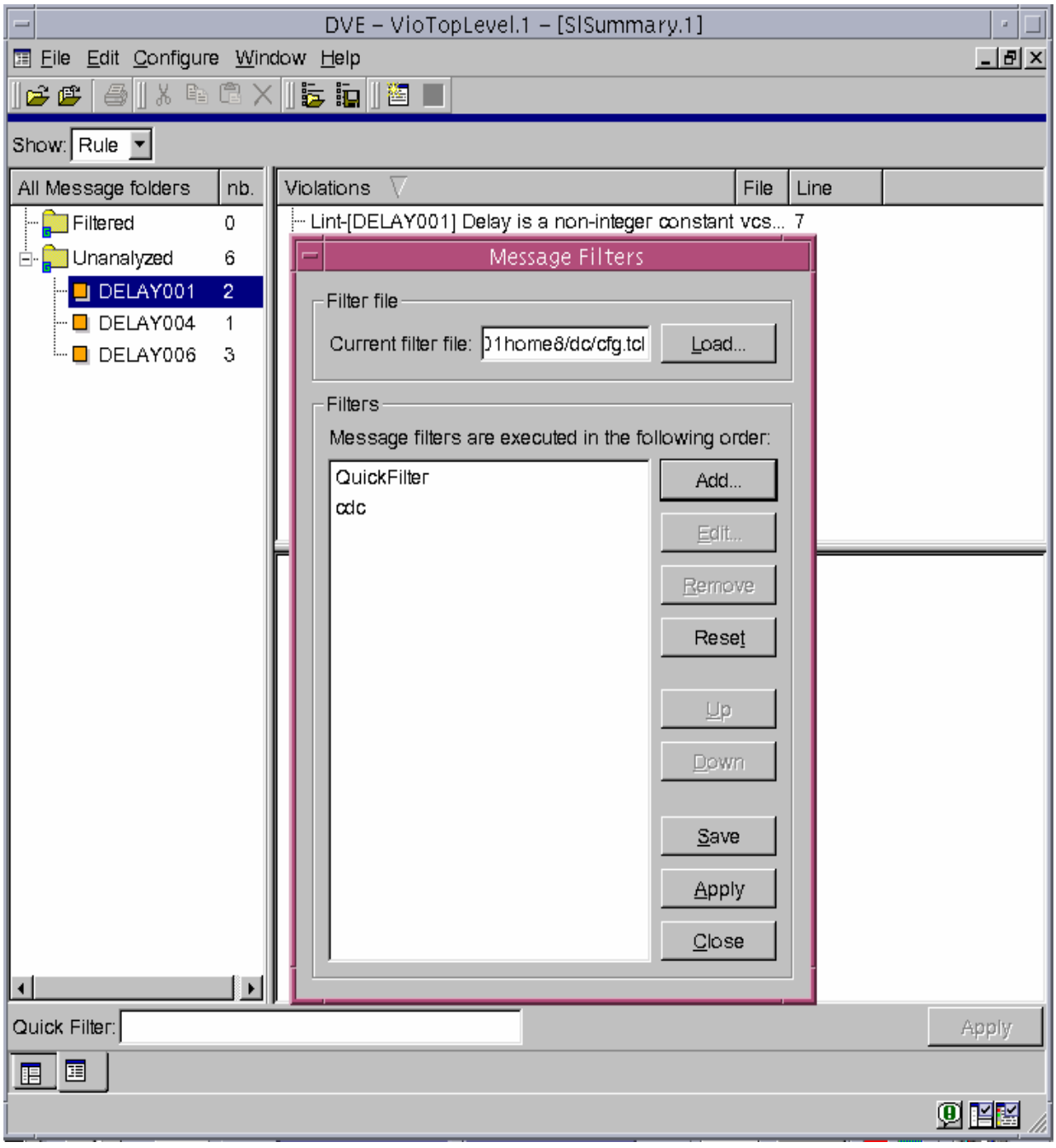
4. Enter a filter name, select the conditions and actions as appropriate, and click **OK**.

   The filter is created and appears in the Message Filters dialog.

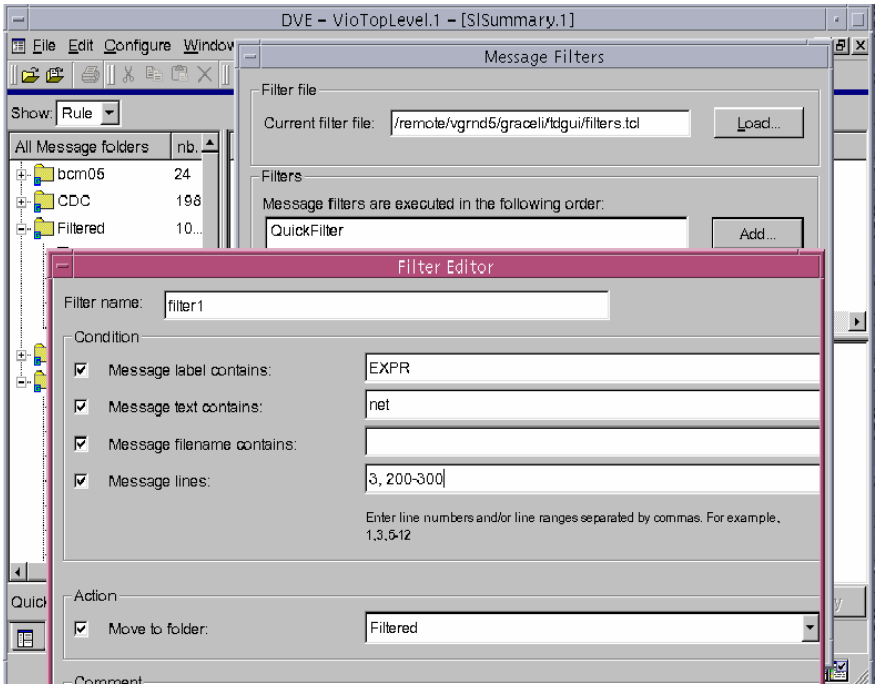
In the Message Filter dialog you can do any of the following:

- Select a filter, and click **Apply** to run the filter on the rule violation messages. You can select the Quick Filter and edit the message condition and apply. The Quick Filter is always executed first and then other filters.
- Arrange the order of filter using the **Up** and **Down** buttons.
- Modify the filter using the **Edit** button.
- Remove the filter using the **Remove** button.
- Load a filter file using the **Load** button.

The following illustration shows the Message Filters dialog:



The following illustration shows the Filter Editor dialog:

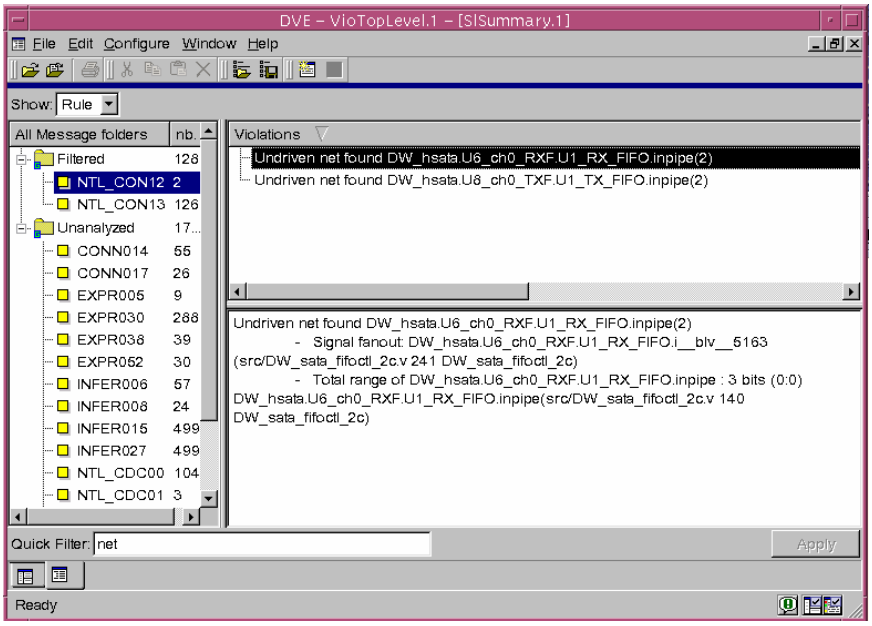


Applying Quick Filter

You can apply a filter using any of the following methods:

- Using the **Quick Filter** option at the bottom of the Summary window.
- Using the Quick Filter in the Filter dialog.
- Using the **Execute Tcl Script** option from the **File** menu.

The following illustration shows the Quick Filter option in the Summary window:



To apply quick filter:

1. Open the Summary window.
2. Enter some text in the Quick Filter field.
3. Click **Apply**. The rule violation messages that matched the text are sorted.

Tcl Commands for Filters

You can also use the following Tcl commands in batch mode for filtering the messages:

gui_slog_summary_create_filter

Creates a new filter for the smartlog Summary window. This command is the main command for creating filters. The syntax of the command is as follows:

```
gui_slog_summary_create_filter
name string: Name of the filter
condition list: Matching conditions of the filter
[-action list]: Action to perform on matching
[-comment string]: Comment associated to the filter
```

For example, you can create a filter using the following command:

```
gui_slog_summary_create_filter -name Actuals -condition
{message Actual} -action {move Connections}
```

This filter will check all rule violation messages, sort the message with Actual as their test, and move matching violation messages to Connections folder.

gui_slog_summary_apply_filters

Populates the Summary window with the smartlog database. This command applies all the created filters to the current smartlog summary (if no smartlog file is opened, the command does nothing).

gui_slog_summary_remove_filter

Removes one or all filters for the smartlog Summary window. The syntax of the command is as follows:

```
gui_slog_summary_remove_filter
[-name string]: Name of the filter
[-all]: remove all filters
```

gui_slog_summary_set_filter_file

Sets the filter file name (filters will be saved in that file) for the smartlog Summary window. This command sets the file in which filters have to be saved. The syntax of the command is as follows:

```
gui_slog_summary_set_filter_file
name string: Name of the filter
```

gui_slog_summary_save_filter_file

Saves current filters in current filter file. This command saves all the current filters in the current filter file (if no filter file specified, does nothing). Make sure that the file is overwritten.

gui_slog_summary_get_filter

Gets filter information for the smartlog Summary window. This command gets the list of all the created filters.