Atrenta Console
Reference Guide

Version N-2017.12-SP2, June 2018

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

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.

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

Report an Error

The SpyGlass Technical Publications team welcomes your feedback and suggestions on this publication. Please provide specific feedback and, if possible, attach a snapshot. Send your feedback to *spyglass_support@synopsys.com*.

Contents

Preface

About This Book

The Atrenta® Console Reference Guide describes various features, menus, and windows. In addition, this guide describes the configuration file that can be used to change the configuration settings of Atrenta Console.

Contents of This Book

The Atrenta Console Reference Guide has the following sections:

Typographical Conventions

This document uses the following typographical conventions:

To indicate	Convention Used
Program code	`OUT <= IN;`
Object names	`OUT`
Variables representing objects names	*`<sig-name>`*
Message	Active low signal name '<sig-name>' must end with _X.
Message location	`OUT` <= IN;
Reworked example with message removed	OUT_X <= IN;
Important Information	**NOTE:** This rule...

The following table describes the syntax used in this document:

Syntax	Description
[ ] (Square brackets)	An optional entry
{ } (Curly braces)	An entry that can be specified once or multiple times
| (Vertical bar)	A list of choices out of which you can choose one
. . . (Horizontal ellipsis)	Other options that you can specify

SpyGlass Concepts

SpyGlass Features

SpyGlass® provides the following features:

- Full language support for Verilog (IEEE 1364, Verilog 2001, and SystemVerilog) and VHDL, that is VHDL (IEEE Std 1987) and VHDL (IEEE Std 1993).
- A rich suite of in-built rules, including:
  - ❐ File checks, such as file names, design units per file and headers
  - ❐ Naming checks on signals, ports, parameters, constants, clocks and other constructs
  - ❐ Style and related checks
  - ❐ Coding for synthesis and related checks
  - ❐ Design practice and related checks
  - ❐ Area, timing, and synchronization checks
  - ❐ Clock and reset checks
  - ❐ SpyGlass DFT solution, SpyGlass Power Verify solution, SpyGlass Constraints solution, SpyGlass ERC solution, and similar checks (cost options)

  - Support for several industry standard HDL analysis and assessment programs, including OpenMORE^{TM} and STARC^{TM}

- A variety of report format options so you can set up your own reports
- Built-in engines, including RTL synthesis and flattening, to enable detailed implementation tests including clocking, reset and synchronization of asynchronous signals
- A Graphical User Interface (GUI) called Atrenta Console
- A batch execution program for integration in corporate design flows

Additional features enable you to customize SpyGlass to meet your company’s unique requirements. Refer to the *SpyGlass Policy Customization Guide* for more details. Customization features include:

- A Perl interface that provides extensive programmability, and customization of error messages, error severity, and other parameters
- User-created rules built as dynamically linked C libraries
- User-programmable reporting that enables you to generate message reports as screen displays, hard copies, files, e-mail, Web pages, and other formats

SpyGlass top-level is a full Perl interpreter that allows extensive integration, enabling you to take advantage of the open-source libraries for Web programming, GUI management, and many other capabilities.

SpyGlass Operations

SpyGlass functional model is summarized in the following figure:

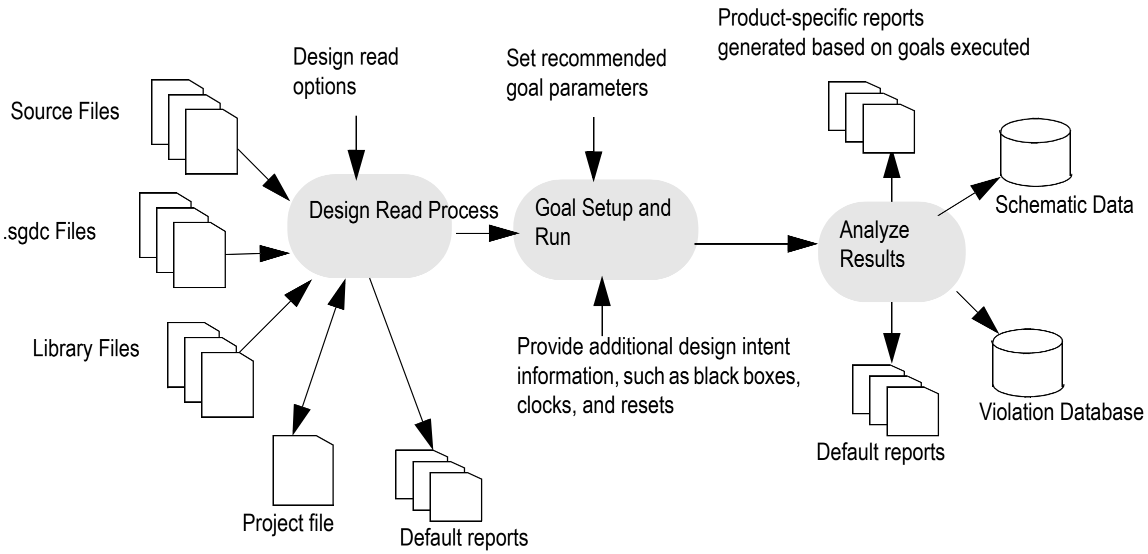


FIGURE 1. SpyGlass Functional Model

How SpyGlass Analyzes Your Design

SpyGlass analyzes your design in various steps, depending upon the goals executed. SpyGlass invokes the analysis tools that meet your request, in order to give you the fastest runtime.

Following are the phases of SpyGlass analysis:

1. SpyGlass interprets precompiler directives.

   SpyGlass first reads and interprets all precompiler directives and `translate_on/translate_off` pragmas.

NOTE: *Pragmas are runtime commands embedded as comments in your source code, and used with specific electronic design automation tools.*

2. SpyGlass checks style/linting rules.

   SpyGlass performs checks for standard style and linting rules. It then logs rule messages and location of each message in the violation database for processing by the report generator. For each unit in the design, SpyGlass first checks for HDL syntax and structural errors and then reports violation of different levels of severity (see section below on SpyGlass error messages).

3. SpyGlass checks inference rules.

   If SpyGlass detects no syntax messages in step *2*, it transparently invokes its RTL synthesis engine. This creates an internal gate-level, hierarchical design built around generic gates that allows SpyGlass to detect inferred logic, such as latches, flip-flops, and counters. The gate-level logic naming is controlled within SpyGlass so any messages found in inferred logic can easily be mapped back to the RTL code that caused it, which simplifies debugging.

4. SpyGlass checks connectivity rules.

   Once SpyGlass has checked the inferred logic, SpyGlass transparently flattens your design so it can check complex connectivity rules. These include issues such as correct synchronization logic at clock domain crossings, reset rules, combinational loop detection, and SpyGlass DFT solution and cone analysis. In effect, this phase can encompass any form of netlist checking. As in the synthesis phase, controlled naming maps any errors SpyGlass detects back to your source code.

5. SpyGlass uses additional analysis engines

   Depending on the complexity of a given rule, SpyGlass will also make use of its internal cycle simulator and/or testability analyzer. This enables advanced connectivity and behavior checking.

SpyGlass identifies and removes duplicate rules in case of multiple selected goals.

If you select multiple goals, the same rule might be present in such goals. In such cases, SpyGlass removes duplicate rules and then runs the rules.

SpyGlass Built-in Checking

While analyzing or synthesizing RTL designs, SpyGlass performs checks on the HDL syntax and structure. These checks are always performed automatically, independently of which SpyGlass rules are requested to be checked.

If any syntax or structure issues are found, SpyGlass generates the corresponding standard error or warning messages (known as *built-in messages*). These built-in messages are different from the rule messages generated during rule-checking.

There are the following classes of such built-in messages:

TABLE 1 Built-in Messages Classes

Message Type	Message Prefix
Syntax errors	STX_
Language Warnings	WRN_
Synthesis warnings	SYNTH_
Synthesis errors	SYNTH_
Post-elaboration syntax errors (VHDL only)	ELAB_

Syntax Errors and Language Warning messages are language-specific, that is, there are separate message sets for Verilog, VHDL, and Mixed-Language respectively. Synthesis warnings and errors are language-neutral for the most part. See the *SpyGlass Built-In Messages Reference* for details of these message sets.

The `ELAB_` messages may appear when a VHDL design is being processed after elaboration. These messages relate to syntax issues and have the same number and content as the syntax messages except they have the `ELAB_` prefix.

NOTE: *Some of the rules in SpyGlass products are mapped to the SpyGlass built-in messages. Thus, when you run SpyGlass with rule-checking, some of the standard warning and error messages are suppressed and equivalent rule messages are generated in their place.*

NOTE: *If SpyGlass detects an internal software error, it reports the error as a* `WARNING` *message. These errors are normally associated with a specific implementation of a* *rule. Any such error messages should be reported to Atrenta Customer Support.*

Processing the HDL Designs

When you run SpyGlass on a design, the following process is followed:

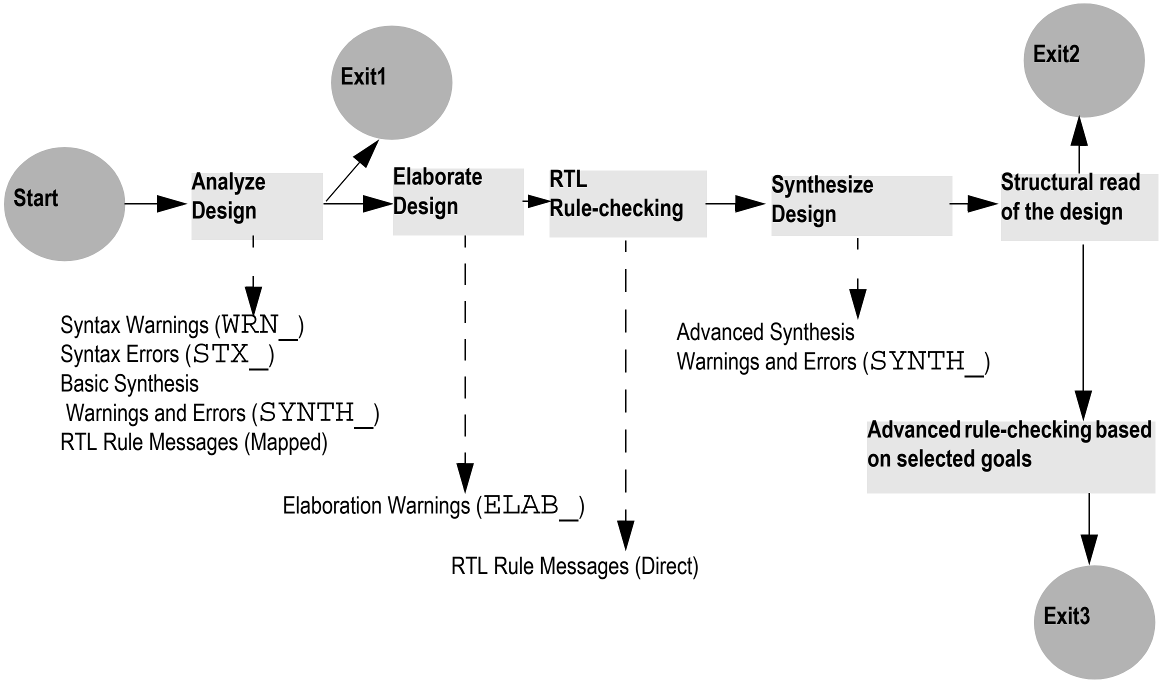

Start
Analyze Design
Exit1
Elaborate Design
RTL Rule-checking
Synthesize Design
Structural read of the design
Exit2
Syntax Warnings (WRN_)
Syntax Errors (STX_)
Basic Synthesis
Warnings and Errors (SYNTH_)
RTL Rule Messages (Mapped)
Elaboration Warnings (ELAB_)
RTL Rule Messages (Direct)
Advanced Synthesis
Warnings and Errors (SYNTH_)
Advanced rule-checking based on selected goals
Exit3

FIGURE 2. SpyGlass Run Process

SpyGlass processing occurs in the following steps:

1. SpyGlass analyzes the design and generates the following types of standard built-in messages:
   - ❒ Syntax warning messages (WRN_ messages)
   - ❒ Syntax error messages (STX_ messages)
   - ❒ Basic synthesis error and warning messages (SYNTH_ messages)

   SpyGlass also generates rule messages (instead of built-in messages mapped to the rules in the selected products)

2. If the design has syntax errors (`STX_` messages) after initial processing, SpyGlass exits (shown as **Exit1**).
3. If the design does not have any syntax errors, SpyGlass may perform elaboration of the RTL design and generate elaboration time errors or warnings (`ELAB_` messages)
4. SpyGlass performs RTL rule-checking and generates rule messages as applicable.
5. SpyGlass synthesizes the design and generates the advanced synthesis error and warning messages (`SYNTH_` messages).
6. SpyGlass performs structural read of the design and then SpyGlass exits (shown as **Exit2**) if no goals are selected.
7. If you have selected goals, SpyGlass runs the rules of the goals, and reports appropriate violation messages.
8. SpyGlass exists (shown as **Exit3**).

SpyGlass Rule Environment

Understanding SpyGlass Rule Definitions

Each SpyGlass product whether standard or custom, is a PERL source file that has the rule definition, the rule group definitions, and PERL subroutines, if any.

Each rule has a number of attributes that decide how the rule will function. Some of the important attributes are as follows:

- *Rule name*, which you use to specify that the particular rule should be run
- *Language* applicable for a rule. This attribute enables you to know if the rule works on Verilog, VHDL, or mixed-language designs.
- *Severity* of the rule, which indicates how important it is to fix your source code
- *Message* displayed if your source code violates the rule
- *Rule primitive* that is called for the rule

Rule Primitives

Rule primitives are C functions that are present either in SpyGlass core or in a shared library. They perform the real work of checking rules against your design and can be parameterized to produce different checks. The primitive called for each rule is specified in the product. However, in normal use, the rule primitives and the rule attributes are transparent to the user. The rule primitive extracts object data from the design database, runs checks as defined by the parameters, and generates a message as defined in the rule, if an error is found.

Rule Types and Order of Execution

SpyGlass rules are categorized to work on a particular design view of your source file. There are rules that work on the source RTL description, rules that work on the synthesized hierarchical netlists, and rules that work on the flattened netlists. Consequently, the rules work in the same order in which the corresponding design becomes available.

The SpyGlass rule types by their order of execution are as follows:

1. SETUP type rules

   The SETUP type rules are checked just before design analysis phase.

   This category contains rules that are independent of HDL source code (Verilog or VHDL) or rules that are checked inside the analysis/synthesis engine.

   The SETUP type rules are as follows:

   ❒ Rules that check for commands (including the product-specific rule parameters) and existence of different custom data requirements.

   ❒ Rules that check for liberty library files (.lib files)

   ❒ Rules that check for SGDC files

   ❒ Enabling of analyzer built-in rules and synthesis built-in rules.

NOTE: *The analyzer built-in rules and synthesis built-in rules are only enabled by standard built-in rule-primitives. Actual checking and rule message generation occurs during actual analysis or synthesis.*

2. RTLALLDULIST type rules

   The RTLALLDULIST type rules operate on all RTL design unit including those from precompiled libraries. For Verilog, a design unit is a module, UDP, or macro-module. For VHDL, a design unit is an entity, architecture, configuration, package, or package body.

   An RTLALLDULIST type rule is checked once on each RTL/Library design unit in the design. These rules can access only the local design unit data. They cannot access hierarchy information.

3. RTLDU type rules

   The RTLDU type rules operate on a single RTL design unit. For Verilog, a design unit is a module, UDP, or macro-module. For VHDL, a design unit is an entity, architecture, configuration, package, or package body.

   An RTLDU type rule is checked once on each design unit in the design. These rules can access only the local design unit data. They cannot access hierarchy information.

   By default, the RTLDU type rules do not run on precompiled design units instantiated in a design. To enable rule checking on such design units, specify the following command in a project file:

   ```
   set_option hdllibdu yes
   ```

4. RTLDULIST type rules

The RTLDULIST type rules work on the complete RTL design (not including the precompiled library design unit), without elaboration.

An RTLDULIST type rule is checked once on the complete design. These rules can access only the local design unit data. They cannot access hierarchy information.

By default, the RTLDULIST type rules do not run on precompiled design units instantiated in a design. To enable rule checking on such design units, specify the following command in a project file:

```
set_option hdllibdu yes
```

5. ELABDU type rules

   The ELABDU type rules work on the elaborated design units.

   An ELABDU type rule is checked once on each design unit, elaborated with unique set of parameters/generics with which it has been instantiated. For example, if a module is instantiated 4 times with a parameter value of 5 and 2 times with a parameter value of 7, the module would be elaborated only twice, (instead of 4+2) once with 5 as the parameter value and once with 7 as the parameter value.

6. RTLTOPDU type rules

   The RTLTOPDU type rules are run on the top design unit, determined by SpyGlass after design elaboration.

7. LEXICAL type rules

   The LEXICAL type rules check on each design file, one at a time. This category generally contains non-electronic and text-based rules, such as line-length, tabs, indentation, naming conventions, etc. By default, the LEXICAL type rules do not run on precompiled RTL files.

   To enable rule checking on such design units, specify the following command in a project file:

   ```
   set_option hdllibdu yes
   ```

8. VSDU type rules

   The VSDU type rules check upon a synthesized object model of a design unit.

   A VSDU type rule is checked once on each synthesized design unit.

9. VSTOPDU type rules

   The VSTOPDU type rules are run on the synthesized object model for each top module hierarchy.

Since the RTL view is not available, such rules' checking is limited to only the synthesized object model.

10. BLOCKDU_CD type rules

The BLOCKDU_CD type rules run on user-specified design units flattened down to the specified flattened partitions (called *blocks*) and the blocks themselves.

In this case, design units are flattened only till block boundaries and each block in turn is flattened till boundaries of its sub-blocks, if any.

The BLOCKDU_CD type rules run on each specified design unit and each block once.

Consider the following example where the design unit `top` has a partition named `blk1`:

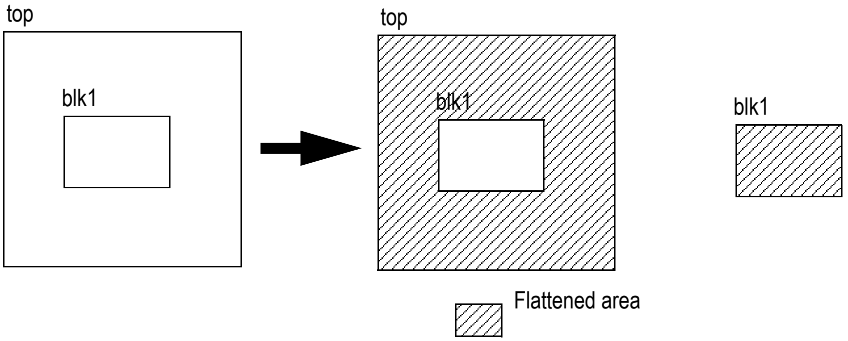


FIGURE 3. Processing with BLOCKDU_CD type rules

In this case, each BLOCKDU_CD type rule is run once on the design unit `top` flattened up to the block `blk1` boundary and once on the flattened block `blk1`.

11. FLATBLOCKDU type rules

The FLATBLOCKDU type rules run on user-specified flattened design units and only the specified flattened partitions (or blocks) within these design units.

In this case, design units and blocks are completely flattened until leaf-level.

The FLATBLOCKDU type rules run on each user-specified flattened design unit and each specified block under the design unit once.

Consider the following example where the design unit `top` has a partition named `blk1`:

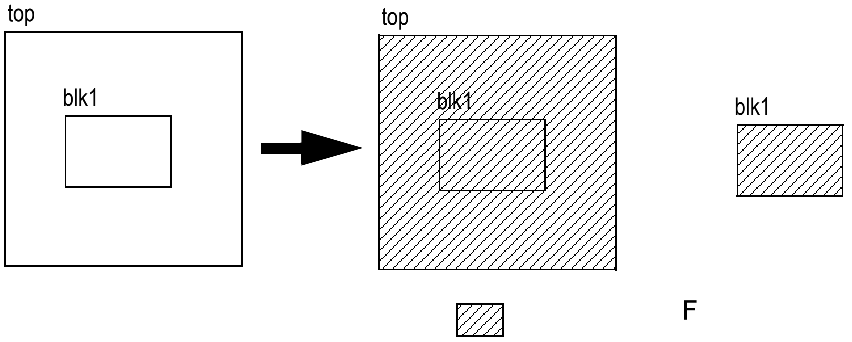


FIGURE 4. Processing with FLATBLOCKDU type rules

In this case, each FLATBLOCKDU type rule is run on the design unit `top` flattened down to leaf-level and on block `blk1` flattened down to leaf-level.

12. FLATDU type rules

    The FLATDU type rules run on the Flat Object Model for each top module in design.

13. FLATDU2_WOL type rules

NOTE: *The FLATDU2_WOL type has been deprecated. Rules registered with this view are internally moved to the FLATDU2_WL view.*

14. FLATDU2_WL type rules

    The FLATDU2_WL type rules are designed to work with SpyGlass Logic Evaluator.

    In this view, .lib instances are not flattened. In addition, .lib functionality is not visible to these rules. Unlike the FLATBLOCKDU type rules, flattening of such rules is applied to the top-level design units only.

    SpyGlass ERC solution has rules of this type, which just use electrical information of a cell but not cell functionality.

15. ALLVIEWS_WL type rules

The ALLVIEWS_WL type rule category is a special deprecated case. The rules of this category require all design views (RTL, Synthesized, and Flattened) in memory and are run after FLATDU2_WL type of rules and before FLATDU type of rules. From flattening point of view, they are similar to FLATDU2_WL type of rules.

16. ALLVIEWS type rules

The ALLVIEWS type rule category is a special deprecated case. The rules of this category require all design views (RTL, Synthesized, and Flattened) in memory and are run only after rules of all other categories have been run.

SpyGlass Standard Products

SpyGlass analyzes HDL source files using pre-defined rules contained in product files. A number of SpyGlass Standard products are installed with SpyGlass.

The SpyGlass Base products that are available with SpyGlass are:

- **SpyGlass lint solution**, which is based on the commonly accepted set of HDL rules for detecting syntax errors and common connectivity errors such as unused signals and undriven inputs.
- **SpyGlass OpenMore solution**, which is based on the OpenMORE standard developed by Synopsys and Mentor Graphics, and which sets forth certain coding styles and practices that enhance the reusability of HDL design modules.
- **SpyGlass STARC/STARC02/STARC05 solution**, which is based on IP Reuse guidelines compiled by Semiconductor Technology Academic Research Center (STARC).
- **SpyGlass area solution**, which includes rules used to identify HDL that may cause potential problems in silicon area downstream.
- **SpyGlass latch solution**, which includes rules related to latch based designs
- **SpyGlass miscellaneous solution**, which includes useful rules that cannot be classified in any other default products.
- **SpyGlass timing solution**, which includes rules to identify potential timing issues in the design.

- **SpyGlass ERC solution**, which checks a design for correct electrical connectivity at the gate level.

In addition, there are Atrenta Advanced products available that can be purchased separately:

- **SpyGlass DFT Solution**, which checks a design for testability issues at the RT level.
- **SpyGlass Constraints Solution**, which checks the suitability of the Synopsys Design Constraints files
- **SpyGlass Power Verify Solution**, which checks the design for the structural, architectural, and system level issues in order to reduce power.
- **SpyGlass CDC Solution**, which include rules to check clocks, resets, and clock-domain-crossings in the design
- **SpyGlass Power Estimate Solution**, which include rules for power estimation and power reduction
- **SpyGlass TXV Solution**, which is used to verify correctness of timing exceptions and/or check if critical paths can be ignored or relaxed during timing sign-off.

Customizing SpyGlass Rules

SpyGlass allows you to customize a number of its functions to meet your company’s needs. For example:

- Add your own rules or modify the pre-defined rules making use of the pre-defined primitives.

  See the *SpyGlass Policy Customization Guide* for more details.
- Modify the SpyGlass defaults for rule severities, messages, allowed name syntax, or other parameters.

  See the *SpyGlass Policy Customization Guide* for more details.
- Create custom reports

Using Rule Mnemonics

Rule mnemonics refer to the custom rule names assigned to the SpyGlass rule. These custom rule names are more descriptive and convey additional details about a rule.

This section describes the following topics:

- *Enabling the Support for Rule Mnemonics*
- *Converting the Rule Names*
- *Viewing the Rule Help*

Enabling the Support for Rule Mnemonics

By default, the support for rule mnemonics is disabled. To enable this feature, specify the *enable_rule_mnemonic* option in the project file, as shown below:

```
set_option enable_rule_mnemonic <yes | no | 0 | 1>
```

When this option is enabled, SpyGlass uses the custom names specified in the rules_mapping.csv file, throughout the SpyGlass run.

The mapping of the existing SpyGlass rules and the rule mnemonics is specified in the rules_mapping.csv file, which is available at the following location:

```
SPYGLASS_HOME/auxi/rules_mapping.csv
```

NOTE: *If the enable_rule_mnemonic option is enabled, you must only specify the rule mnemonic throughout the SpyGlass run. That is, you must use the rule mnemonics in all the input files, such as, project files, Tcl files, and so on. Otherwise, SpyGlass will not process the rule.*

Following is the format for a rules_mapping.csv file:

```
<existing_rule_name>,<descriptive_rule_name>,<product_name>,
"<comment>"
```

Converting the Rule Names

To ease the process of replacing SpyGlass rule names with the corresponding rule mnemonics in the input files, you can use convert_rule_names.pl file, which is available at the following location:

```
SPYGLASS_HOME/auxi/convert_rule_names.pl
```

This script uses the rules_mapping.csv file to check the existing rule mapping and replaces the SpyGlass rule names with the rule mnemonics accordingly.

Specify the following command to run the convert_rule_names.pl script:

```
perl SPYGLASS_HOME/auxi/convert_rule_names.pl  -i
<input_file>  -o <output_file>
```

Where,

- `<input_file>` refers to any input file in the SpyGlass run, such as, .prj file.
- `<output_file>` refers to the output file, containing the converted rule names.

For more information on the covert_rule_names.pl script, specify the following command:

```
perl SPYGLASS_HOME/auxi/convert_rule_names.pl  --help
```

Viewing the Rule Help

You can view the help of the SpyGlass rules and the corresponding rule mnemonics using the following ways:

- **HTML Help**: Enables you to search for a rule mnemonic and displays the help for the corresponding rule
- **The spyexplain Utility**: Enables you to view the short and long help for both the SpyGlass rules and rule mnemonics

Processing Messages and Displaying Reports

Based on the selected goals, SpyGlass reports appropriate violation messages to indicate various design issues. These messages are written to in report files and SpyGlass log file.

About Rule Severity

SpyGlass supports two levels of rule severities - severity label and severity class.

Each SpyGlass rule has a severity label and each severity label is classified under one of the predefined severity classes. Severity labels are product-specific and severity classes are predefined for SpyGlass.

After SpyGlass analysis run, a rule-checking summary is printed that reports total rule messages found under each rule severity-label and total rule messages found under each predefined rule severity-class that is, the sum of total rule messages found under each rule severity-label classified under that rule-severity class.

Predefined Rule Severity-Classes

SpyGlass supports the following five rule severity-classes:

FATAL Rule Severity-Class

The FATAL rule severity-class is primarily used for a stopper situation related to design rule-checking. Occurrence of a rule message of this class aborts further rule-checking by SpyGlass and requires immediate attention from the user so that rule-checking can be resumed.

Examples of messages of the FATAL rule severity-class are as follows:

1. Syntax Error in the input design source

   Almost all syntax errors cause immediate termination of further rule-checking by SpyGlass since syntax error indicates inability to create a consistent design view that is required by later stages of rule-checking by SpyGlass. Hence, such messages are identified as FATAL class messages. However, certain simulation-related syntax errors may be classified as non-FATAL messages because such errors do not affect the synthesis-oriented design view that is required by SpyGlass even when the constructs are syntactically incorrect as per LRM specifications.

2. Custom product assumptions about the design that are found to be invalidated at runtime

   One example of such assumption is when a custom product requires that `assign` statements are not used in the Verilog design source. Then, there would be a rule in the product that will run first and flag all uses of `assign` statements. If this rule is violated, that is, if such statement are actually found in a design, further rule-checking cannot be performed because all subsequent rules do not take the Verilog `assign` statement functionality into account.

In a large majority of SpyGlass runs, FATAL rule severity-class messages are not expected. Occurrence of such messages indicates an interrupted run and would normally require re-run of SpyGlass after the reported problem has been fixed.

A rule of FATAL rule severity-class is always run completely. However, if a rule message is found for a rule of FATAL rule severity-class, no further rule-checking is performed.

ERROR Rule Severity-Class

The ERROR rule severity-class is normally used to indicate a design error that would cause design functionality to be compromised. Normally, such messages indicate an immediate bug in the design from the perspective of design-integrity aspect being analyzed by the SpyGlass run. Presence of such ERROR rule severity-class messages require user to fix the design after suitable analysis. Applying waiver on an ERROR rule severity-class message may require detailed justification and approval in the SpyGlass use-model adopted by a typical user.

In contrast, other lower-precedence messages indicate only a potential error situation or non-compliance with design development guidelines that do not directly affect design functionality.

Normally a few of these ERROR rule severity-class messages would be expected on a typical in-development design. For a known good design, ERROR rule severity-class messages will be few and typically report only those design errors that are not verifiable by traditional simulation and other verification procedures.

WARNING Rule Severity-Class

The WARNING rule severity-class is the next less-severe message class after the ERROR rule severity-class. As described earlier, a key characteristic of such messages is that these messages indicate only a potential error situation or non-compliance with design development guidelines that do not directly affect design functionality.

Therefore, you can apply selective waivers on such messages with relatively more freedom (and confidence) than say, an ERROR rule severity-class message.

However, from SpyGlass perspective, such messages still carry a hint of caution. Therefore, you should not ignore such messages without proper analysis.

INFO Rule Severity-Class

The INFO rule severity-class represents all SpyGlass rule-checking output that is of informative nature. Such output either may be general design statistics or may represent any kind of design query that a user/product may wish to perform.

Hence, while the count of FATAL, ERROR, or WARNING rule severity-class messages can be used as a measure of design quality, reporting of INFO rule severity-class output is not expected to be a measure of design quality. Instead, it will depend on the design size and amount of design statistics and query data that a user is intending to extract.

DATA Rule Severity-Class

The DATA rule severity-class represents SpyGlass output not belonging to any of other rule severity-classes. Such cases may be secondary data to debug any of preceding rule messages or may represent information that is not reported as part of usual SpyGlass message reports or GUI display.

Typically, you would not be aware of this output. Such output is intended for better diagnostics and usability support within SpyGlass environment. You should not assume anything about the existence or other details of this data.

Rule messages of DATA severity class are not displayed in SpyGlass reports except in the *Session Log Page* of *The Message Window*, where the total reported messages are categorized according to the above-predefined rule severity classes (a category is shown in the Session Log Page only if any message is reported for that category).

SpyGlass Results Summary

At the end of SpyGlass analysis run, a results summary is generated as shown in the following example:

```
------------------------------------------------------------------
Results Summary:
-----------------------------------------------------------------
Goal Run  : spyglass_cmdline_goal
                     (Rules from above goal(s) have been changed)
Command-line read  :   0 error,   1 warning,  0 information message
Design Read        :   0 error,    0 warning,  3 information messages
Found 2 top modules: cdc   (file: ../../Ac_cdc08/cdc.v)
                     ovl   (file: ../../Ac_cdc08/ovl.v)

** black box Resolution: 50 errors, 0 warning, 0 information message
SGDC Checks        :      0 error,  3 warnings, 0 information message
Policy clock-reset :     1 error,  4 warnings, 0 information message
-----------------------------------------------------------------
Total              :    51 errors, 8 warnings, 3 information messages

Total Number of Generated
Messages                    : 63 (51 errors, 8 warnings, 4 Infos)
Number of Waived Messages   : 1  (0 error, 0 warning, 1 Info)
Number of Reported Messages : 62 (51 errors, 8 warnings, 3 Infos)

NOTE: It is recommended to first fix/reconcile fatals/errors
reported on lines starting with ** as subsequent issues might be
related to it.
Please re-run SpyGlass once ** prefixed lines are fatal/error
clean.


-----------------------------------------------------------------
SpyGlass Exit Code 0 (Rule-checking completed with errors)
```

The results summary contains the following information:

- Count of FATAL, ERROR, WARNING, and INFO severity class messages.

The count of FATAL severity class message is displayed first in the results summary. This column is not printed if there are no FATAL severity-class messages.

If there is no message for any of the ERROR, WARNING, and INFO severity classes, the corresponding count set is reported as zero.

- Count of total number generated messages, the waived messages, and the reported messages is displayed as in the following example:

```
Total Number of Generated Messages : 3 (2 errors, 0 warning, 1 Info)
Number of Reported Messages        : 3 (2 errors, 0 warning, 1 Info)
Number of Waived Messages          : 1(0 errors, 1 warning, 0 Info)
```

If you have set a rule message-reporting limit by using the *lvpr* command and some messages are suppressed, the count of suppressed messages is reported. An example is shown below:

```
...
Number of Overlimit Messages :  25
(2 errors, 11 warnings, 12 Infos
...
```

- SpyGlass exit status

For details, see *SpyGlass Exit Status*.

SpyGlass Exit Status

For better integration with other stream tools, SpyGlass generates exit status code that gives you the exact status of SpyGlass run.

By default, SpyGlass reports an exit code of 0 for a successful run and prints one of the following messages:

- `SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)`
- `SpyGlass Exit Code 0 (Rule-checking completed with warnings)`
- `SpyGlass Exit Code 0 (Rule-checking completed with errors)`
- `SpyGlass Exit Code 0 (Informational command executed, rule-checking not done)`

You can set the *enable_pass_exit_codes* command to `yes` in a project file to report different exit codes for each of the above cases depending on the type of message generated in the current run. For details, see *Messages Printed When SpyGlass Run is Complete*.

NOTE: *Waived messages are not considered while deciding the exit status. Only reported messages are considered.*

SpyGlass may exit abruptly if it is unable to set the stack size to unlimited.

By default, Spyglass internally sets the stack size to unlimited. However, if it is unable to do so due to insufficient permissions then the following warning message appears:

```
WARNING: Stacksize could not be set as unlimited through ulimit
command.This may lead to nondeterministic tool behavior
```

In such cases, contact system administrator to check for permissions to change stack size.

Messages Printed When SpyGlass Run is Complete

This section describes the exit codes when the SpyGlass run is complete.

These exit codes appear when you set the *enable_pass_exit_codes* command to `yes`. If you set this command to `no`, SpyGlass reports the default exit code of 0 in place of these exit codes, as described below.

- `SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)`

  The above message appears when a SpyGlass run is complete without any error or warning messages.

  Typically, this situation indicates that your design is clean with respect to the executed goals.

- `SpyGlass Exit Code 11 (Rule-checking completed with warnings)`

  The above message appears when a SpyGlass run is complete without any error message but with warning messages.

  This status indicates that some rules of the warning severity are reported. See *WARNING Rule Severity-Class* for understanding and handling warning messages.

- `SpyGlass Exit Code 12 (Rule-checking completed with errors)`

  The above message appears when a SpyGlass run is complete with error messages (and possibly warning messages).

  This status indicates that some rules of the error severity are reported. See *ERROR Rule Severity-Class* for understanding and handling error messages.

- `SpyGlass Exit Code 20 (Informational command executed, rule-checking not done)`

  The above message appears if you have executed an informational option of SpyGlass.

  Typically, no further action is required for this exit status.

Messages Printed When SpyGlass Run is not Complete

This section describes the exit codes when the SpyGlass run is not complete.

SpyGlass reports these exit codes irrespective of the value of the *enable_pass_exit_codes* command.

- `SpyGlass Exit Code 1 (Abnormal termination - termination not trapped by software)`

  The above message appears when SpyGlass run is terminated because of an abnormal error that is not trapped by SpyGlass.

  This exit status indicates some operating system-related problem, such as sack overflow and out of memory issue.

  In such cases, the SpyGlass log file may also be incomplete as SpyGlass was not able to trap an error signal and report suitably in the log file.

  In such cases, check available machine resources and do the required correction. If the problem persists, report the problem to Atrenta Support.

- `SpyGlass Exit Code 3 (Abnormal termination trapped by software)`

  The above message appears when SpyGlass run is terminated because of an error that is trapped by SpyGlass.

  This exit status indicates that some operating system-related problem, such as segmentation fault or memory corruption has occurred but SpyGlass was able to trap the error signal and report suitably in log file.

  In such cases, check the gdb trace and stack trace printed in the SpyGlass log file and take corrective action.

  After SpyGlass traps the error signal, it tries to generate the moresimple report that contains error messages and/or rule violations reported before the end of run. If the moresimple report is generated, it may contain useful information, which may help in debugging the situation. Along with the error details given in the SpyGlass log file, send the moresimple report to Atrenta Support.

- `SpyGlass Exit Code 4 (License failure, rule-checking aborted)`

  The above message appears when a SpyGlass run is terminated because of a license failure.

  Check the license status and take a corrective action.

- `SpyGlass Exit Code 5 (Rule-checking interrupted by User)`

  The above message appears when you forcibly terminate a SpyGlass run by killing the corresponding process.

In this case, SpyGlass-generated results may be incomplete and should not be used.

- `SpyGlass Exit Code 6 (Rule-checking terminated due to FATAL errors - design syntax error)`

  The above message appears when a SpyGlass run is terminated because of a fatal design error (syntax errors). See *FATAL Rule Severity-Class* for understanding and handling fatal messages.

  In this case, check design inputs and take a corrective action.

- `SpyGlass Exit Code 7 (Rule-checking terminated due to FATAL errors - usage or run error)`

  The above message appears when a SpyGlass run is terminated because of incorrect usage and incorrect/incomplete inputs. See *FATAL Rule Severity-Class* for understanding and handling fatal messages.

  In this case, check the specified inputs and take a corrective action.

- `SpyGlass Exit Code 8 (Design database save failure, rule-checking aborted)`

  The above message appears when a design cannot be saved because of reasons, such as incorrect save database directory.

  In this case, rectify the save and restore-related commands.

- `SpyGlass Exit Code 137 (run killed by user, or by system due to lack of resources like memory etc.)`

  The above message appears when a signal is sent to SpyGlass process to terminate the session immediately.

  The signal is sent either by the user or by the operating system because of lack of resources, such as memory.

  In this case, run SpyGlass on a machine that has higher memory.

- `SpyGlass Exit Code 153 (File size limit exceeded)`

  The above message appears when a signal is sent to SpyGlass process to terminate the session if size of files generated by SpyGlass exceeds the maximum limit allowed by the operating system.

  In this case, increase the maximum allowed size of a file to enable SpyGlass process to complete successfully.

The Menu Bar in Atrenta Console

Overview

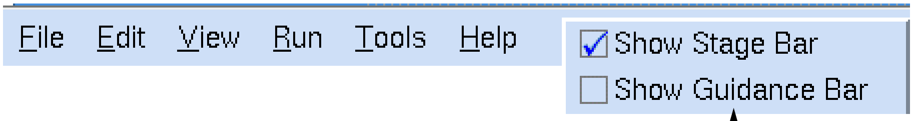
This chapter describes the menu bar in the Atrenta Console GUI.

The Menu Bar

The Atrenta Console menu bar contains the following menus:

File Menu	*Edit Menu*	*View Menu*	*Run Menu*
Tools Menu	*Help Menu*		

In addition to the above menus, the following shortcut menu appears if you right-click on the right side of the menu bar:



Shortcut menu

FIGURE 1. The Atrenta Console Menu Bar

From the above shortcut menu,

- If you de-select the *Show Stage Bar* option, all the three stage tabs appear parallel to the menu bar, as shown in the following figure:

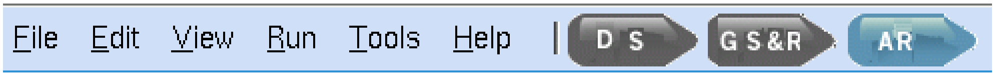


FIGURE 2. Deselect Show Stage Bar Option

- If you select the Show Guidance Bar option, the guidance bar appear. For details, see *View > Show Guidance Bar*.

File Menu

When you click the *File* menu, the following options appear:

File > New Project	*File > Open Project*
File > Reload Project	*File > Close Project*
File > Save Project	*File > Save Project As*
File > Import Source(s)	*File > Project Properties*
File > Print File	*File > Exit*

File > New Project

This menu option is used to create a project (.prj) file.

NOTE: *When you invoke Atrenta Console, SpyGlass automatically opens a new* Untitled - 1.prj *project file.*

When you select this menu option, the *Save Project File* dialog appears. You can type the name of the project file in the *File Name* text field and navigate to the directory where you want to save the file. Next, click *Save* to save the project file in the selected directory.

You can also open a new project by pressing the <*Ctrl*>+<*N*> key combination on your keyboard.

File > Open Project

This menu option is used to open an already saved project (.prj) file.

When you select this menu option, the Open File window appears. You can navigate through your directory structure to locate the required goal file. Open the file by either double-clicking the file or selecting the file and then clicking *Open*.

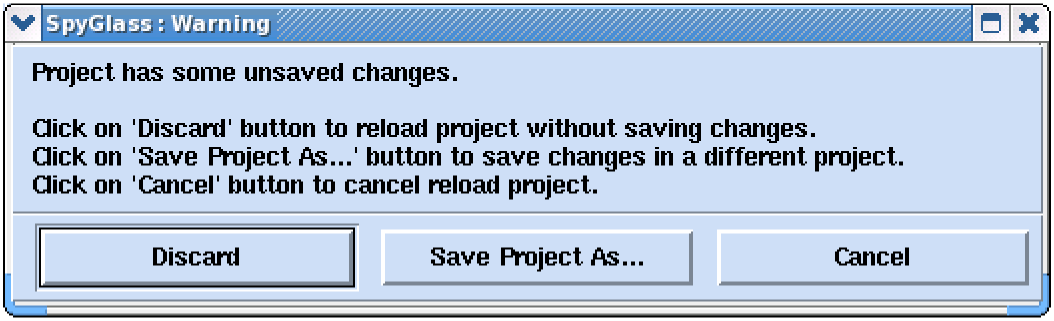
You can also open an already saved project by pressing the <*Ctrl*>+<*O*> key combination on your keyboard.

NOTE: *When you open a saved project file, the stage that you were working the last time when you saved the project opens automatically.*

File > Reload Project

This menu option is used to reload the current project if that project has been modified through an external editor.

When you select this menu option, the following *Warning* dialog appears:



SpyGlass : Warning

Project has some unsaved changes.

Click on 'Discard' button to reload project without saving changes.
Click on 'Save Project As...' button to save changes in a different project.
Click on 'Cancel' button to cancel reload project.

FIGURE 3. Unsaved Project Warning Message

In the above dialog:

- If you click the *Discard* button, all unsaved changes in the current project are lost the project is reloaded.
- If you click the *Save Project As*, the *Save Current Project File* dialog appears in which you can save the current project in a different project file before reloading the current project.
- If you click the *Cancel* button, the reload operation aborts.

File > Close Project

This menu option is used to close an already open project.

File > Save Project

This menu option is used to save a currently open project.

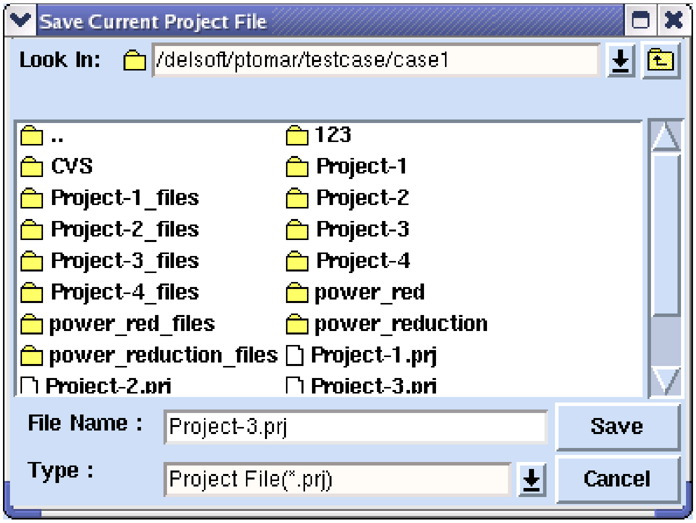
When you click this menu option, the project is saved in the same location from where it was opened.

You can also save a project by pressing the <*Ctrl*>+<*S*> key combination on your keyboard.

File > Save Project As

This menu option is used to save the project with a different name and to a different location.

When you select this menu option, the *Save Project File* dialog appears as shown in the following figure:



Look In: /delsoft/ptomar/testcase/case1

File Name : Project-3.prj

Type : Project File(*.prj)

FIGURE 4. Save Current Project File

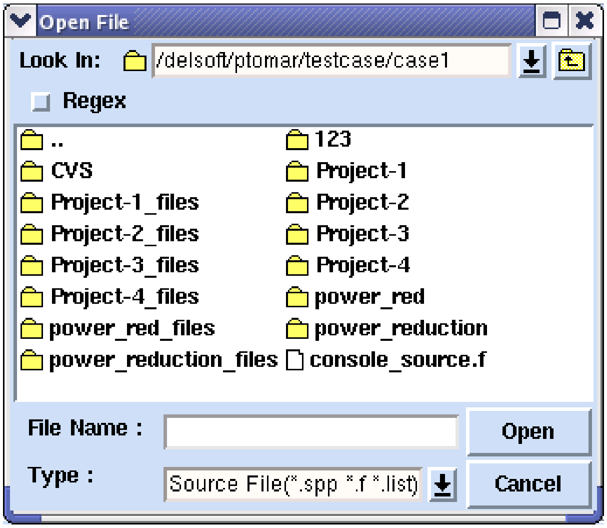
When you save an existing project with a different name, SpyGlass names the project as Project -1.prj. You can save the project by this name or type the name of the project file in the *File Name* text field and navigate to the directory where you want to save the project file. Next, click *Save* to save the project file in the selected directory.

NOTE: *If a project by the name Project -1.prj already exists, SpyGlass names the project as Project-2.prj and so on.*

File > Import Source(s)

This menu option is used to import an existing SpyGlass profile (.spp) file that contains the setup information from an earlier SpyGlass run, such as the current working directory, VHDL library mapping information, and so on.

When you select this menu option, the *Open File* window appears, as shown in the following figure:



Look In: /delsoft/ptomar/testcase/case1

..
CVS
Project-1_files
Project-2_files
Project-3_files
Project-4_files
power_red_files
power_reduction_files
123
Project-1
Project-2
Project-3
Project-4
power_red
power_reduction
console_source.f

File Name :

Type : Source File(*.spp *.f *.list)

FIGURE 5. Open File

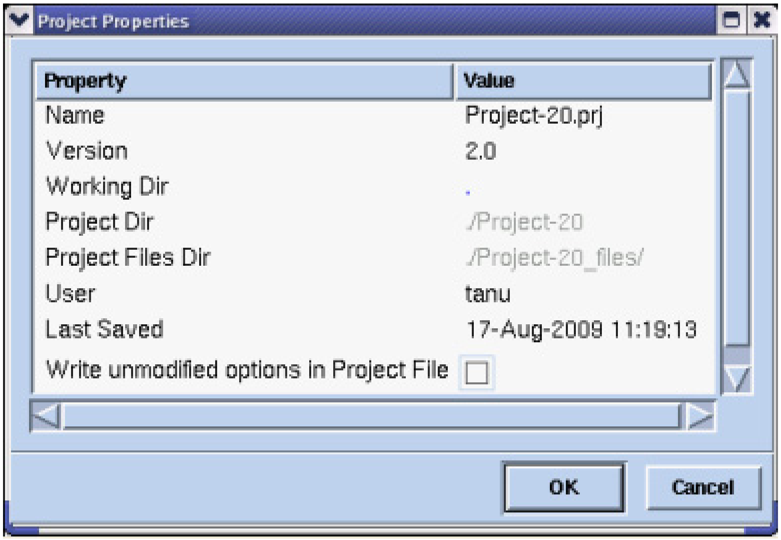
You can navigate through your directory structure to locate the .spp file that contains the information you want to analyze.

When you open an existing profile all the option settings propagate to their proper positions in Atrenta Console.

File > Project Properties

This menu option is used to view the project properties, such as the name of the project file, version, the directory in which the project is saved, name of the user working on the project, and the date on which the project was last modified.

When you select this menu option, the *Project Properties* dialog appears, as shown in the following figure:



Property	Value
Name	Project-20.prj
Version	2.0
Working Dir	.
Project Dir	./Project-20
Project Files Dir	./Project-20_files/
User	tanu
Last Saved	17-Aug-2009 11:19:13
Write unmodified options in Project File	□

FIGURE 6. Project Properties

The *Project Properties* dialog contains two columns, *Property* and *Value*. The *Property* column displays various project properties, and the *Value* column displays the value associated with a property.

You can change the following project properties in the *Project Properties* dialog:

- *Working Directory*

  This is the directory where the project file (.prj) is saved. You can specify a different directory by clicking *Working Dir* and typing the new location of the directory in the provided text field or by browsing to the location where you want to save the project file.

- *Write unmodified options in Project file*

  Selecting this option will automatically write the design read options that you have not modified in the Tcl-based project file in the Common Read Options section.

File > Print File

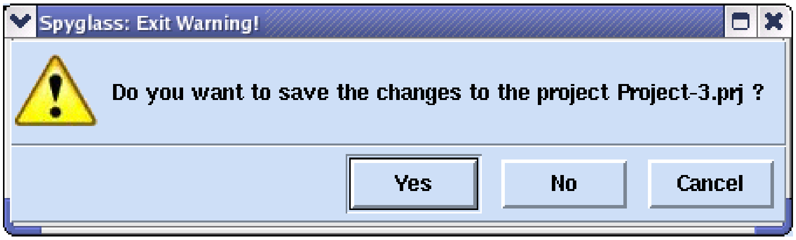
This option prints the currently displayed RTL source file.

When you select this menu option with a valid RTL file displayed, the *Print Command* dialog appears. Set the print command and click *Print*.

The displayed RTL source file is printed.

File > Exit

The menu option is used to exit SpyGlass. When you select this option, the SpyGlass prompts you to save the current project.



Spyglass: Exit Warning!

Do you want to save the changes to the project Project-3.prj ?

FIGURE 7. SpyGlass Exit Warning

Click *Yes* to save the changes and exit SpyGlass. Click *No* to exit SpyGlass without saving the changes. Click *Cancel* to return to the Atrenta Console user interface.

You can also exit SpyGlass by pressing the <*Ctrl*>+<*Q*> key combination on your keyboard.

Edit Menu

When you click the *Edit* menu, the following options appear:

Edit > Clear All Selection	*Edit > Display Line Numbers*
Edit > Edit Current File	*Edit > Reread HDL*
Edit > Search	*Edit > Goto Next Message*
Edit > Goto Previous Message	*Edit > Goto Next HDL Line*

Edit > Clear All Selection

This menu option allows you to clear all current selections in Atrenta Console.

You can also clear all items by pressing the <*Shift*>+<*C*> key combination on your keyboard.

Edit > Display Line Numbers

This menu option is used to toggle on/off line numbering.

When you select this menu option, line numbering will be toggled on or off in the HDL Viewer.

You can also toggle line numbers by pressing the <*L*> key on the keyboard.

Edit > Edit Current File

This menu option is used to edit your source files (both Verilog/VHDL files and SpyGlass Design Constraints files).

When you select this menu option, Atrenta Console launches your text-editing program of choice, as defined by the *Specify external editor program* setting in the *Miscellaneous Page* of the *Tools > Preferences* menu option. If you have not specified a text editor in this setting, the text editor pointed to by the EDITOR environment variable is invoked. Otherwise, SpyGlass attempts to invoke the commonly-used text editor of your operating platform.

The file currently being viewed in the HDL Viewer will open for editing.

NOTE: *If you have set gedit, kedit, or kwrite as the editor of your choice (as defined by the Specify Editor program setting in the Misc Page of the Tools > Preferences... menu option), the cursor in the editor program will not be set to the line highlighted in the HDL Viewer. Note that this behavior is specific to editors, such as gedit, kedit, and kwrite. However, if VI or GVIM is set as editor, the cursor in the editor is automatically set to the line highlighted in the HDL Viewer of SpyGlass.*

You can also launch your editor to modify the current file by pressing the <*E*> key on the keyboard.

Edit > Reread HDL

This menu option is used to re-load the HDL file on the *HDL Viewer*. If someone has edited the HDL file, this menu option enables you to load the latest HDL file in the *HDL Viewer*.

You can also use the <*R*> key on the keyboard to load the HDL file in the HDL Viewer.

Edit > Search

This menu option is used to search for source files, files in the Session Log, Msg Tree, and Msg Summary pages. When you select this option, the *Search* dialog appears as shown in the following figure:

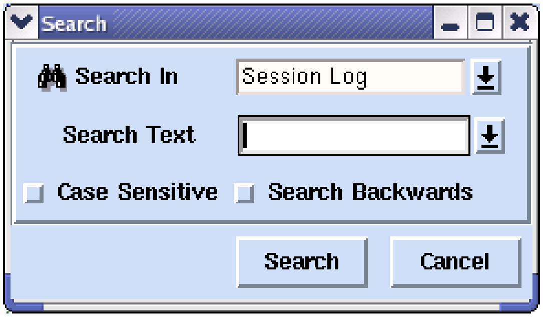


FIGURE 8. Search

To initiate the search operation, perform the following steps:

1. Select an appropriate option from the *Search In* drop-down list where you want to perform the search.
2. Specify the text to be searched in the *Search Text* field.
3. Refine your search by selecting the *Case Insensitive* and/or *Search Backwards* options.

   Details of these options are given in the following table:

Option Name	Description
Case Insensitive	Select this option to search for each occurrence of the specified text irrespective of the casing.
Search Backwards	Select this option to search for the specified text from the current point towards the top of your source file.

You can also open the Search dialog by pressing the *<Ctrl>*+*<F>* key combination on the keyboard.

Edit > Goto Next Message

This menu option is used to go to the next message in the design. The source code line with the next message is also highlighted.

You can also go to the next message in the source code by pressing the *<N>* key on the keyboard. Additionally, you can view the relevant message description, which is highlighted in the Msg Tree page of the Message window, although you may need to expand the message tree to view the highlighted message.

NOTE: *You can also view all the messages associated with the highlighted source code line in the Line page of the Message window.*

Edit > Goto Previous Message

This menu option is used to go to the previous message in the design.

The source code line with the previous message is also highlighted.

You can also go to the previous message in the source code by pressing the *<P>* key on the keyboard. Additionally, you can view the relevant message description, which is highlighted in the Msg Tree page of the Message window, although you may need to expand the message tree to view the highlighted message.

Edit > Goto Next HDL Line

This menu option is used to highlight the next source code line with messages after the source code line at the cursor location.

You can also advance to the next source code line with messages in the source code by pressing the <*M*> key on the keyboard.

View Menu

When you click the *View* menu, the following options appear:

View > Show Stage Bar	*View > Show Guidance Bar*
View > Design Setup	*View > Goal Setup & Run*
View > Analyze Results	

View > Show Stage Bar

This menu option is used to show or hide tabs of different stages in Atrenta Console. These tabs are *Design Setup*, *Goal Setup & Run*, and *Analyze Results*.

View > Show Guidance Bar

This menu option is used to show or hide the guidance bar that guides you to perform the required task at a particular stage.

The following figure shows the guidance bar:

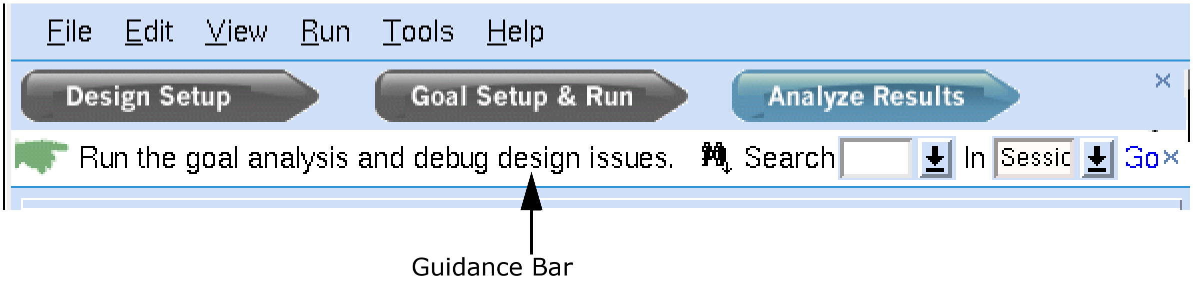


FIGURE 9. Show Guidance Bar

View > Design Setup

This menu option is used to show the page under the *Design Setup* tab.

View > Goal Setup & Run

This menu option is used to show the page under the *Goal Setup & Run* tab.

View > Analyze Results

This menu option is used to show the page under the *Analyze Results* tab.

Run Menu

When you click the *Run* menu, the following options appear:

Run > Design Read	*Run > Analysis*

Run > Design Read

This menu option is used to run SpyGlass analysis in the Design Setup stage to clear any syntactical errors present in the design.

When you select this option, Atrenta Console starts the analysis with your current setup. You can view the SpyGlass status in the Session Log window as analysis is performed.

Refer to the *Running Design Read* section in the *Atrenta Console User Guide* for more details.

Run > Analysis

This menu option is used to start SpyGlass analysis in Analyze Results stage.

When you select this option, Atrenta Console starts the analysis with your current setup. You can view the SpyGlass status in the Session Log window as analysis is performed.

Tools Menu

When you click the *Tools* menu, the following options appear:

Tools > Report	*Tools > Property Manager*
Tools > Waiver Editor	*Tools > Text Viewer*
Tools >Power Intent View	*Tools > View Logfiles*
Tools > Aggregated Project Results	*Tools > Datasheet Report*
Tools > Dashboard Report	*Tools >Methodology Configuration Window*
Tools > Preferences	

Tools > Report

When you click the *Report* menu, the following options appear:

- The *Default* menu option that lists standard SpyGlass reports. For details, refer to the *Default Reports* topic.
- One menu option for each product that was run and for which reports can be generated.

  Refer to the respective rules reference documentation to know more about the product-specific reports.

Selecting the corresponding menu option displays a new window containing the corresponding report. Each windows provides the *Save* and *Print* options to save or print the report.

NOTE: *All those reports that do not have any relevant information appear disabled in the available list of reports. Atrenta Console provides a tool-tip for such disabled reports explaining the reason of not generating that report.*

To find a string in the displayed report, specify that string in the *Find* textbox in the report window.

When you click Find in the report window, the *Case Insensitive* and *Search Backwards* options appear. Select any or both of these options to make your search case-sensitive or case-insensitive, forward and/or backward in the report window. You can also search by specifying regular expressions in the *Find* textbox.

Tools > Property Manager

This menu option is used to view the assertion/constraint details of the SpyGlass Auto Verify and SpyGlass CDC product.

To know more about the Property Manager, refer to *Enabling and Disabling Assertions* section in the *Auto Verify Rules Reference Guide* or the *CDC Rules Reference Guide*.

Tools > Waiver Editor

This menu option is used to waive messages displayed in the *Results* pane.

When you select this menu option, the *SpyGlass Waiver Editor window* appears. Alternatively, hit the <*F*> key on the keyboard to display this window.

For details on this window, see *Waiver Editor Window*.

Tools > Text Viewer

This menu option is used to view text files in the Atrenta Console Text file viewer.

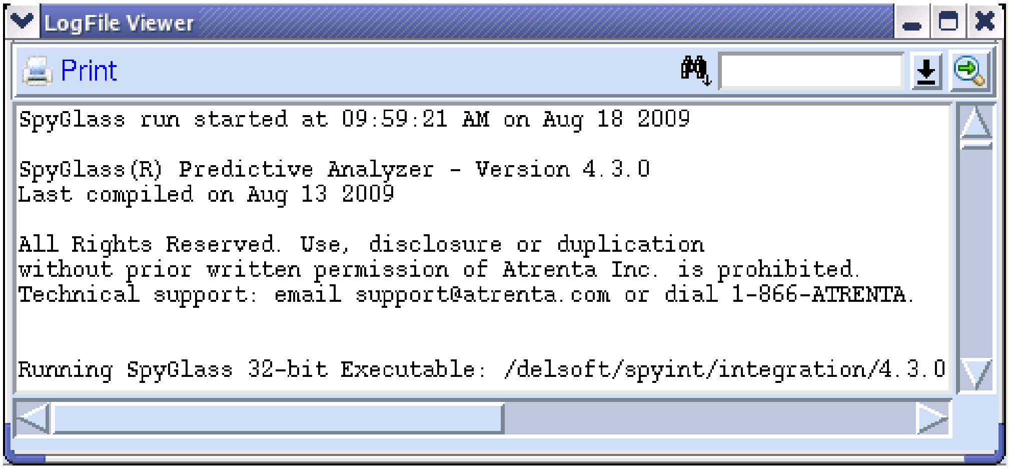
When you select this menu option, the Open window appears. Select the text file to be viewed. The Atrenta Console Text file viewer appears displaying the selected text file. If the Atrenta Console Text file viewer is already open displaying a text file selected earlier, the new text file replaces the earlier text file.

To find a string in the displayed text, use *Find*. The search can be case-sensitive or case-insensitive, both forward and backward in the file, and by regular expressions.

To print the displayed text file, use *Print*.

Tools > View Logfiles

This menu option is used to view to the log file created by a SpyGlass run. When you select this option, the *LogFile Viewer* dialog appears, as shown in the following figure:



```
SpyGlass run started at 09:59:21 AM on Aug 18 2009

SpyGlass(R) Predictive Analyzer - Version 4.3.0
Last compiled on Aug 13 2009

All Rights Reserved. Use, disclosure or duplication
without prior written permission of Atrenta Inc. is prohibited.
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.


Running SpyGlass 32-bit Executable: /delsoft/spyint/integration/4.3.0
```

FIGURE 10. LogFile Viewer

To find a string in the displayed text, click *Find*. The search can be case-sensitive or case-insensitive, both forward and backward in the file, and by regular expressions.

To print the displayed file, click *Print*.

To close the LogFile Viewer dialog click the *Close* button.

Tools > Aggregated Project Results

This menu option is used to view aggregate report that contains the combined results from multiple single-user projects.

For details on this report, refer to the *Generating Aggregate Reports* topic of *Atrenta Console User Guide*.

Tools > Datasheet Report

This menu option is used to view the DataSheet report. For details on this report, refer to the *Generating DataSheet Report* topic of *Atrenta Console User Guide*.

Tools > Dashboard Report

This menu option is used to view the Dashboard report. For details on this report, refer to the *Generating Dashboard Report* topic of *Atrenta Console User Guide*.

Tools >Methodology Configuration Window

This menu option is used to open the Methodology Configuration System window. For details, refer to the *Working with Methodologies* topic of *Atrenta Console User Guide*.

Tools > Preferences

This menu option is used to set various preferences in Atrenta Console:

When you select this option, the *Preferences* dialog appears. You can view the following pages in this dialog:

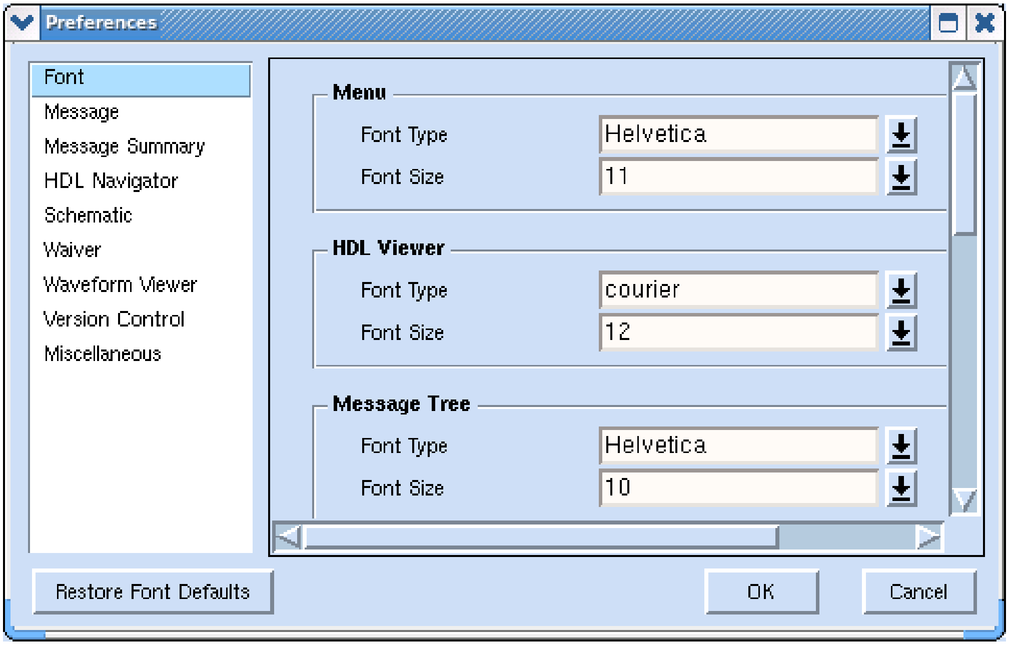
Font Page	*Message Page*
Message Summary Page	*HDL Navigator Page*
Schematic Page	*Waiver Page*
Waveform Viewer Page	*Version Control Page*
Miscellaneous Page	

Font Page

Use the Font page to set a font style and size of text appearing in *The Menu Bar*, *HDL Viewer Pane*, *Msg Tree Page*, *Msg Summary Page*, and *Message Help Section*.

In addition, you can set a font size of names appearing in *The Modular Schematic Window*/*The Incremental Schematic Window*. You can also set font size for different type of objects, such as net names, pin names, instance names, and attributes.

The following figure shows the Font page:



Menu

Font Type: Helvetica
Font Size: 11

HDL Viewer

Font Type: courier
Font Size: 12

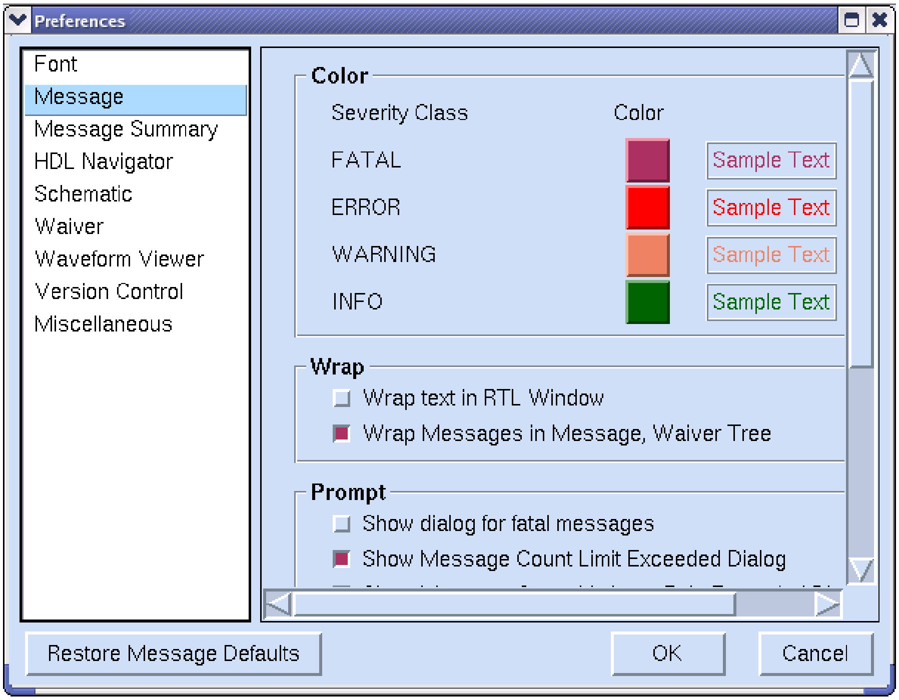
Message Tree

Font Type: Helvetica
Font Size: 10

FIGURE 11. The Atrenta Console Preferences Window - Font Page

Message Page

Use the Message page to set the color for each severity in the Msg Tree and the Msg Summary windows.



Color

Severity Class	Color	
FATAL	■	Sample Text
ERROR	🟥	Sample Text
WARNING	■	Sample Text
INFO	🟩	Sample Text

Wrap

- □ Wrap text in RTL Window
- ■ Wrap Messages in Message, Waiver Tree

Prompt

- □ Show dialog for fatal messages
- ■ Show Message Count Limit Exceeded Dialog

FIGURE 12. The Atrenta Console Preferences Window - Message Page

You can set the following options:

Severity Class

You can set the color for each severity class so that a message of the given severity class is highlighted in the selected color. To set the color, click the displayed color and the Choose color dialog appears:

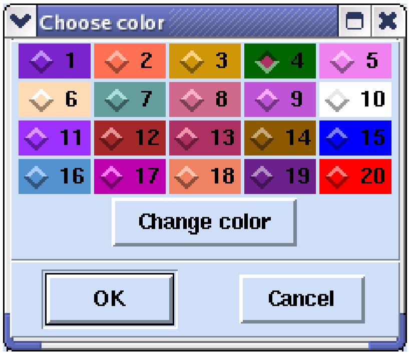


FIGURE 13. The Atrenta Console Preferences Window - Choose Color

Choose a highlighting color from the predefined color palette or click *Change Color* to select more colors for the palette. Click *OK* to apply the selected color.

The sample text is also displayed for each severity class.

Set the color scheme to your liking and click *OK* to apply the color scheme to the HDL Viewer.

Wrap text in RTL Window

Set this option to wrap the text displayed in the RTL tab of the HDL Viewer to fit the text in the current width of the HDL Viewer and hence avoid the use of horizontal scroll bar.

Wrap Messages in Message, Waiver Tree

Set this option to wrap the message text displayed in the Msg Tree page and Waived page of the Message window.

Show dialog for fatal messages

Set this option to enable the display of fatal messages in a pop-up window.

Show Message Count Limit Exceeded Dialog

Set this option for Atrenta Console to show a dialog when the total message count for a rule has exceeded the limit specified using the *Maximum Messages Per Rule* setting in the *Set Read Options* tab.

Show Message Count Limit per rule Exceeded Dialog

Set this option for Atrenta Console to show a dialog when the total message count for a rule has exceeded the limit specified using the *Maximum Messages Per Rule* setting in the *Set Read Options* tab.

Show Tool-tip.

Set this option to enable or display the information that is displayed as a tool-tip in the Msg Tree window, Msg Summary window and so on.

Show default waiver file name in message tree

Set this option to display the name of the default waiver file in the message tree.

Show Message ID

Set this option to enable or disable unique hexadecimal message IDs in the relevant sections of the Message window, such as the *Msg Tree Page* and *Msg Summary Page* (by default, this is enabled). These indexes (when enabled) allow you to cross-probe between the *Message Window* and other Atrenta Console windows.

Annotate Message ID in Reports

Set this option to enable or disable unique hexadecimal message ids in the standard SpyGlass reports (by default, this is enabled). These ids (when enabled) allow you to cross-probe between standard SpyGlass reports and other Atrenta Console windows.

Max No. of Messages

This option sets the maximum number of rule messages to be displayed in the Msg Tree page. By default, 100000 messages are displayed.

To view the balance messages, additional navigation options have been provided from the *View Pages* option in the shortcut menu of the Msg Tree page.

Max No. of Messages per rule

This option sets the maximum number of messages per rule to be displayed in the message tree. By default, 1000 messages are displayed.

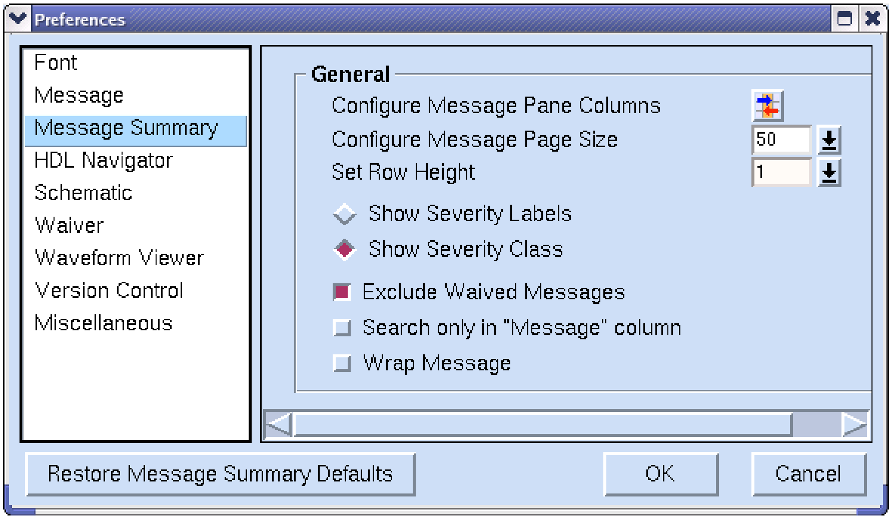
To view the balance messages, additional navigation options have been provided from the *View Pages* option in the shortcut menu of the message tree.

Enable Rule Tag Based Message Grouping

This option is used to group messages of the *Ac_sync_group* rules of SpyGlass CDC solution based on instance names or user-specified names.

Message Summary Page

Use the Message Summary page to set the appearance of the Msg Summary page.



Preferences

- Font
- Message
- Message Summary
- HDL Navigator
- Schematic
- Waiver
- Waveform Viewer
- Version Control
- Miscellaneous

General

Configure Message Pane Columns

Configure Message Page Size: 50

Set Row Height: 1

◇ Show Severity Labels

◆ Show Severity Class

☒ Exclude Waived Messages

□ Search only in "Message" column

□ Wrap Message

FIGURE 14. The Atrenta Console Preferences Window - Message Summary Page

You can set the following preferences for the Message Summary window:

Configure Message Pane Columns

Use this option to add/remove/rearrange the displayed columns in the Message List window. When you select this option, the Configure Columns dialog appears, as shown below:

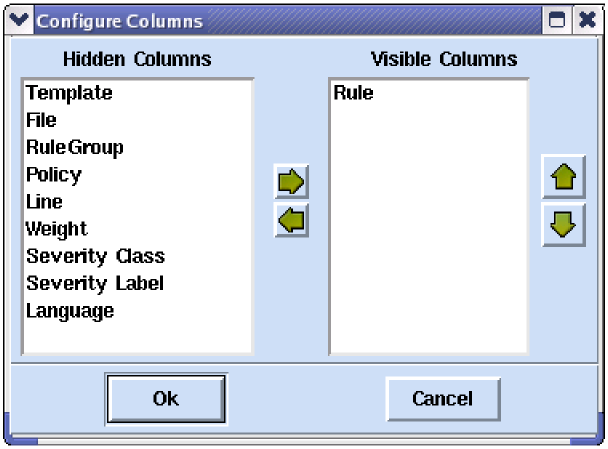


FIGURE 15. The Atrenta Console Preferences Window - Configure Columns

You can select the columns to be displayed or hidden by moving the column names between the Visible Columns section and the Hidden Columns section by using the corresponding move arrow buttons.

You can also rearrange the column order in the Visible Columns section by using the corresponding move-up or move-down arrow.

Configure Message Page Size

Use this drop-down list to increase/decrease the number of violations displayed per page of the Message List window.

Set Row Height

Use this drop-down list to increase/decrease the height of the rows in the Message List window. This option is useful to view the complete message that has been wrapped.

Show Severity Labels

Use this option to display the number of violations displayed by severity labels in the Summary Matrix window.

Show Severity Class

Use this option to display the number of violations displayed by severity class in the Summary Matrix window.

`Exclude Waived Messages`

Use this option to show or hide the waived messages in the Msg Summary page.

`Search only in “Message” Column`

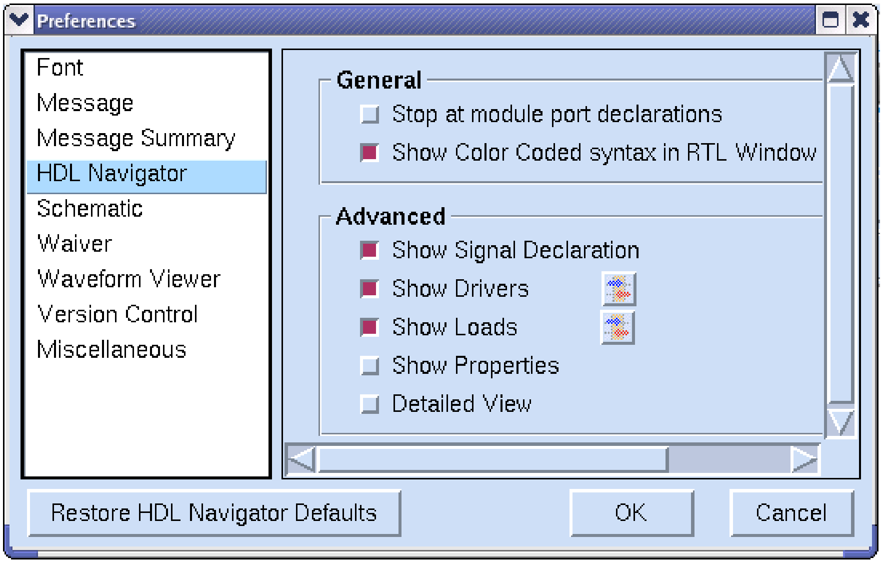
Use this option to limit the search to the Message column only.

`Wrap Message`

Set this option to wrap the message text displayed in the Message List window of the Msg Summary page.

HDL Navigator Page

Use the HDL Navigator page to show or hide the color of the RTL source code. In addition, use the HDL Navigator page to view detailed information about the drivers and loads declared for a signal.



General

- ☐ Stop at module port declarations
- ☑ Show Color Coded syntax in RTL Window

Advanced

- ☑ Show Signal Declaration
- ☑ Show Drivers
- ☑ Show Loads
- ☐ Show Properties
- ☐ Detailed View

FIGURE 16. The Atrenta Console Preferences Window - HDL Navigator Page

You can set the following HDL Navigator page preferences:

`Stop at module port declarations`

Use this option to restrict the Loads and Drivers results such that while traversing the input/output cone, if module boundary is found before any other instance, the declaration is returned as the load/driver. When this option is not selected, even if module boundary is found, the traversing continues until an instance or a primary port (outside the module) is found.

`Show Color Coded syntax in RTL window`

Use this option to view the color-coding of the RTL source in *HDL Viewer Pane*.

`Show Signal Declaration`

Use this option to view the location of the signal declared in the HDL Viewer.

`Show Drivers`

Use this option to show or hide the columns of the table that contains information about drivers declared for a signal. When you select the *Detailed View* option, the (⇄) button is enabled. Clicking the (⇄) button displays the Configure Columns dialog.

To hide a column in this dialog, select the name of the column from the *Visible Columns* list and click the (⇦) button.

To show a column, select the column from the *Hidden Columns* list and click the (⇨) button. You can also change the order in which the columns appear in the table by using the (⇧) and (⇩) buttons.

For example, in the above figure, if you want to display the *HDL column* before the *File column*, select the *HDL column* and click the (⇧) button.

`Show Loads`

Use this option to show or hide the columns of the table that contains information about loads declared for a signal. When you select the Detailed View option, the (⇄) button is enabled. Clicking the (⇄) button displays the Configure Columns dialog. You can configure the columns as explained for the *Show Drivers* section.

Show Properties

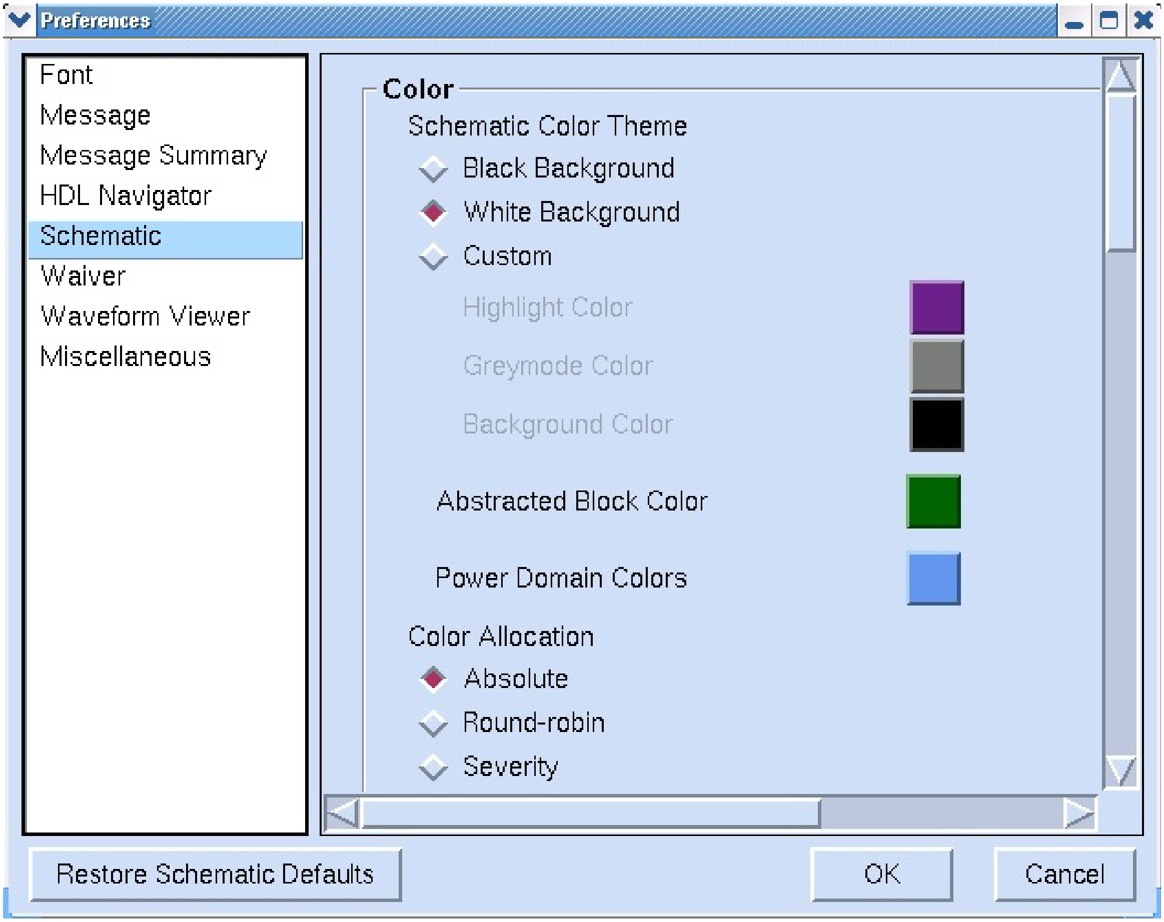
Use this option to show or hide the Properties field in the HDL Navigator window

Detailed View

Use this option to view detailed information, about the drivers and loads that are declared for a signal in the HDL Viewer. When you select this option, the information about the drivers and loads appears in the *HDL Navigator Pane* in a tabular format with columns containing information such as the file name, line number, and the HDL code.

Schematic Page

Use the Schematic page to set the appearance of the schematic windows.



Color

Schematic Color Theme

○ Black Background

● White Background

○ Custom

Highlight Color 🟪

Greymode Color ▨

Background Color ■

Abstracted Block Color 🟩

Power Domain Colors 🟦

Color Allocation

● Absolute

○ Round-robin

○ Severity

FIGURE 17. The Atrenta Console Preferences Window - Schematic Page

You can set the following schematic preferences:

Highlight Color

Choose the highlighting color of the selected object in the schematic windows by clicking the displayed color and by selecting a different color from the color palette.

Greymode Color

Choose the color of the schematic display (except the message path) when a message is cross-probed to the schematic windows (from the Message tree view in Msg Tree page of the Message window) by clicking the displayed color and selecting a different color from the color palette.

Background Color

Choose the background color of the schematic windows by clicking the displayed color and selecting a different color from the color palette.

Show names at net corners

Displays net names at net corners.

Show names at window borders

Displays net names at a window boundary if a net does not fit in the current display.

Power Domain Colors

You can change the default colors used for Power and Voltage domains displayed in the Schematic by clicking on a color on the right. This opens the color palette, which allows you to change individual colors similar to the highlighted color palette.

Atrenta Console now provides a new set of palette with lighter colors added to reduce color interference and better visibility. If you want to use the old palette and overwrite the new color set, added the following setting in the .spyglass.setup file:

```
SDE_CONFIG_OPTIONS=initial_preference:spy_AllPDColors=#5f7d9
eb8a082 #cdd268728978 #ba9f553fd3b5 #ffffffffffff
#9ba53021ffff #a5a12a3d2a3d #b0b030306060 #8b855a5e0000
#00000000ffff #4f5c947acdd2 #bdf30000b0a3 #eeee82826262
#686822228b8b #8faabcab8f93 #645c9592edd2 #bdb1b76f6b5d
#f0e4f8edffff #7f7cffffd4d1 #1e3590e8ffff #ffffd8920000
```

You can also specify the colors of your choice using the above command. However, the list specified should have 20 color items.

Color Allocation

Select *Absolute* to use the severity class colors as the probe or message colors, select *Round-robin* to automatically assign probe or message colors from the color palette on round-robin basis, or select *Severity* to assign colors by message severity.

In the Absolute mode, the probe or message colors in the schematic windows can be the same as the severity class colors (set in the *Message Page* of the *Tools > Preferences*... menu option). The probe or message colors in the HDL Viewer will be the same as the severity class colors (set in the *Message Page* of the *Tools > Preferences...* menu option).

In Round-robin mode, the probe or message colors in HDL Viewer and schematic windows are always same.

Show Tool-tip On Schematic Objects

Set this option to view a tool-tip whenever the cursor is placed over an object in the schematic windows. The tool-tip displays the full name of the object (net, pin, port, pin-bus, or instance). By default, this option is set and the tool-tip is displayed whenever the cursor is placed over an object in the schematic windows.

Show Instance Names

Set this option to turn on/off the display of instance names in the schematic windows.

NOTE: *This option is useful because in a netlist design, the debugging of a violation message using the schematics becomes difficult if the instance names are too long. In such a scenario, part of the instance name of the first gate is overwritten by the instance name of the next gate and thus the information becomes unreadable.*

Show Module Names

Set this option to turn on/off the display of module names in the schematic windows.

Show Atrenta Primitive Names

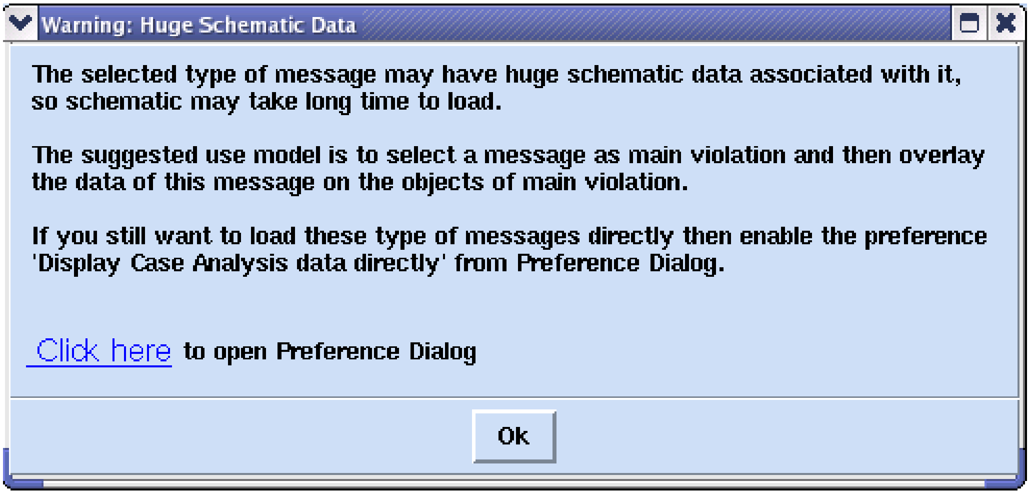
Set this option to turn on/off the display of module names of Atrenta generic primitives (such as RTL_FD, RTL_AND, RTL_BUFF, etc.) in the schematic windows. By default, this option is set and the module names of the Atrenta primitives are not displayed in the schematic windows. If this option is modified while the schematic windows are open, the schematic windows must be reloaded to view the modified display.

Display Case Analysis Data Directly

Set this option to display data for informational rules, such as *Info_testmode*, *Info_Case_Analysis*, and *Show_Case_Analysis*.

Such informational rules generate a lot of data in the schematic, which results in large loading time and high memory usage. Therefore, by default, this option is turned off and the data for such informational rules is not loaded in the schematic.

If this option is turned off and you try to load the violation of such rules, following dialog is displayed if the data dumped for the violation is huge:



Warning: Huge Schematic Data

The selected type of message may have huge schematic data associated with it, so schematic may take long time to load.

The suggested use model is to select a message as main violation and then overlay the data of this message on the objects of main violation.

If you still want to load these type of messages directly then enable the preference 'Display Case Analysis data directly' from Preference Dialog.

Click here to open Preference Dialog

FIGURE 18. The Atrenta Console Preferences Window - Huge Schematic Data Warning

Automatically Load Case Analysis Data

Set this option to load static data, if present, associated with selected violation, automatically every time violation is selected. When this option is disabled, associated data is not loaded and you need to search for the violation to debug in the message tree.

Expand all signals of the Pinbus on Double Clicking

Set this option to have the schematic expanded to show all connected nets and the objects connected to these connected nets when a pin bus is double-clicked.

Show List/Trace Connected Nets Dialog on Double Clicking

Set this option to show the *List/Trace Connected Nets* dialog whenever a pin bus/net bundle/port bus is double-clicked.

NOTE: *This option is useful because in a netlist design, the debugging of a violation message using the schematics becomes difficult if the instance names are too long. In such a scenario, part of the instance name of the first gate is overwritten by the instance name of the next gate and thus the information becomes unreadable.*

Maintain Relative Hierarchies for Instances in IS

This option is used to add the parent hierarchies of the instances that are loaded/appended or cross-probed into Incremental Schematic from any other window (such as, from RTL, Instance Browser, Power Browser, Design Tree etc.). By default, this option is set to on.

Show Full Cone

Set this option to display all possible paths in the schematic windows while extracting the fan-in/fan-out cone. By default, not all paths are displayed.

Show net Bundles in IS

Set this option to enable the display of net bundles in the Incremental Schematic (IS). The net bundle takes the attributes of the constituent nets. For example, if a net contained in the net bundle is partial (dashed) or highlighted (colored), the net bundle would also appear the same.

Disable Auto Filtering in Schematic Find

Set this option to disable the auto-filtering feature in the *Find in Modular Schematic* dialog.

Show Full Connectivity on double click

Set this option to display complete connectivity on double-clicking on an input pin/port, output pin/port, and partial net.

For details, refer to the *Double-clicking on an input or output pin/port and partial* *net*.

Enable Single-click Probe

Set this option to enable single-click probe in the schematic.

Maximum Nets to Probe

Set this option to probe the specified number of net bits in a net bundle. The default value for this option is 64, that is, SpyGlass will probe only 64 or lesser number of bits in a net bundle.

Skip text annotation for instances with pins more than

Set this option to disable showing the text attribute on instances with pins more than set value. The default value for this option is 1024.

Hierarchical view enabled with instance count greater than

Set this option to switch to the hierarchical view when the number of instances reported reach the specified value. The default value of this option is 100.

NOTE: *This option is enabled only for the SpyGlass DFT product. Please refer to the DFT Rules Reference Guide for more information on the rules that support hierarchical view.*

Highlight Width

Choose the highlighting width of the selected object in the schematic windows from 1(thinnest) to 9 (thickest). The default width is 3.

Probing Mode

Use this option to set the probing modes. You can set the probing modes to the following:

- *Normal (Single Color)* - When you select this option, the selected probes or messages are highlighted with one color only (the highlight color set using the *Highlight Color* option).

  All un-probed instances are shown in blue color and all un-probed nets are shown in yellow color in both schematic windows.
- *Grey (Single Color)* - When you select this option, the selected probes or messages are highlighted with one color only (the highlight color set using the *Highlight Color* option).

All un-probed instances and nets are shown with grey mode color (set using the *Greymode Color* option).

- *Grey (Multiple Colors)* - When you select this option, the selected probes or messages are highlighted with different colors automatically selected from the color palette.

  All un-probed instances and nets are shown with grey mode color (set using the *Greymode Color* option).

Auto Expand

Use this option to auto-expand all groups and their sub-groups in the current module. The other modules remain unaffected.

Auto collapse any expanded groups before selecting a new violation

Use this option to auto-collapse all expanded groups and their sub-groups in the current module before a new violation message is selected. The other modules remain unaffected.

Allow sequential logic in IS clouds, if allowed by rule

Use this option to show the sequential logic grouped under clouds in the Incremental Schematic, if the rule allows. When this option is not set, the sequential logic will not be displayed as grouped under clouds. By default, this option is enabled.

Show Registers outside RTL groups

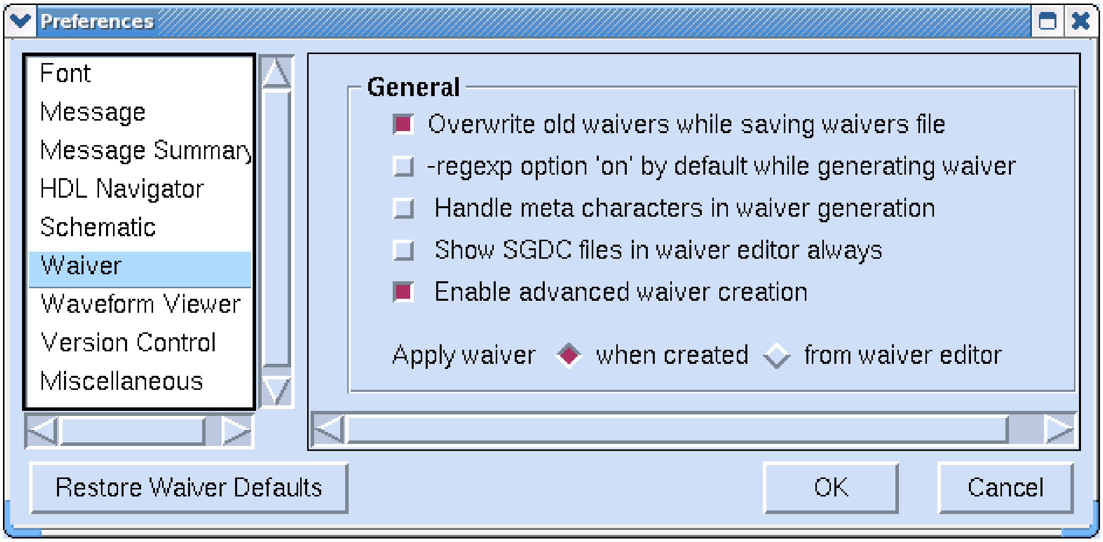
Use this option to view the registers outside the grouped RTL blocks.

When the preferences for schematic windows are set, click *OK*.

Waiver Page

Use the *Waiver* page to set waiver options for violation messages.

The following figure shows the *Waiver* page:



General

☑ Overwrite old waivers while saving waivers file

□ -regexp option 'on' by default while generating waiver

□ Handle meta characters in waiver generation

□ Show SGDC files in waiver editor always

☑ Enable advanced waiver creation

Apply waiver ● when created ○ from waiver editor

FIGURE 19. The Atrenta Console Preferences Window - Waiver Page

You can set the following waiver options:

`Overwrite old waivers while saving waivers file`

Set this option to overwrite old waivers with the new ones while saving the waivers file. If this option is not set, old waivers are commented out and new waivers are appended to the waivers file while saving the file.

`-regexp option ‘on’ by default while generating waiver`

Set this option to keep the regular expression support on by default while creating waivers.

`Handle meta-characters in waiver generation`

Set this option to handle meta-characters (such as `*`) while generating waivers in GUI.

This option is used to handle meta-characters when waivers are generated by selecting the *Waive Selected Messages* option from the right-click menu of a violation message in the *Msg Tree*.

`Show SGDC files in waiver editor always`

Shows the *SGDC Files* node in the *Tree-View Section*. Under this node, SGDC files containing waiver commands are displayed.

`Enable advanced waiver creation`

Set this option to apply waivers based on different criteria, such as waive by file, module, rule, severity, or selected message.

After setting this option, when you right-click on a message in the *Msg Tree*, the *Waive* option appears. From the sub-menu of the *Waive* option, you can select the required option to waive messages based on a criterion.

`Apply Waiver`

Set any of the following options from this category:

- *When Created*

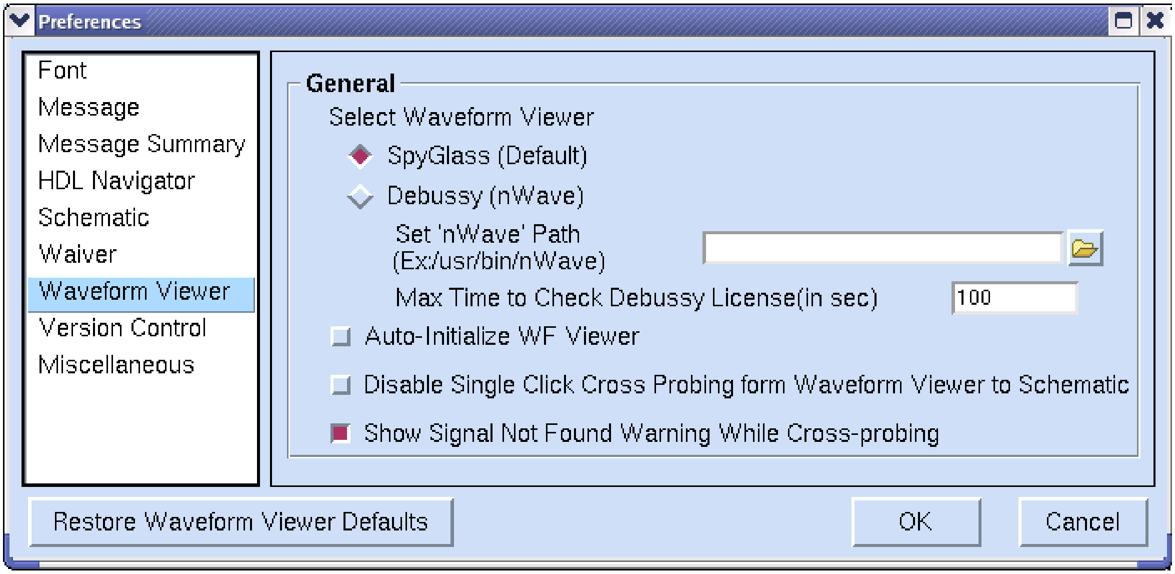
  Set this option if you want to apply waiver on the selected messages directly from the *Results* pane.

- From Waiver Editor

  Set this option if you want to apply waiver on the selected messages through *Waiver Editor Window*.

Waveform Viewer Page

Use the Waveform Viewer Page to set the waveform viewer to view the waveform associated with a message.



Preferences

- Font
- Message
- Message Summary
- HDL Navigator
- Schematic
- Waiver
- Waveform Viewer
- Version Control
- Miscellaneous

General

Select Waveform Viewer

● SpyGlass (Default)

○ Debussy (nWave)

Set 'nWave' Path (Ex:/usr/bin/nWave)

Max Time to Check Debussy License(in sec) 100

□ Auto-Initialize WF Viewer

□ Disable Single Click Cross Probing form Waveform Viewer to Schematic

☒ Show Signal Not Found Warning While Cross-probing

FIGURE 20. The Atrenta Console Preferences Window - Waveform Viewer Page

You can set the following waveform options:

Select Waveform Viewer

You can select either *SpyGlass (default)* viewer or the *Debussy (nWave)* waveform viewer. If you select Debussy nWave waveform viewer, you need to set the path of nWave in the corresponding field. Click the *Browse* button to browse to the directory location of nWave executable and select the required filename.

NOTE: *You need to have a license to run the Debussy nWave waveform viewer. Atrenta does not provide any license to run the Debussy waveform viewer.*

Disable Single Click Cross Probing from Waveform Viewer to Schematic

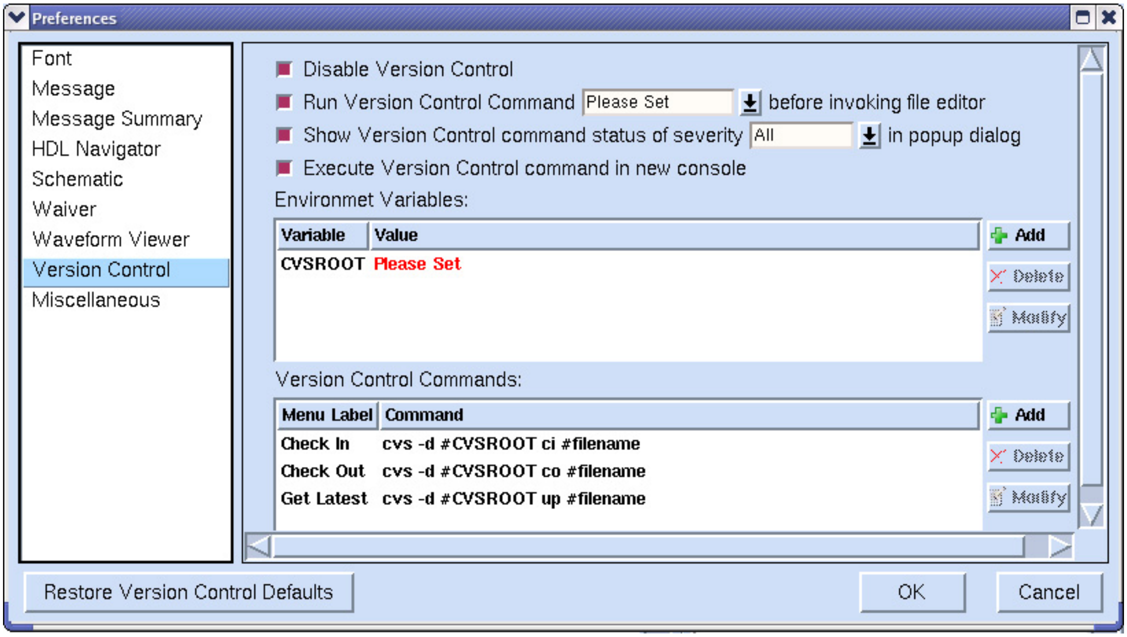
Set this option to disable cross-probing from the *Waveform Viewer* window to a schematic window.

Version Control Page

Use the Version Control page to configure a version control tool in SpyGlass. Configuring a version control tool ensures that you are working on the latest HDL file. In addition, a version control tool ensures that the changes in the file are not lost while modifying a design.

NOTE: *By default, SpyGlass uses the CVS version control tool. However, you can integrate your own version control tool with SpyGlass.*

Select the *Enable Version Control* check box from the Version Control page to display the version control options as shown below:



■ Disable Version Control

■ Run Version Control Command Please Set before invoking file editor

■ Show Version Control command status of severity All in popup dialog

■ Execute Version Control command in new console

Environmet Variables:

Variable	Value
CVSROOT	Please Set

Version Control Commands:

Menu Label	Command
Check In	cvs -d #CVSROOT ci #filename
Check Out	cvs -d #CVSROOT co #filename
Get Latest	cvs -d #CVSROOT up #filename

FIGURE 21. The Atrenta Console Preferences Window - Version Control Page

You can set the following options in the Version Control page:

Disable Version Control

Use this option to disable the version control tool.

Run Version Control Command before invoking the file editor

Use this option to specify the version control command that should be executed before editing a file. Select a command that you want to execute from the drop-down list, before invoking the file editor.

NOTE: *Only the commands that have been specified in the Version Control Commands section are displayed in the drop-down list.*

Show Version Control command status of severity in popup dialog

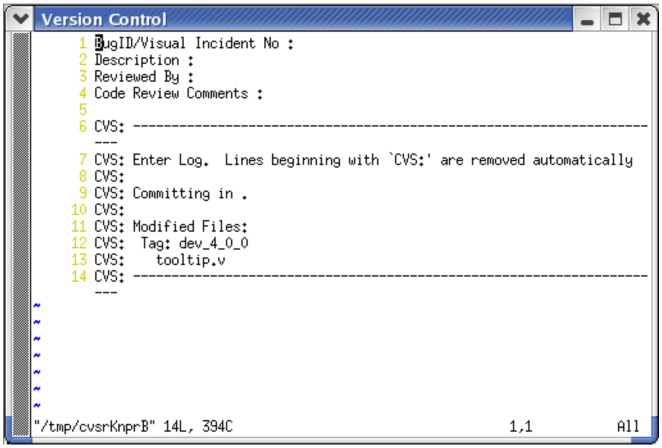
Use this option to view the success result of the version control command that you have executed for a file defined on the *Adding the Design File* section in the *Atrenta Console User Guide*.

You can select from the following available options:

- Error: Selecting this option displays an error in a pop dialog, if a version control command is not executed successfully.
- All: Selecting this option displays all results related to the execution of the version control commands in a popup dialog.
- Not Set: Selecting this option does not display the popup dialog.

Execute Version Control Command in new console

Use this option to execute the version control command on a new Atrenta Console window as shown below:



```
BugID/Visual Incident No :
Description :
Reviewed By :
Code Review Comments :

CVS: ----------------------------------------------------------------------
---
CVS: Enter Log.  Lines beginning with `CVS:' are removed automatically
CVS:
CVS: Committing in .
CVS:
CVS: Modified Files:
CVS:  Tag: dev_4_0_0
CVS:    tooltip.v
CVS: ----------------------------------------------------------------------
---
~
~
~
~
~
~
~
"/tmp/cvsrKnprB" 14L, 394C                                  1,1           All
```

FIGURE 22. Version Control

Environment Variables

Use this section to add, modify, or delete the system configuration variables related to the version control tool that you are using.

To add a variable, click (✚ Add). The Add a Variable dialog appears as shown below:

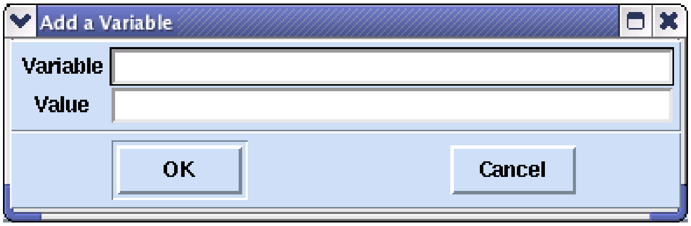


FIGURE 23. Adding an Environment Variable

Enter the variable name in the *Variable* field and the value of the variable in the *Value* field. Click *OK* to add the variable.

You can also modify an already specified variable. To do so, select the variable that you want to modify in the Environment Variables section and click the *Modify* button. Modify the variable in the Modify System Variable dialog that appears and click *OK* to save the new variable setting.

To delete a variable, select the variable and click the *Delete* button.

Version Control Commands

Use this section to add, modify, or delete the commands related to the version control tool that you are using.

To add a command, click ✚ Add . The Add a Menu Label dialog appears as shown below:

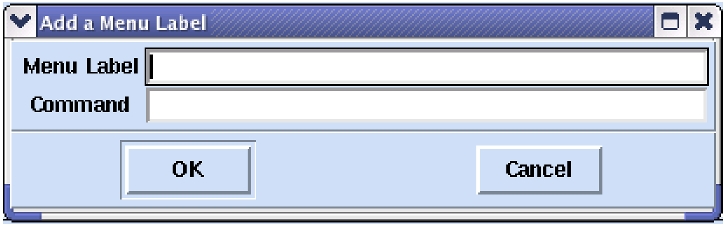


FIGURE 24. Adding a Menu Label

Add a label for the command name in the *Menu Label* text field and the version control command in the *Command* field. Click *OK* to add the command.

You can also modify an already specified command. To do so, select the command that you want to modify in the Version Control Commands section and click the *Modify* button. Modify the command in the Modify System Variable dialog that appears and click *OK* to save the new command setting.

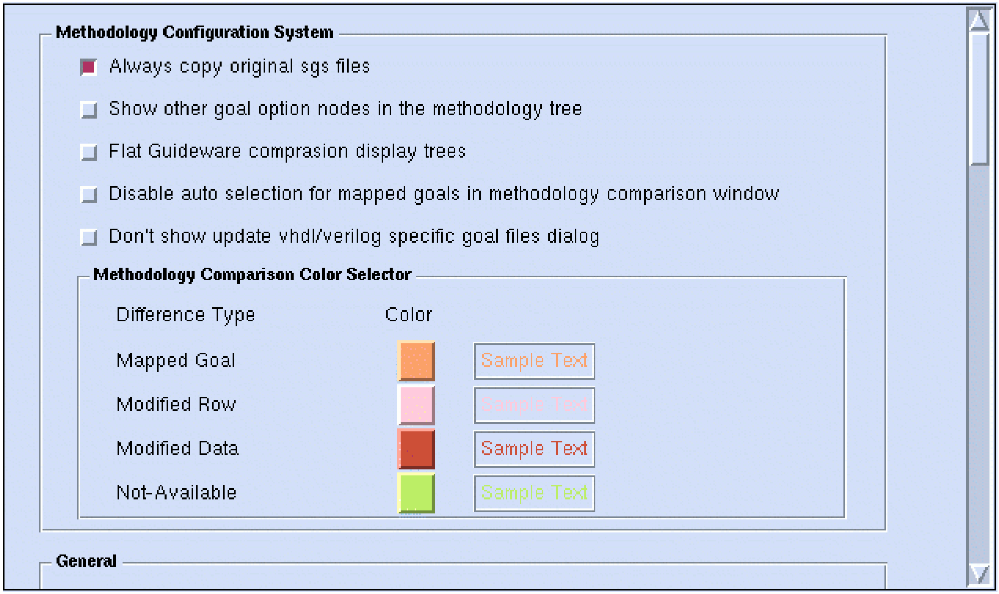
To delete a command, select the command and click the *Delete* button.

Click the *Restore Defaul*t button restore the default values of CVS, which is the default version control tool used by SpyGlass.

When the selection is to your liking, click *OK*.

Miscellaneous Page

Use the *Miscellaneous* page to set miscellaneous preferences.



Methodology Configuration System

☑ Always copy original sgs files

□ Show other goal option nodes in the methodology tree

□ Flat Guideware comprasion display trees

□ Disable auto selection for mapped goals in methodology comparison window

□ Don't show update vhdl/verilog specific goal files dialog

Methodology Comparison Color Selector

Difference Type	Color	
Mapped Goal	🟧	Sample Text
Modified Row	🟪	Sample Text
Modified Data	🟥	Sample Text
Not-Available	🟩	Sample Text

General

FIGURE 25. The Miscellaneous Page

You can set the following miscellaneous preferences:

Methodology Comparison Color Selector

Use this option to specify a color coding scheme for the methodology or goals comparison results.

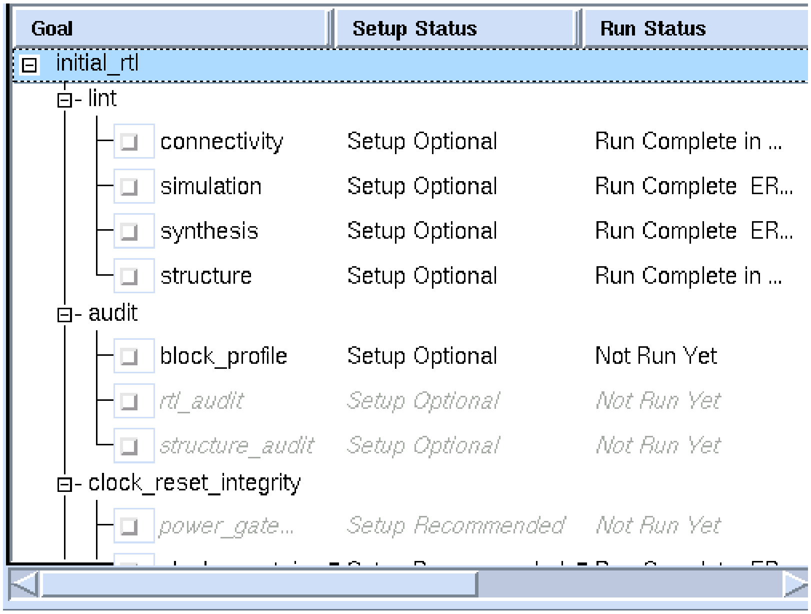
Enforce Execution of Prerequisite Goals

Use this option to enforce the execution of prerequisite goals so that the prerequisite goals are run first before the selected goals.

After enforcing the execution of prerequisite goals, if you try running a goal without first running its prerequisite goals, Atrenta Console displays a *Warning* dialog. This dialog lists the prerequisite goals that have not yet been run or selected for the current run. You can select all the prerequisite goals by selecting the *Select all required prerequisite goal(s)* option in this dialog. However, if you do not want to run the prerequisite goals, select the *Do not enforce the execution of prerequisite goals* option in this dialog.

Show Optional Goals

Set this option to display optional goals under the *Select Goal* tab. All the optional goals appear in italics, as shown in the following figure:



Goal	Setup Status	Run Status
⊟ initial_rtl		
⊟- lint		
□ connectivity	Setup Optional	Run Complete in ...
□ simulation	Setup Optional	Run Complete ER...
□ synthesis	Setup Optional	Run Complete ER...
□ structure	Setup Optional	Run Complete in ...
⊟- audit		
□ block_profile	Setup Optional	Not Run Yet
□ *rtl_audit*	*Setup Optional*	*Not Run Yet*
□ *structure_audit*	*Setup Optional*	*Not Run Yet*
⊟- clock_reset_integrity		
□ *power_gate...*	*Setup Recommended*	*Not Run Yet*

FIGURE 26. Show Optional Goals

You can select the optional goals like any other goal in the above page.

Enable Scenario Support

Use this option to enable a user to work with scenarios, such as creating, modifying, and deleting scenarios.

For details on scenarios, refer to *Working With Scenarios* topic in *Atrenta Console User Guide*.

Display Precompiled File(s) associated with HDL Library

Use this option to view precompiled files associated with an HDL library appearing in the *HDL Libraries* section under the *Design Setup* tab.

The following figure shows precompiled files appearing in the *HDL Libraries* section:

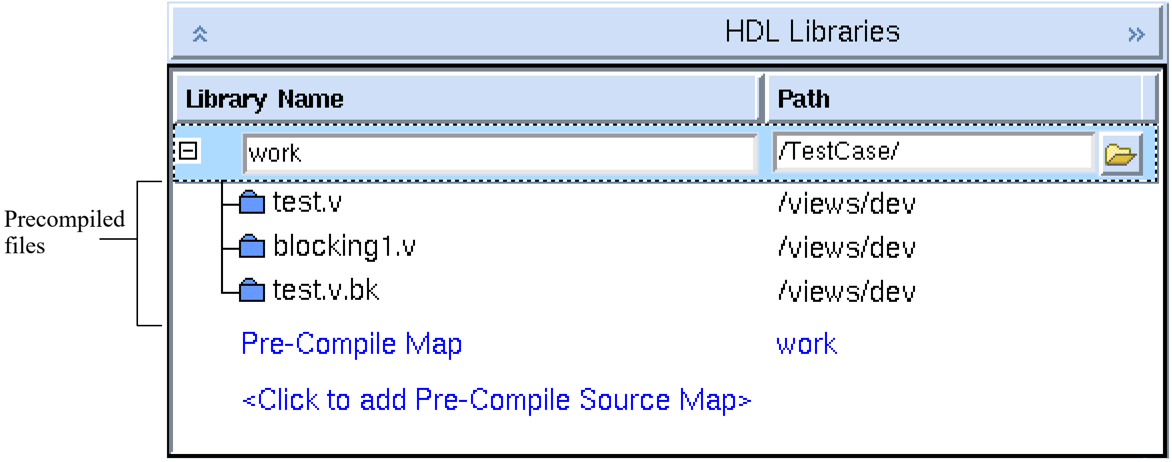


FIGURE 27. Display Precompiled Files

`Always copy original sgs files`

Use this option to copy the setup guide script (sgs file), if any, of a goal(s.

`Enable Save-restore Flow`

Use this option to enable the design save-restore feature. The save-restore feature enables you to analyze a design without synthesizing the same (unaltered) design for each Atrenta Console run. Therefore, selecting this feature significantly improves the runtime on repeated Atrenta Console runs when the HDL source is unchanged.

`Enable Incremental Mode for All Goals`

Use this option to compare the results of a previously run goal with the current goal.

If the incremental mode is turned on and the goal is already run, the .vdb file generated from a previous run is used as the reference VDB. If the reference VDB is corrupt then the incremental mode is automatically turned off.

`Show Guidance Bar Frame`

Set this option to turn on/off the display of the guidance bar.

Show Startup Screen

Use this option to enable the display of the startup screen every time Atrenta Console is invoked.

Show warning if precompiled libraries compatible with current platform do not exist.

Set this option to display a *Warning* dialog informing the user that a project file created on a 32-bit or 64-bit machine is being opened on a 64-bit or 32-bit machine, respectively.

Show Modules in Alphabetic Order in Module Tree

Use this option to alphabetically sort modules in the *Module View Page*.

Specify host_config_file

Specify a host configuration file in this field.

If a host configuration file is already specified by the `HOST_CONFIG_FILE` key in .spyglass.setup, the details of that file appear under this field so that you can edit the details as per your requirement.

This file is used for parallel execution of goals on different machines.

Allow multiple windows to be opened for displaying spreadsheets

Use this option to enable opening of multiple spreadsheets at a time.

Use Master Slave (1 Master + 1 Slave) configuration for spreadsheet

Use this option to enable opening of a master spreadsheet and one of its corresponding slave spreadsheet at a time.

In this case, the master spreadsheet remains opened and the slave spreadsheet keeps getting refreshed with a different slave spreadsheet based on user’s action.

The master spreadsheet closes only when you open another master spreadsheet.

Don't display icon on left bar

Use this option to hide icons, such as the schematic icon and the waveform viewer icon appearing on the left-side of the spreadsheet.

Specify Pdf file reader

This option sets the path to the PDF file viewer that is required for viewing the printable SpyGlass documentation files.

This field is controlled as follows:

- The value of the `SG_PDF_VIEWER` environment variable, if set, has the highest priority. Every time, you start Atrenta Console, this field shows the value of the `SG_PDF_VIEWER` environment variable, if set and uses that PDF viewer.
- You can change the PDF viewer for the current session by typing the complete path and file name of the PDF viewer executable or by clicking *Browse* and then search and select the executable file. If the `SG_PDF_VIEWER` environment variable has not been set, this setting will persist from session to session.
- If the `SG_PDF_VIEWER` environment variable has not been set and you do not set the *Specify pdf file reader* preference, Atrenta Console will try to invoke the commonly-used PDF viewer of the particular platform.

Editor Environment variable

This field shows the value of the `EDITOR` environment variable as used by the *Specify external editor program* setting.

Specify external editor program

This field sets the path to the text editor for viewing and editing text files.

Type the complete path and file name of the executable text editor or browse to the executable file.

If you do not set this preference, Atrenta Console invokes the text editor specified in the `EDITOR` environment variable or the commonly-used text editor of the particular platform.

Specify external editor line flag

This field sets the line flag (line number format) specific to an editor, which is specified in the *Specify external editor program* field.

Line flag is a way of specifying the line number from which an editor should open.

Specify the format by using `%s`, which gets replaced with the line number specified by the user while opening that editor.

The following table shows examples of few editors and their respective line flags:

Editor	Line Flag	Example
gvim	+%s	To display the test.v file from line 400 in `gvim`, the following command is used: `gvim +400 test.v`
nedit	-line %s	To display the test.v file from line 400 in `nedit`, the following command is used: `nedit -line 400 test.v`

Use external editor for displaying reports/logs

Select this option to open reports and log files in the text editor specified by the *Specify external editor program* field.

Specify HTML Browser program

This field sets the path to the HTML File Browser for viewing the SpyGlass HTML-based Help files.

This field is controlled as follows:

- The value of the `SG_HTML_BROWSER` environment variable, if set, has the highest priority. Every time, you start Atrenta Console, this field shows the value of the `SG_HTML_BROWSER` environment variable, if set and uses that HTML File Browser.
- You can change the HTML File Browser for the current session by typing the complete path and file name of the HTML File Browser executable or by clicking *Browse* and then search and select the executable file. If the `SG_HTML_BROWSER` environment variable has not been set, this setting will persist from session to session.
- If the `SG_HTML_BROWSER` environment variable has not been set and you also do not set this preference, Atrenta Console will try to invoke the commonly-used HTML File Browser of the particular platform.

Create cdc_false_path from:

Use this option to set the `cdc_false_path` constraint from a source flip-flop to a destination flip-flop (*Source FF, Dest FF* option) or from a source clock to a destination clock (*Source Clk, Dest Clk* option).

`Do not Show cdc_false_path setup dialog again`

Use this option to show or hide the Select Clock/Flop dialog when creating the `cdc_false_path` constraints for the Ac_unsync01 rule.

`SGDC to Schematic Cross Probing`

Use this option to specify the maximum number of objects that can be cross-probed from an SGDC file.

You can also select the *Do not show dialog to ask count of objects that can be cross-probed from SGDC file* option to hide the display of the dialog that appears at the time of cross-probing.

`Do not Show Aggregated Project Results dialog again`

Use this option to not display the *Aggregated Project Results* dialog again.

`Automatically launch Report browser when goal results are loaded`

Use this option to display Power Report Browser containing power reduction results.

For details on this browser, refer to *SpyGlass Power Family Rules Reference Guide*.

Tools >Power Intent View

This menu option is used to view the view the power intent of the design. For details of this option, refer to the The **Power Intent View Window section** of the *SpyGlass Power Verify Rules Reference Guide*.

NOTE: *By default, this option is hidden. To enable this option, enable UPF and run a goal from the SpyGlass Power Verify solution.*

Help Menu

When you click the *Help* menu, the following options appear:

Help > SpyGlass Help	*Help > SpyGlass Manuals*
Help > Search SpyGlass Manuals	*Help > Icons Quick Reference*
Help > Shortcut Keys	*Help > Atrenta Console User Guide*
Help > Atrenta Console Reference Guide	*Help > BuiltIn Rules Reference*
Help > Methodology Guides	*Help > SpyGlass Release Summary*
Help > SpyGlass Release Notes	*Help > SpyGlass KPNS*
Help > Online Support	*Help > About SpyGlass*

Help > SpyGlass Help

This menu option invokes the `spyhelpviewer` utility to display SpyGlass documentation in an HTML format.

By default, the `spyhelpviewer` utility searches for the `netscape` executable in your machine's path for displaying the PDF files. Use the *Tools > Preferences* > *Miscellaneous Page* > *Specify HTML Browser program* option or the new `SG_HTML_BROWSER` environment variable to set your HTML Browser.

Help > SpyGlass Manuals

This menu option invokes the `spydocviewer` utility that displays SpyGlass printable (PDF) documentation in a tree format for easy access. In addition, the SpyGlass Standard Rule-Primitive documentation (in text format) is also accessible.

By default, the `spydocviewer` utility searches for the `acroread` or `xpdf` executable in your machine's path for displaying the PDF files. Use the *Tools > Preferences* > *Miscellaneous Page* > *Specify Pdf file reader* option or the new `SG_PDF_VIEWER` environment variable to set your PDF viewer.

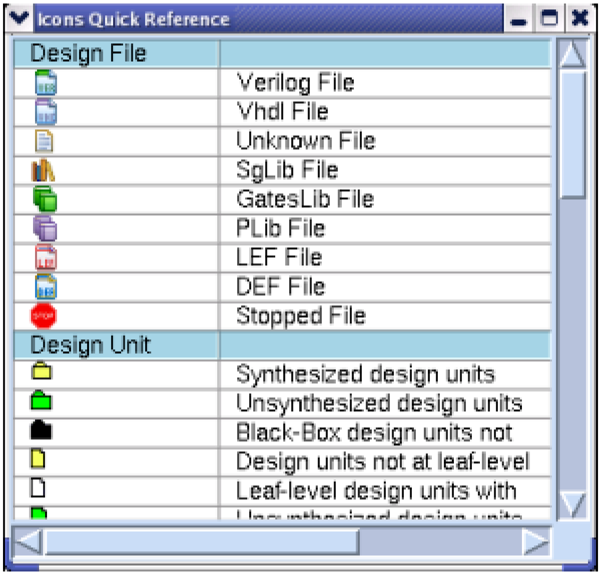
You can also use the F1 key on your keyboard to invoke the on-line manuals.

Help > Search SpyGlass Manuals

This menu option displays the *Search* dialog using which you can search for specific information in various SpyGlass documents.

Help > Icons Quick Reference

This menu option invokes the *Icons Quick Reference* window that displays various icons used in Atrenta Console.

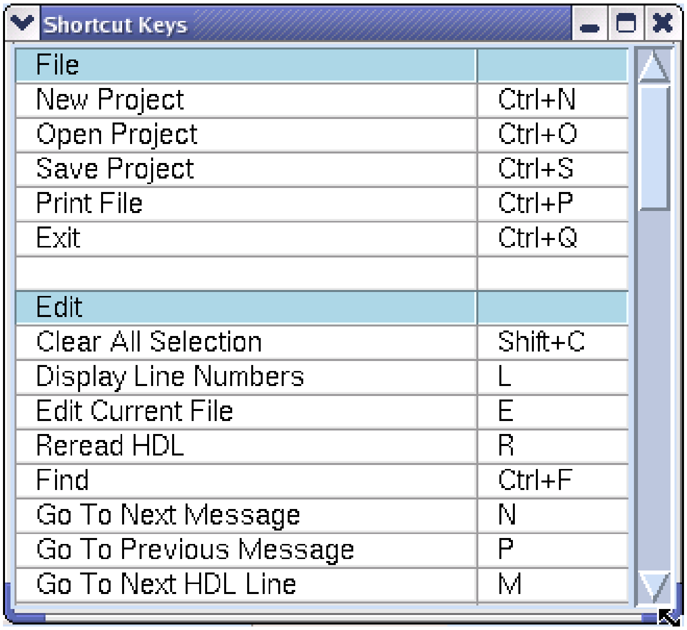


Design File	
	Verilog File
	Vhdl File
	Unknown File
	SgLib File
	GatesLib File
	PLib File
	LEF File
	DEF File
	Stopped File
Design Unit	
	Synthesized design units
	Unsynthesized design units
	Black-Box design units not
	Design units not at leaf-level
	Leaf-level design units with

FIGURE 28. Icon Quick Reference

Help > Shortcut Keys

This menu option invokes the Shortcut Keys window that contains the names of the shortcut keys combination used to open different SpyGlass windows.



File	
New Project	Ctrl+N
Open Project	Ctrl+O
Save Project	Ctrl+S
Print File	Ctrl+P
Exit	Ctrl+Q
Edit	
Clear All Selection	Shift+C
Display Line Numbers	L
Edit Current File	E
Reread HDL	R
Find	Ctrl+F
Go To Next Message	N
Go To Previous Message	P
Go To Next HDL Line	M

FIGURE 29. Shortcut Keys

You can also use the <*Ctrl*>+*F1* key combination on your keyboard to invoke the Shortcut Keys window.

Help > Atrenta Console User Guide

This menu option displays the *Atrenta Console User Guide* document.

Help > Atrenta Console Reference Guide

This menu option displays the *Atrenta Console Reference Guide* document.

Help > BuiltIn Rules Reference

This menu option displays the *BuiltIn Rules Reference Guide* document.

Help > Methodology Guides

This menu option displays a sub-menu that lists all methodology guides. Select the required option from the sub-menu to display the corresponding methodology guide.

Help > SpyGlass Release Summary

This menu option displays the *SpyGlass Release Summary* document that contains a summary of enhancements made in the current SpyGlass version.

Help > SpyGlass Release Notes

This menu option displays the *SpyGlass Release Notes* document that contains details about all enhancements made in the current SpyGlass version.

Help > SpyGlass KPNS

This menu option displays the *SpyGlass Known Problems and Solutions* document that contains details about known problems in SpyGlass and their corresponding solution.

Help > Online Support

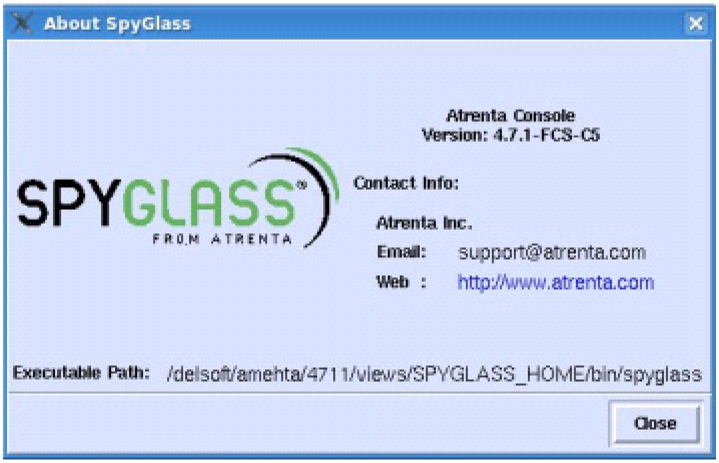
This menu option provides online support for various details, such as updates, training, and FAQ.

When you select this menu, a sub-menu appears that lists various options to provide different types of online support.

Help > About SpyGlass

This menu option is used to display Atrenta Console version number and Atrenta Contact information.

When you select this menu option, the About SpyGlass window is displayed.



Atrenta Console
Version: 4.7.1-FCS-C5

Contact Info:

Atrenta Inc.
Email: support@atrenta.com
Web : http://www.atrenta.com

Executable Path: /delsoft/amehta/4711/views/SPYGLASS_HOME/bin/spyglass

FIGURE 30. About SpyGlass

Click the *Close* button to close this window.

The Methodology Configuration System Menu Bar

The Methodology Configuration System menu bar provides you access to the functions that have been categorized into the following menus:

- *File Menu*
- *Edit Menu*
- *Tools menu*
- *Help Menu*

File Menu

When you click the *File* menu, the following options appear:

File > New Methodology	*File > Open Methodology*
File > Save Methodology	*File > Save Methodology As*
File > Reload Methodology	*File > Methodology Properties*
File > Close	

File > New Methodology

This menu option is used to create a new methodology.

To know details on creating a new methodology, refer to the *Creating a New Methodology* topic of *Atrenta Console User Guide*.

File > Open Methodology

This menu option is used to open an existing methodology.

When you select this menu option, Atrenta Console first prompts you to save the currently loaded methodology by displaying the *Save Methodology* dialog. Once you perform the required actions in this dialog, Atrenta Console displays the *Open Methodology* dialog, as shown in the following figure:

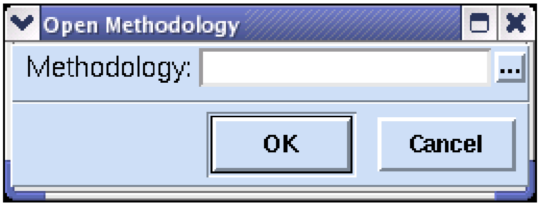


FIGURE 31. Open Methodology

In the above dialog, you can specify the path of the directory where the methodology resides in the *Methodology* text field. Alternatively, click the (...) button, and browse to the directory where the methodology is present.

NOTE: *You can also use the <Ctrl> + <O> key combination on the keyboard to display the Open Methodology dialog.*

File > Save Methodology

This menu option is used to save the currently loaded methodology.

When you select this menu option, the *Save Methodology* dialog appears, as shown in the following figure:

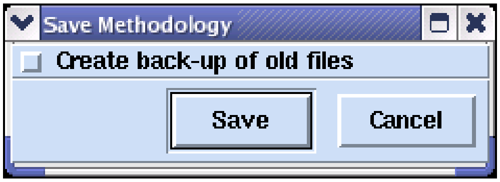


FIGURE 32. Save Methodology

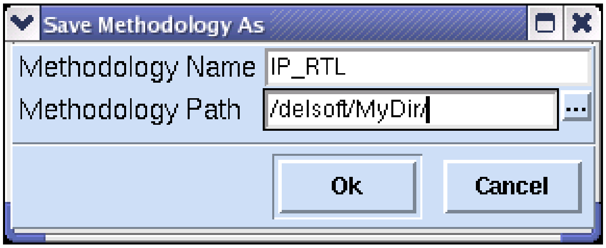
In the above dialog, click the *Save* button to save the methodology. If you want to create a backup of the methodology, select the *Create back-up of old files* option, and then click the *Save* button.

NOTE: *You can also use the <Ctrl> + <S> key combination on the keyboard to open the Save Methodology dialog.*

File > Save Methodology As

This menu option is used to save the current methodology to a different location.

When you select this menu option, the *Save Methodology As* dialog appears, as shown in the following figure:



Methodology Name IP_RTL

Methodology Path /delsoft/MyDir/

FIGURE 33. Save Methodology As

In the above dialog, you can specify the methodology name and path in appropriate fields. Once you specify the required details, click the *OK* button to save the changes.

File > Reload Methodology

This menu option is used to reload a methodology.

File > Methodology Properties

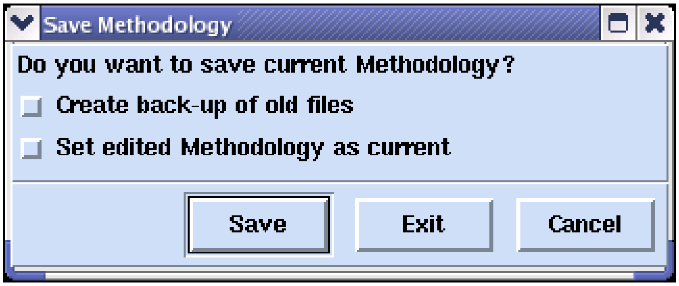
This menu option is used to view/change methodology properties, such as name, path, and help descriptions.

For details on modifying the methodology properties, refer to the *Modifying a Methodology* topic of *Atrenta Console User Guide*.

File > Close

This menu option is used to close the *Methodology Configuration System* window.

When you select this menu option, Atrenta Console prompts you to save the methodology by displaying the *Save Methodology* dialog, as shown in the following figure:



Do you want to save current Methodology?

□ Create back-up of old files

□ Set edited Methodology as current

FIGURE 34. Save Methodology

If you want to load the edited methodology in the *Goal Selection* window, select the *Set edited Methodology as current* option. In addition, if you want to create a backup of the old methodology files, select the *Create back-up of old files* option. Once you select the required options, click the *Save* button. However, if you want to exit MCS without saving the changes in the currently loaded methodology, click the *Exit* button.

NOTE: *You can also use the <Ctrl> + <Q> key combination on the keyboard to open the Close confirmation dialog.*

Edit Menu

When you click the *Edit* menu, the following menu options appear:

Edit > Add Sub-Methodology	*Edit > Add New Goal*
Edit > Import Goal(s)	*Edit > Copy Sub-Methodology*
Edit > Paste Sub-Methodology	*Edit > Copy Goal*
Edit > Paste Goal	

Edit > Add Sub-Methodology

This menu option is used to add a sub-methodology to an existing methodology.

For more details, refer to the *Adding a Sub-Methodology* topic of *Atrenta Console User Guide*.

NOTE: *You can also use the <Ctrl> + <M> key combination on the keyboard to open the Add Sub-Methodology dialog.*

Edit > Add New Goal

This menu option is used to add a goal to the selected methodology. For details on adding a new goal, refer to the *Adding Goals* topic of *Atrenta Console User Guide*.

Edit > Import Goal(s)

This menu option is used to import goal/goals into the selected the methodology.

For details on importing goals, refer to the *Importing Goals* topic of *Atrenta Console User Guide*.

Edit > Copy Sub-Methodology

This menu option is used to copy a sub-methodology to create multiple instances of a sub-methodology.

NOTE: *This menu option is visible in the Edit menu when you have selected a sub-methodology in the Goals section.*

When you copy a sub-methodology, it is displayed in italics in the *Goals* section of the MCS window.

When you copy a sub-methodology, the goals displayed under the sub-methodology tree are also copied.

You can also copy a sub-methodology by performing any of the following:

- Right-click a sub-methodology, and select the *Copy Sub-Methodology* shortcut menu option.
- Use the *<Ctrl> + C* key combination on your keyboard to copy a sub-methodology.

Edit > Paste Sub-Methodology

Use this menu option is used to create multiple instances of the copied sub-methodology.

NOTE: *This menu option is visible in the Edit menu only if you have copied a sub-methodology from the Goals section.*

You can also paste a sub-methodology by performing any of the following:

- Right-click on a sub-methodology, and select the *Paste Sub-Methodology* option from the shortcut menu.
- Use the <*Ctrl*> +V key combination on your keyboard to paste the sub-methodology in a different location.

Edit > Copy Goal

This menu option is used to copy goals from one sub-methodology to another thus enabling you to create multiple instances of the same goal.

When you copy a goal, it is displayed in italics in the *Goals* section of the MCS window.

You can also copy a goal by performing any of the following:

- Right-click on a goal, and select the *Copy Goal* option from the shortcut menu.
- Use the <*Ctrl*> +C key combination on your keyboard to paste a goal in a different location.

Edit > Paste Goal

This menu option is used to create another instance of the same goal across different sub-methodologies

You can also paste a goal by performing any of the following:

- Right-click a goal, and select the *Paste Goal* shortcut menu option.
- Use the <*Ctrl*> + V key combination on your keyboard to paste a goal under a different sub-methodology.

Tools menu

When you click the *Tools* menu, the following options appear:

Tools > Compare	*Tools > Preferences*

Tools > Compare

When you select this menu option, the following options appear in the sub-menu:

- *Goal(s) with Goal(s)*
- *Methodologies*

Goal(s) with Goal(s)

Select this menu option to compare two methodologies or goal files.

The comparison result shows the comparison between the rules and parameters present in the two methodologies or goal files. See *Figure 38*.

One methodology or goal is called the reference data and the other methodology or goal is called the target data. The target data is compared with the reference data.

When you select this option, the *Methodology Comparison* window appears, as shown in the following figure:

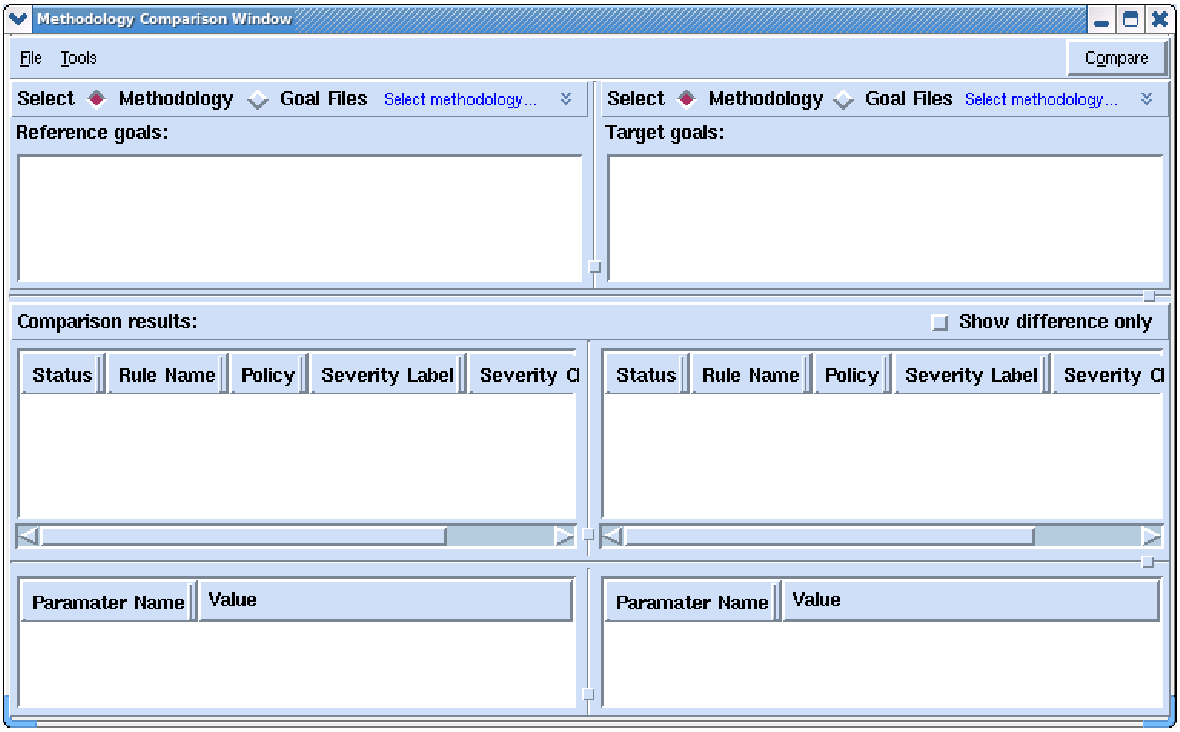
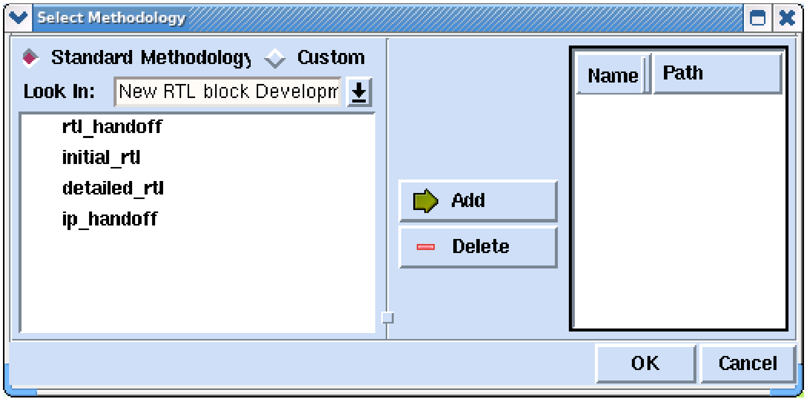


FIGURE 35. Methodology Comparison Window

To compare two methodologies, perform the following steps:

1. Select the *Methodology* option in the *Reference goals* section.
2. Click the *Select Methodology* link.

The *Select Methodology* dialog appears. The following figure shows the *Select Methodology* dialog:



● Standard Methodology ○ Custom

Look In: New RTL block Developm

rtl_handoff
initial_rtl
detailed_rtl
ip_handoff

FIGURE 36. Select Methodology

3. In the above dialog, select the *Standard Methodology* or the *Custom* option depending upon the type of methodology you want to set as the reference data.
4. Select a methodology in the left-most pane and click the *Add* button.

   The name and path of the selected methodology appears in the right-most pane in this dialog.
5. Click the *OK* button to close the above dialog.

   After performing the above steps, Atrenta Console loads the details of the selected methodology in the *Reference goals* section, as shown in the following figure:

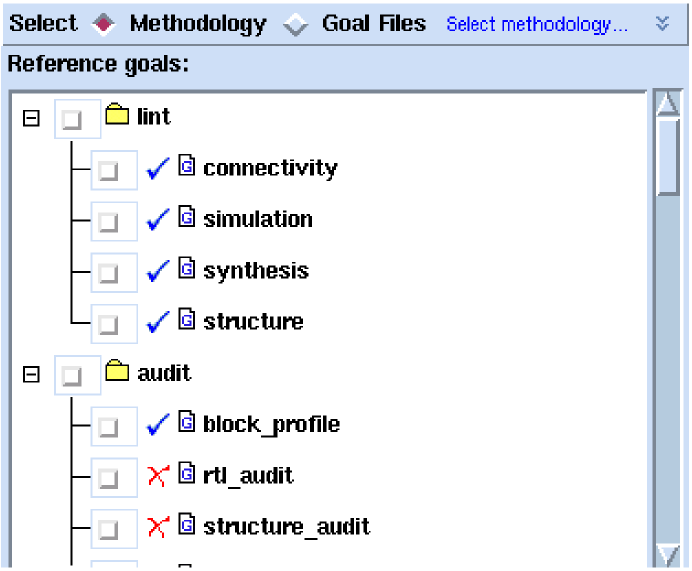


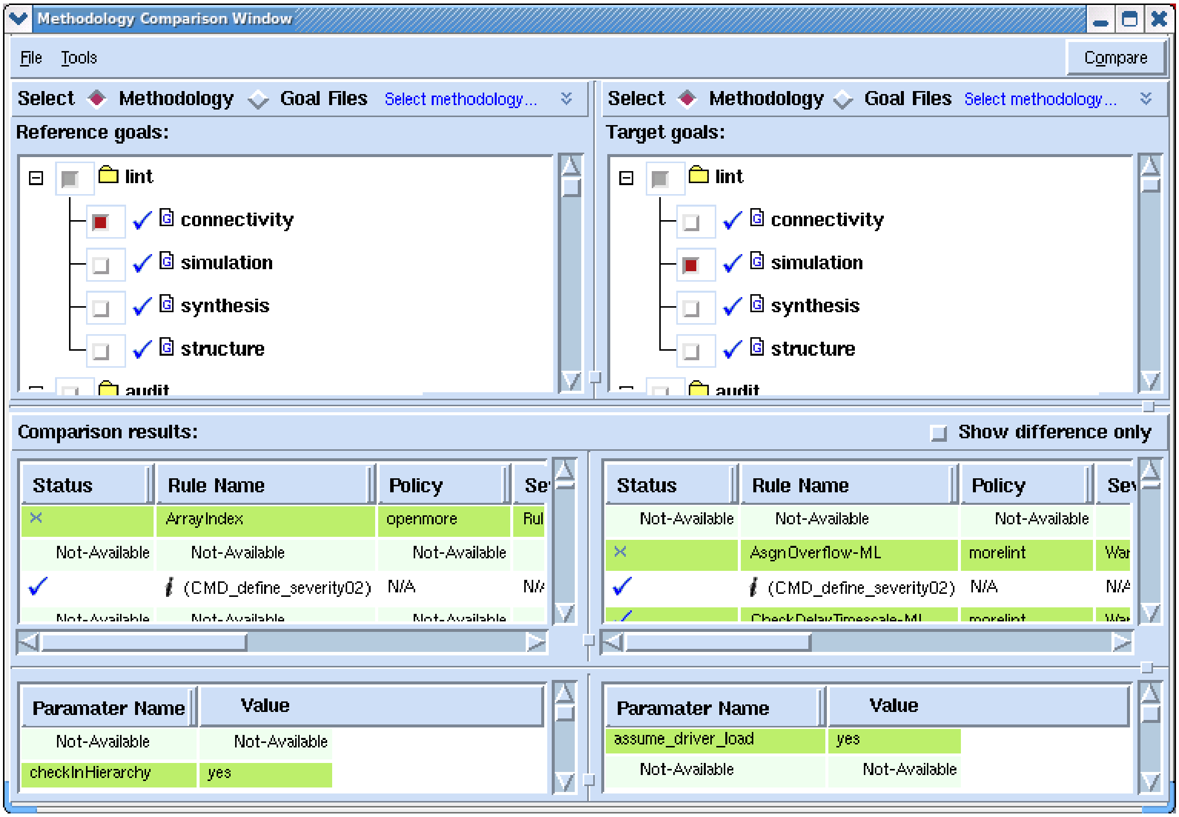
FIGURE 37. Select Reference Goals

6. Select the *Methodology* option in the *Target goals* section.
7. Repeat steps *3* to *5* to load the details of the target methodology.
8. Select the goals to be compared from the *Reference goals* section and the *Target goals* section.

NOTE: *You can specify a map file to trace back the reference of the new goal (present in GuideWare 2.0) to the original goal (present in GuideWare 1.0). For details on the map file, refer to the* ***Map File*** *section of the Atrenta Console User Guide.*

9. Click the *Compare* button.

After performing the above steps, the comparison results appear in the *Comparison Results* section, as shown in the following figure:



Status	Rule Name	Policy	Se
✕	ArrayIndex	openmore	Rul
Not-Available	Not-Available	Not-Available	
✓	(CMD_define_severity02)	N/A	N/A
Not-Available	Not-Available	Not-Available	

Status	Rule Name	Policy	Sev
Not-Available	Not-Available	Not-Available	
✕	AsgnOverflow-ML	morelint	War
✓	(CMD_define_severity02)	N/A	N/A
✓	CheckDelayTimescale-ML	morelint	War

Paramater Name	Value
Not-Available	Not-Available
checkInHierarchy	yes

Paramater Name	Value
assume_driver_load	yes
Not-Available	Not-Available

FIGURE 38. Methodology Comparison Results

Similarly, you can compare two goals by using the above window.

Methodologies

Select this option to compare two methodologies and to merge the contents of one goal into another.

When you select this menu option, the *Methodology Comparison* dialog appears, as shown in the following figure:

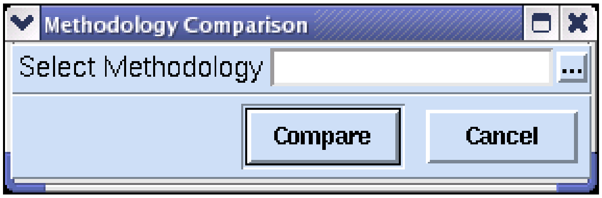


FIGURE 39. Select Methodology for Comparison

For comparison, the currently selected methodology is considered as a reference. In this dialog, specify the location of the methodology that you want to compare in the *Select Methodology* field, and click the *Compare* button.

The differences in the methodology will be shown in a hierarchical format. The reference methodology is displayed in the left column of the Methodology Comparison window and the methodology being compared is displayed in the right column of the Methodology Comparison window.

The differences are displayed with levels like goal, product, rules, parameter, and parameter value as shown below.

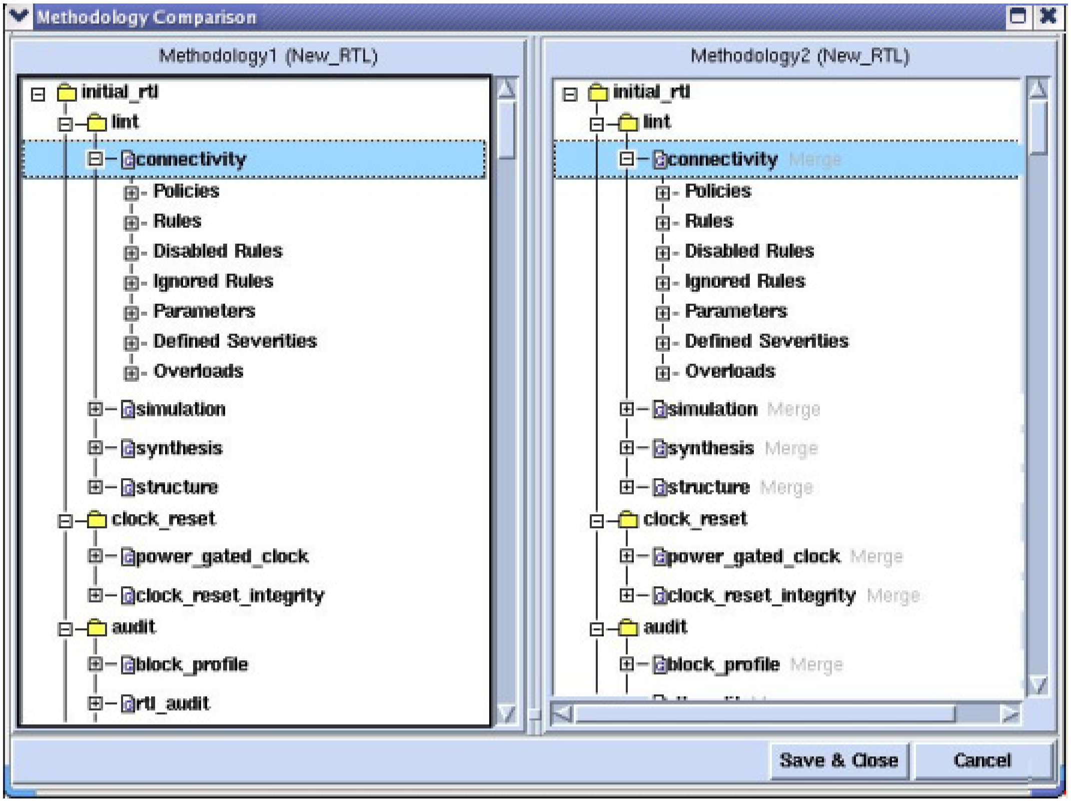


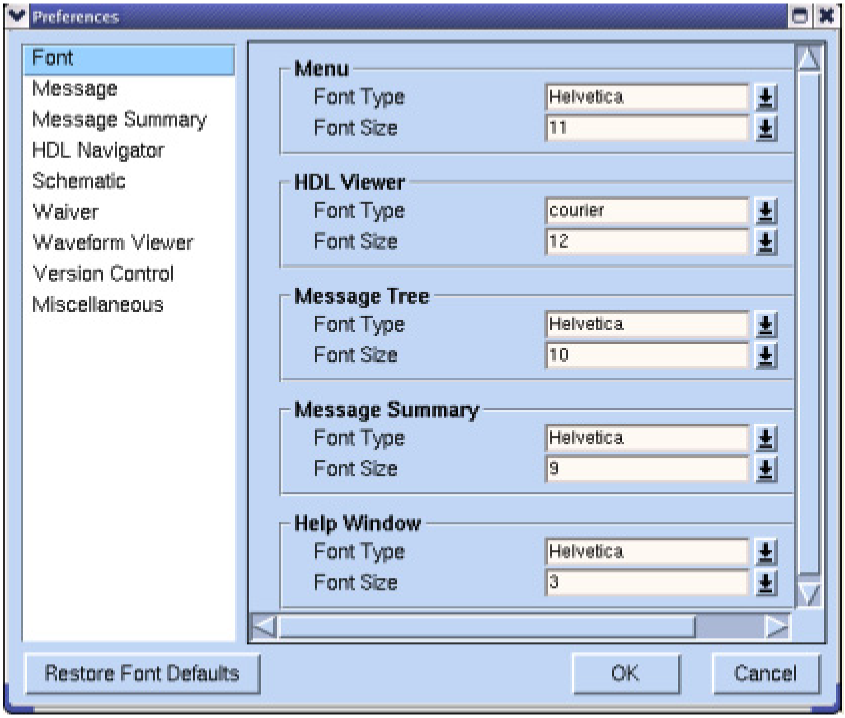
FIGURE 40. Methodology Comparison Result Levels

You can also merge settings of a methodology being compared on basis of goals into the base/reference methodology.

To do this, click the *Merge* link on the methodology that is being compared. Then the changes made in the methodology are merged into the reference methodology.

Tools > Preferences

This menu option displays the *Preferences* dialog, as shown in the following figure:



Menu
Font Type Helvetica
Font Size 11

HDL Viewer
Font Type courier
Font Size 12

Message Tree
Font Type Helvetica
Font Size 10

Message Summary
Font Type Helvetica
Font Size 9

Help Window
Font Type Helvetica
Font Size 3

FIGURE 41. Preferences - Font

In the above dialog, you can provide the required settings for *Font*, *Message*, *Message Summary*, etc.

Help Menu

When you click the *Help* menu, the following options appear:

Help > Methodology Configuration System Help	*Help > On-line Manuals*
Help > On-line Help	*Help > Icons Quick Reference*
Help > About SpyGlass	

Help > Methodology Configuration System Help

This menu option is used to display the online help for Methodology Configuration System.

Help > On-line Manuals

This menu option is used to display the *Documentation* window that contains links to various on-line manuals in SpyGlass. To open a particular manual, double-click on that manual name.

Help > On-line Help

This menu option is used to display the online help of SpyGlass documentation. When you select this menu option, Atrenta Console displays the browser window in which you can traverse through the online help on different topics.

Help > Icons Quick Reference

This menu option invokes the *Icons Quick Reference* window that contains a brief description of various icons used in the *Methodology Configuration System* window.

Help > About SpyGlass

This menu option is used to invoke the *About SpyGlass* window that displays various details such as Atrenta Console version number and Atrenta Contact information.

Windows and Panes in Atrenta Console

Overview

This chapter describes the following windows and panes in Atrenta Console:

The File/Module/Instance/Constraints Pane

This pane is used to display source and library files (in the File view page), design hierarchy (in the Module view page), Constraints files and Synopsys Design Constraints files (in the Constraints view page), and design unit instances (in the Instances view page).

Select the required view from the Show: drop-down list, as shown in the following figure:

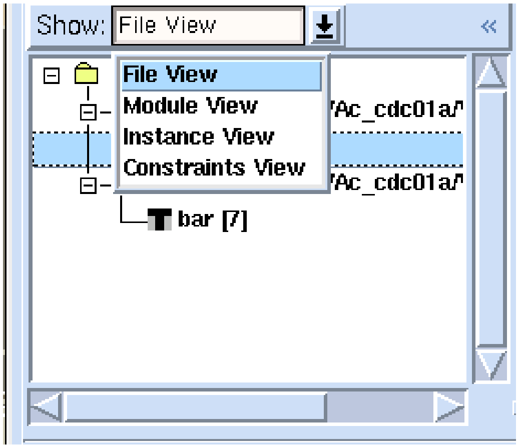


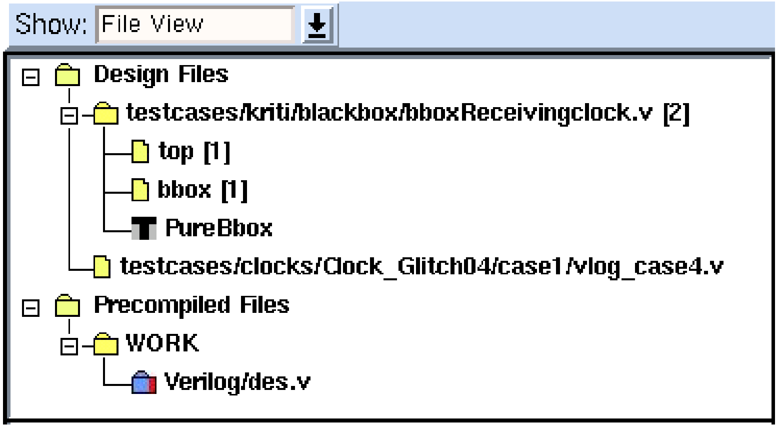
FIGURE 1. The File/Module/Instance/Constraints Pane

NOTE: *You can show/hide this page by clicking the Show HDL Viewer/Hide HDL Viewer link.*

File View Page

The File View page displays the list of source and library files under appropriate groups.

The following figure displays the File View page:



Show: File View

- Design Files
  - testcases/kriti/blackbox/bboxReceivingclock.v [2]
    - top [1]
    - bbox [1]
    - PureBbox
  - testcases/clocks/Clock_Glitch04/case1/vlog_case4.v
- Precompiled Files
  - WORK
    - Verilog/des.v

FIGURE 2. File View

Precompiled Files in the File View Page

The File View page lists all the precompiled files under the Precompiled Files node. Under this node, the precompiled files are displayed in a hierarchical structure based on their logical library names.

Different icons are used to identify different types of precompiled files, as explained in the following table:

Icon	Indicates
🟦	Precompiled files that are used in the current SpyGlass run
🟦🟥	Precompiled files that are not used in the current SpyGlass run but are a part of the libhdlfile specification
🔒	Precompiled files that are encrypted

To add/remove a precompiled file, right-click on that file and select the

Add/Remove Precompile File option from the shortcut menu. The Precompile File Mapping dialog appears in which you can delete the required mapping.

File Names in the File View Page

Atrenta Console displays the source and library file names based on their position relative to the current working directory at the time the file was added.

If you have added files located below the current working directory in the directory tree (call this Scenario1), Atrenta Console lists the files in the *File View* page by their position in the directory path relative to the position of the project file. For example, if the current working directory's full path name is /SPY/verilog/version1/ and the file to be analyzed has the full path name /SPY/verilog/version1/sourcefiles/chip.v, the *File View* page entry will be sourcefiles/chip.v.

On the other hand, if you have added files located above the current working directory in the directory tree (call this Scenario2), the files are listed in the *File View* page by their position in the full directory path position. For example, if the current working directory's full path name is /SPY/verilog/version1/sourcefiles/analysis1/ and the file to be analyzed has the full path name /SPY/verilog/version1/sourcefiles/chip.v, the File view page entry will be as follows:

/SPY/verilog/version1/sourcefiles/chip.v

This difference in two scenarios will become relevant in the following predicaments:

In Scenario1, suppose that the working directory (/SPY/verilog/version1/) does not have a project file. Now consider that you invoke Atrenta Console GUI from the working directory. Then, you add the /SPY/verilog/version1/sourcefiles/chip.v file as a source file (once you set up the rest of the options (such as design read and goal selection), and run SpyGlass analysis.

Suppose, that you do not run analysis at this point in time. Instead, you save the setup that you have made in a profile file (such as testcase1.spp) for analysis at a future time (such as later that night).

In Scenario2, suppose that the working directory (/SPY/verilog/version1/sourcefiles/analysis1/) does not have a project file. Now consider that you launch Atrenta Console GUI from the working directory. Then, you add the /SPY/verilog/version1/sourcefiles/chip.v file as a source file, once you set up the rest of the options (such as design read and goal selection), you are ready to run the SpyGlass analysis.

Suppose, again, that you DO NOT run the analysis at this point in time. Instead, you save the setup that you have made in a profile file (such as testcase2.spp) for analysis at a future time (such as later that night).

Now, the predicament occurs when the current working directory's full path name has changed and if someone were to rename the /SPY/verilog/version1 directory to /SPY/verilog/version2. If everything else were left the same, testcase1.spp from Scenario1 would still be valid while testcase2.spp from Scenario2 would be unable to find the chip.v file.

The reason for this result is the difference in how the source files are recorded. With the relative relationship that the profile file and the source files maintain in Scenario1, changing the path does not affect the relationship of the files. However, with Scenario2, there is no relative relationship, and a change in the full directory path will cause the profile file to look for the source files in a directory that no longer contains them (or even exists).

Viewing Files

When you double-click on a file name in the *File View* page, Atrenta Console displays the corresponding file in the *Source* section of the *HDL Viewer* with the cursor at the first line of the design unit description. Commented lines at the beginning of the file are skipped.

Right-Click Menu Options in File View Page

Different options appear when you right-click on a file name, module, or precompiled file in the *File View* page.

Right-Clicking on a File Name

When you right-click on a file name in the *File View* page, the following options appear in the shortcut menu:

Find File

Select this menu option to search for a file in the *File View* page.

When you select this option, the *Search* dialog appears in which you can specify the required search criteria.

Add File

Select this menu option to add a file in the *File View* page.

When you select this menu option, the *Add File(s)* dialog appears in which you can select the required file to be added.

Edit File

Select this menu option to edit the selected file appearing in the *File View* page.

When you select this menu option, an editor window appears in which you can make the required updates.

You can specify the type of editor window to be displayed in the *Specify external editor program* field of the *Miscellaneous Page* in the *Preferences* dialog. If you have not specified any value in this field, Atrenta Console invokes the text editor specified by the EDITOR environment variable.

If you edit and save a source file listed in the *File View* page, the file name is displayed in red color to indicate that the file has been modified since last analysis run.

Delete File

Select this menu option to remove the selected file from the *File View* page.

The deleted file is only removed from the *File View* page and is not physically deleted from the hard disk. In addition, violation messages corresponding to the deleted file remain displayed until you run the SpyGlass analysis again.

Copy File Path

Select this menu option to copy the path of the selected file.

Waive Rules By File

Select this menu option to waive rule messages corresponding to the selected file.

When you select this option, the *Waiver Editor Window* appears in which you can specify the required details.

Set 'Stop File'

Select this menu option to set the selected file as a stop file.

The stop file is indicated with the stop file icon (🛑) prefixed to the design file name.

Set 'Ignore File'

Select this menu option to ignore the selected file so that all design units specified in that file are ignored during SpyGlass analysis.

The ignored file is indicated with the ℹ icon prefixed to the design file name.

Right-Clicking on a Module

When you right-click on a module in the *File View* page, the following options appear in the shortcut menu:

Find File

Functionality of this option is similar to *Find File* option.

Copy

Select this menu option to copy the name or path of the selected module.

When you select this option, a sub-menu appears containing two options, *Module Name* and *File Path*.

Set Top Module

Select this menu option to set a module as a top module.

When you select this option, the *Set Module Top* dialog appears with the name of the module displayed in the *Top Module* field. The following figure illustrates the *Set Module Top* dialog:

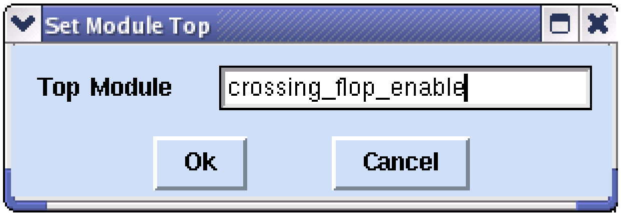


FIGURE 3. Set Top Module

Click the *Ok* button to set the module as top module. Atrenta Console then prefixes the specified module with the T icon to indicate it as the top module.

If a top module already exists in the design hierarchy and you try to set another module as the top module, Atrenta Console displays a warning message to indicate that a top module already exists with a higher/lower priority. You can set the new module as the top module or cancel the action.

Remove Top Module

Select this menu option to not consider the selected module as a top module.

NOTE: *To trace the top module setting/removing GUI operations, refer to the spyglass_cmdline_debug.log file.*

Set Stop Module

Select this menu option to set a module as a stop module.

When you select this option, the *Set Module Stop* dialog appears with the name of the module displayed in the *Stop Module* field. The following figure illustrates the *Set Module Top* dialog:

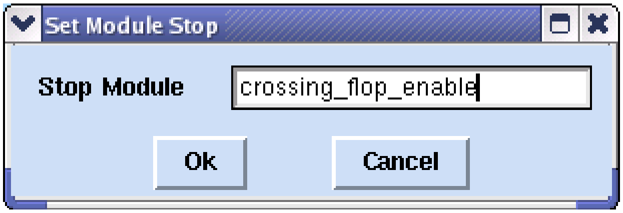


FIGURE 4. Set Stop Module

Click the *Ok* button to set the module as stop module. Atrenta Console then prefixes the specified module name with the S icon to indicate it as a stopped module.

Remove Stop Module

Select this menu option to not consider the selected module as a stop module.

NOTE: *To trace the stop module setting/removing GUI operations, refer to the spyglass_cmdline_debug.log file.*

Right-Clicking on a Precompiled File

When you right-click on a precompiled file in the *File View* page, the following options appear in the shortcut menu:

Find File

Functionality of this option is similar to *Find File* option.

Edit File

Functionality of this option is similar to *Edit File* option.

Add/Remove Precompile File

Select this menu option to add/remove a precompiled from a logical library.

When you select this menu option, the *Precompile File Mapping* dialog appears in which you can make the required modifications.

Copy File Path

Functionality of this option is similar to *Copy File Path* option.

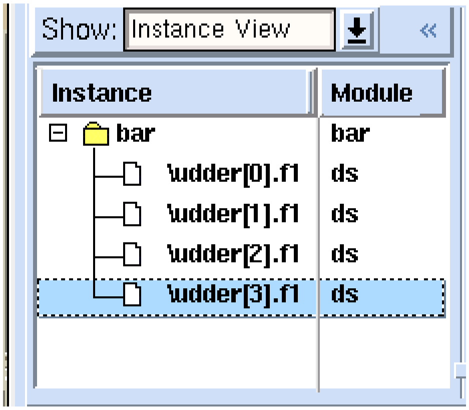
Waive Rules By File

Functionality of this option is similar to *Waive Rules By File* option.

Instance View Page

The *Instance View* page is the instance hierarchy browser that displays design unit instances in each source and library file that you are analyzing with the current project file. This page remains blank until SpyGlass analysis has been completed or a valid project file is loaded.

The following figure illustrates the *Instance View* page:



Show: Instance View

Instance	Module
bar	bar
\udder[0].f1	ds
\udder[1].f1	ds
\udder[2].f1	ds
\udder[3].f1	ds

FIGURE 5. Instance View

Right-clicking over the instance displays a right-click menu as discussed in the *Right-Click Menu Options of Instance View Page* section:

When you double-click on an instance name, the corresponding source file containing that instance appears in the source code section of the HDL Viewer. In addition, Atrenta Console highlights the first line of the instance description (initial commented lines are skipped). Similarly, the corresponding gate is also highlighted in *The Modular Schematic Window*.

The details of the above page are discussed in the following points:

- The page displays the instance hierarchy containing multiple levels and dependencies of design unit instances that you have analyzed using SpyGlass.
- The first column displays instance names of a design unit.
- The second column displays module/design unit corresponding to the instances listed in the first column.
- Top-level instances are displayed as either root-level instances or leaf-level instances, depicted as yellow folders in the hierarchical tree.
- Root-level instances are instances that contain child instances (instances of other design units). A child instance can contain leaf-level instances, instances of black box modules, or children of its own.

- Leaf-level instances are instances that contain no children. Leaf-level instances can be top-level instances, or children of other design unit instances (depicted as white file icons).
- Un-synthesizable design unit instances are the instances that contain coded behavior and descriptions, but cannot be synthesized. In the hierarchical tree, root-level un-synthesizable instances are depicted as green folders and leaf-level un-synthesizable instances are depicted as green files.
- Instances of black box modules, depicted with black file icons in the hierarchical tree, are design unit instances that contain no coded behavior or description.

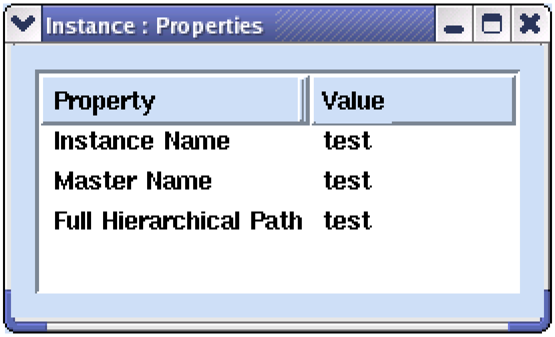
Right-Click Menu Options of Instance View Page

When you right-click on an instance in the *Instance View* page, the following options appear in the shortcut menu:

Properties

Select this menu option to view the properties of listed instances.

When you select this option, the *Instance : Properties* dialog appears, as shown in the following figure:



Instance : Properties

Property	Value
Instance Name	test
Master Name	test
Full Hierarchical Path	test

FIGURE 6. Instance View - Properties

The above dialog displays the instance name, master module name, and the complete hierarchical path of the instance.

You can individually copy contents of this window by using the right-click shortcut menu options or *<Ctrl>+<C>* key combination.

Copy

Select this menu option to copy instance name, complete instance path, or module file name.

When you select this option, a sub-menu appears containing three options, *Instance Name*, *Instance Path*, and *Module File Name*.

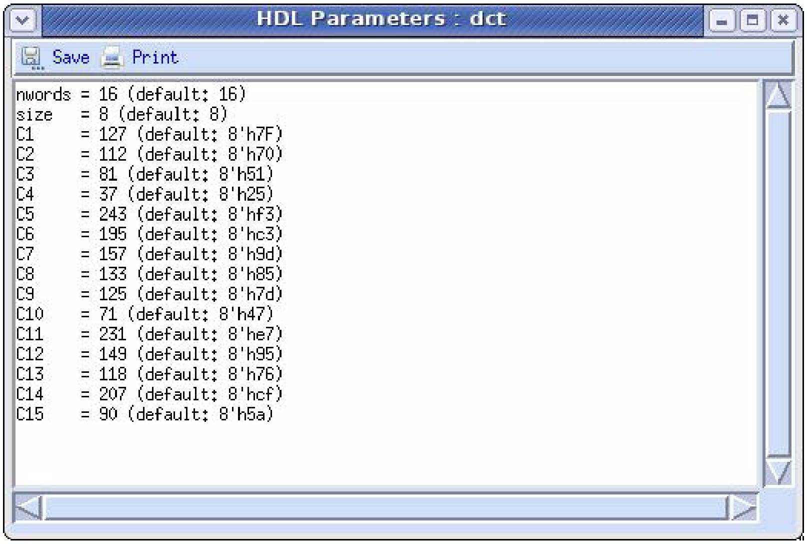
Show Blackbox Info

Select this menu option to view information about black box modules.

For details, see *Black Box Viewer Window*.

Show HDL Parameters

Shows the list of HDL parameters for an instance. This option is available only for those modules or instances which have at least one HDL parameter defined. Following figure illustrates the sample HDL Parameters window for the module, `dct`:

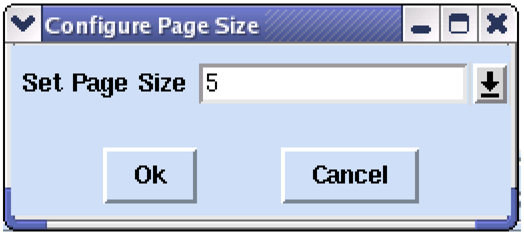


```
nwords = 16 (default: 16)
size   = 8 (default: 8)
C1     = 127 (default: 8'h7F)
C2     = 112 (default: 8'h70)
C3     = 81 (default: 8'h51)
C4     = 37 (default: 8'h25)
C5     = 243 (default: 8'hf3)
C6     = 195 (default: 8'hc3)
C7     = 157 (default: 8'h9d)
C8     = 133 (default: 8'h85)
C9     = 125 (default: 8'h7d)
C10    = 71 (default: 8'h47)
C11    = 231 (default: 8'he7)
C12    = 149 (default: 8'h95)
C13    = 118 (default: 8'h76)
C14    = 207 (default: 8'hcf)
C15    = 90 (default: 8'h5a)
```

Configure Page Size

Select this menu option to set the number of instances that should be displayed for each level in the instance hierarchy.

When you select this option, the *Configure Page Size* dialog appears, as shown in the following figure:



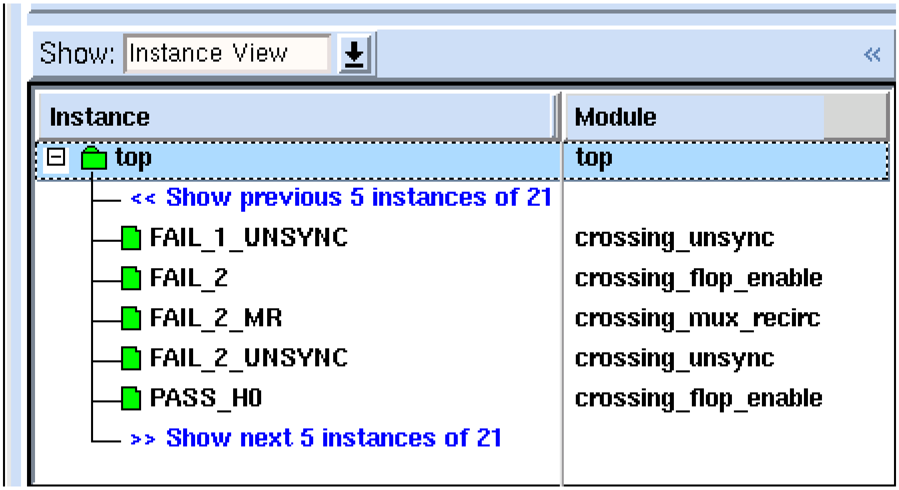
Set Page Size 5

FIGURE 7. Configure Page Size

In the above dialog, specify the required value in the *Set Page Size* field.

For a design unit instance, if the number of instances in a hierarchy level exceed the current page size, a new link(s) appear in the *Instance View* page that redirects you to the next and previous set of instances.

A sample figure is shown below:



Show: Instance View

Instance	Module
top	top
<< Show previous 5 instances of 21	
FAIL_1_UNSYNC	crossing_unsync
FAIL_2	crossing_flop_enable
FAIL_2_MR	crossing_mux_recirc
FAIL_2_UNSYNC	crossing_unsync
PASS_H0	crossing_flop_enable
>> Show next 5 instances of 21	

FIGURE 8. Instance View - Next and Previous Instances

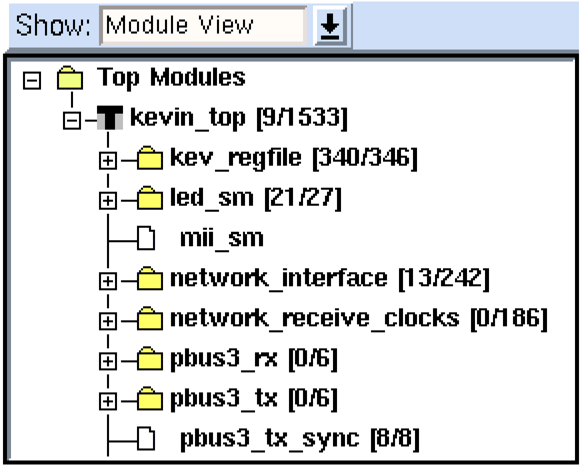
Module View Page

The *Module View* page displays the modules and instances in each source and library file that you are analyzing with the current project.

This page remains blank until SpyGlass analysis has been completed or a valid project file is loaded.

To view the *Module View* page, select the *Module View* option from the *Show:* drop-down list.

The following figure displays the *Module View* page:



Show: Module View

```
Top Modules
  kevin_top [9/1533]
    kev_regfile [340/346]
    led_sm [21/27]
    mii_sm
    network_interface [13/242]
    network_receive_clocks [0/186]
    pbus3_rx [0/6]
    pbus3_tx [0/6]
    pbus3_tx_sync [8/8]
```

FIGURE 9. Module View

In the above view, a module name appears in the following format:

<module-name> [x/y]

Where:

- x refers to the number of violations reported for the module *<module-name>*.
- y refers to x + (number of violations reported within the hierarchy of the module *<module-name>*)

Right-clicking over the module/instance displays a right-click menu as discussed in the *Right-Click Menu Options of Module View Page* section.

When you double-click on a module, the source code of that module appears in the source section with the first line of the module description highlighted (initial commented lines are skipped). Similarly, the corresponding gate is also highlighted in *The Modular Schematic Window*.

Details of the above page are discussed in the following points:

- The design hierarchy in this page displays the multiple layers and dependencies of the modules that you have analyzed using SpyGlass through a hierarchical tree.
- Top-level modules are displayed as either root-level modules or leaf-level modules, depicted as yellow folders in the hierarchical tree.
- Root-level modules are modules that contain child modules (instances of other modules). A child module can contain leaf-level modules, black box modules, or children of its own.
- Leaf-level modules are modules that contain no children. Leaf-level modules can be top-level modules, or children of other modules (depicted as white file icons).
- Un-synthesizable modules are the modules that contain coded behavior and descriptions, but cannot be synthesized. In the design hierarchy, root-level un-synthesizable modules are depicted as green folders and leaf-level un-synthesizable modules are depicted as green files.
- Black box modules, depicted as black squares with a white “X” in the hierarchical tree, are modules that contain no coded behavior or description. These modules are usually children of other modules.
- Encrypted design modules are indicated by the 🔒 icon.

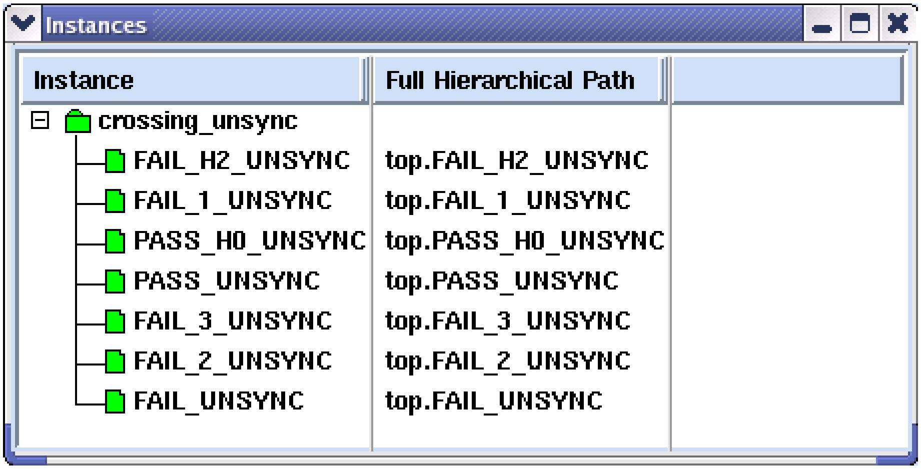
Right-Click Menu Options of Module View Page

When you right-click on a module in the *Module View* page, the following options appear in the shortcut menu:

Show Instance List

Select this menu option to view a list of instances in the design.

When you select this option, the *Instances* window appears, as shown in the following figure:



Instance	Full Hierarchical Path
⊟ crossing_unsync	
FAIL_H2_UNSYNC	top.FAIL_H2_UNSYNC
FAIL_1_UNSYNC	top.FAIL_1_UNSYNC
PASS_H0_UNSYNC	top.PASS_H0_UNSYNC
PASS_UNSYNC	top.PASS_UNSYNC
FAIL_3_UNSYNC	top.FAIL_3_UNSYNC
FAIL_2_UNSYNC	top.FAIL_2_UNSYNC
FAIL_UNSYNC	top.FAIL_UNSYNC

FIGURE 10. Module View - Instance List

The above window lists instances along with complete hierarchical path of each instance.

Encrypted instances are indicated by the 🔒 icon.

When you right-click on an instance in the above window, following options appear in the right-click menu:

- *Properties*

  Select this option to display the *Instance List: Properties* window that shows instance properties, such as instance name, master module name, and complete hierarchical path of the instance.

- *Copy Instance Path*

  Select this option to copy instance path. You can then paste the path at a desired location.

- *Save Instance List*

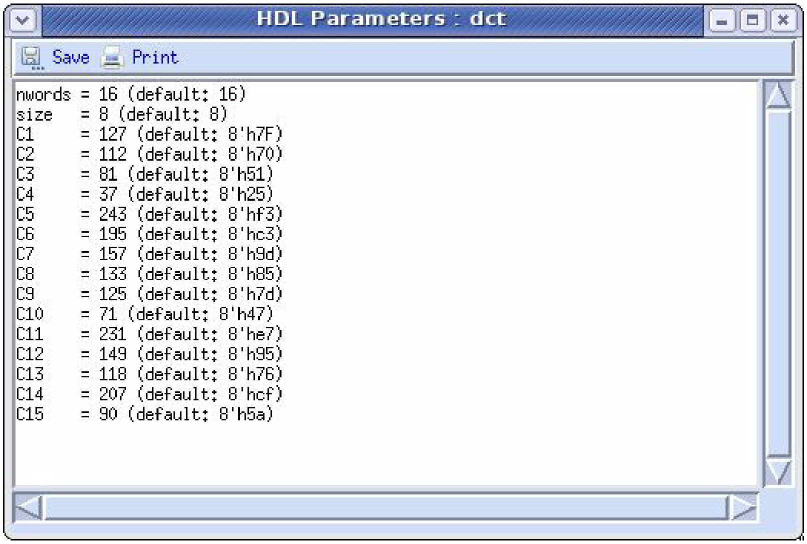
  Select this option to save the instance list at the desired location.

- *Configure Page Size*

  Select this option to specify the maximum number of instances to be displayed at a time in the *Instances* window.

Show HDL Parameters

Shows the list of HDL parameters for an instance. This option is available only for those modules or instances which have at least one HDL parameter defined. Following figure illustrates the sample HDL Parameters window for the module, dct:



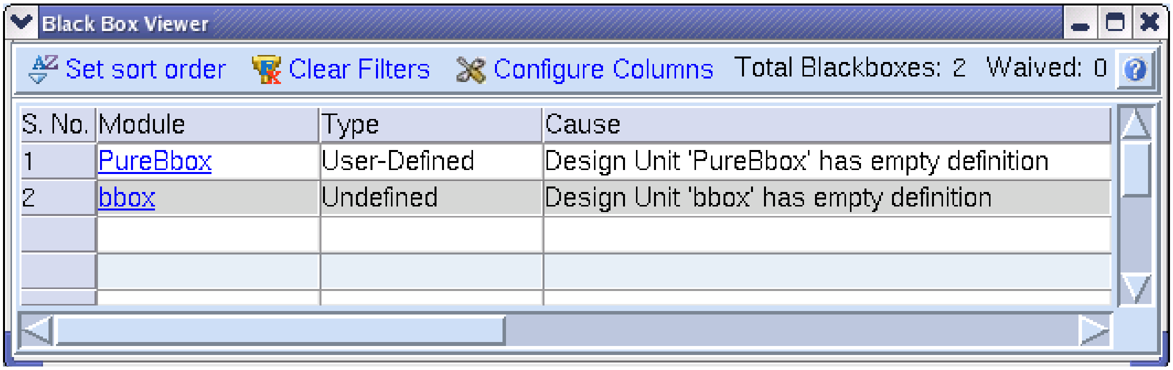
```
nwords = 16 (default: 16)
size   = 8 (default: 8)
C1     = 127 (default: 8'h7F)
C2     = 112 (default: 8'h70)
C3     = 81 (default: 8'h51)
C4     = 37 (default: 8'h25)
C5     = 243 (default: 8'hf3)
C6     = 195 (default: 8'hc3)
C7     = 157 (default: 8'h9d)
C8     = 133 (default: 8'h85)
C9     = 125 (default: 8'h7d)
C10    = 71 (default: 8'h47)
C11    = 231 (default: 8'he7)
C12    = 149 (default: 8'h95)
C13    = 118 (default: 8'h76)
C14    = 207 (default: 8'hcf)
C15    = 90 (default: 8'h5a)
```

Show BlackBox Info

Select this menu option to view the information about all the black boxes in the currently loaded module.

A black box can be identified with the ■ icon.

When you select this option, the *BlackBox Viewer* window appears, as shown in the following figure:



S. No.	Module	Type	Cause
1	PureBbox	User-Defined	Design Unit 'PureBbox' has empty definition
2	bbox	Undefined	Design Unit 'bbox' has empty definition

FIGURE 11. Black Box Viewer

For details on the above window, see *Black Box Viewer Window*.

Copy

Select this menu option to copy the name or path of the selected module.

When you select this option, a sub-menu appears containing two options, *Module Name* and *File Path*.

Waive Rules By Module

Select this menu option to waive rules by the selected module.

When you select this option, the *Waiver Editor Window* appears in which you can specify the required details.

Waive Rules By IP

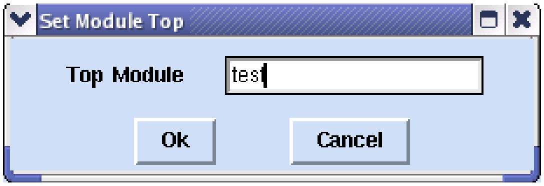
Select this menu option to waive rules by IP.

When you select this option, the *Waiver Editor Window* appears in which you can specify the required details.

Set Top Module

Select this menu option to set the selected module as top module.

When you select this option, the *Set Module Top* dialog appears with the name of the module displayed in the *Top Module field*, as shown in the following figure:



test

FIGURE 12. Set Top Module

Click *Ok* to set the module as top module. The specified module is then prefixed with T icon to indicate it as the top module.

If a top module already exists in the design hierarchy and you try to set another module as the top module, a warning message is displayed to indicate that a top module already exists with a higher/lower priority. Then, you can set the new module as the top module or cancel the action.

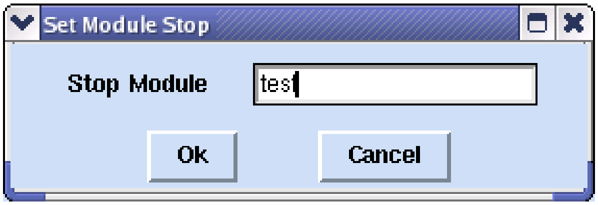
Remove Stop Module

Select this menu option to not consider the selected module as a top module.

Set Stop Module

Select this menu option to set a module as stop module.

When you select this option, the *Set Module Stop* dialog appears with the name of the module displayed in the *Stop Module* field, as shown in the following figure:



test

FIGURE 13. Set Stop Module

Click *Ok* to set the module as stop module. The specified module is the prefixed with the S icon to indicate it as the stop module

Remove Stop Module

Select this menu option to not consider the selected module as a stopped module.

Set Ignore Module

Select this menu option to ignore the selected VHDL design unit or Verilog module for SpyGlass analysis.

Tip : Message Count

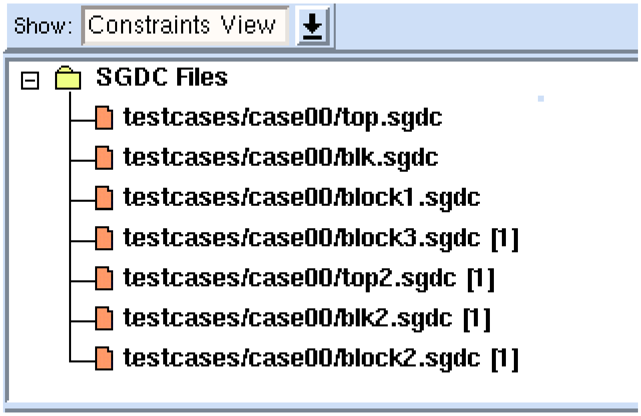
Select this menu option to view the violation message count.

When you select this option, a pop-up window appears displaying the violation count for parent module and its child modules.

Constraints View Page

This page displays the list of constraint files (.sgdc), CPF files (.cpf), and UPF files (.upf) specified in the current SpyGlass run. These files are listed under appropriate categories, that is, *SGDC Files*, *CPF Files*, and *UPF Files* categories.

The following figure shows the *Constraints View Page* listing SGDC files:



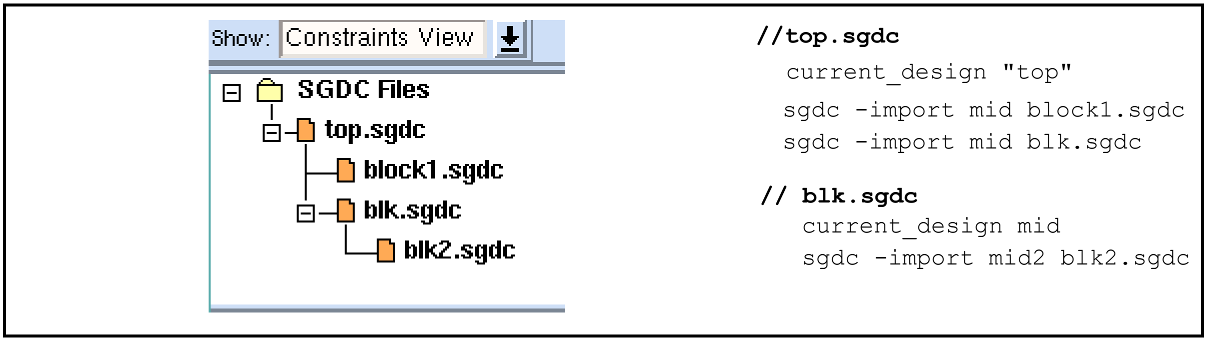
SGDC Files
- testcases/case00/top.sgdc
- testcases/case00/blk.sgdc
- testcases/case00/block1.sgdc
- testcases/case00/block3.sgdc [1]
- testcases/case00/top2.sgdc [1]
- testcases/case00/blk2.sgdc [1]
- testcases/case00/block2.sgdc [1]

FIGURE 14. Constraints View

In the above view, when you double-click on a constraints file name in the *Constraints View* page, the corresponding file appears in the *Source* section with the cursor at the first line of the constraints file.

Viewing the Imported SGDC Files

If an SGDC file contains the `-import` command specifications to import other SGDC files, the *Constraints View Page* shows the SGDC files hierarchically in the manner they are imported. This is explained in the following figure:



```
//top.sgdc
  current_design "top"
  sgdc -import mid block1.sgdc
  sgdc -import mid blk.sgdc

// blk.sgdc
   current_design mid
   sgdc -import mid2 blk2.sgdc
```

FIGURE 15. View Imported Constraints

Viewing the Abstract Block-Level Files

If you specify an abstract block level SGDC file by using the `abstract_file` constraint, the *Constraints View Page* shows that file under the imported SGDC file.

For example, consider the following example:

```
# top.sgdc
current_design top
sgdc -import mid block/spyglass_reports/abstract_view/
block1_cdc_abstract.sgdc

# block1_cdc_abstract.sgdc
abstract_file -version 0.0 -scope cdc -block_file  block/
```

```
spyglass_reports/abstract_view/cdc/clock.sgdc
current_design mid
set_case_analysis -name am -value 1
```

For the above example, the following *Constraints View Page* appears:

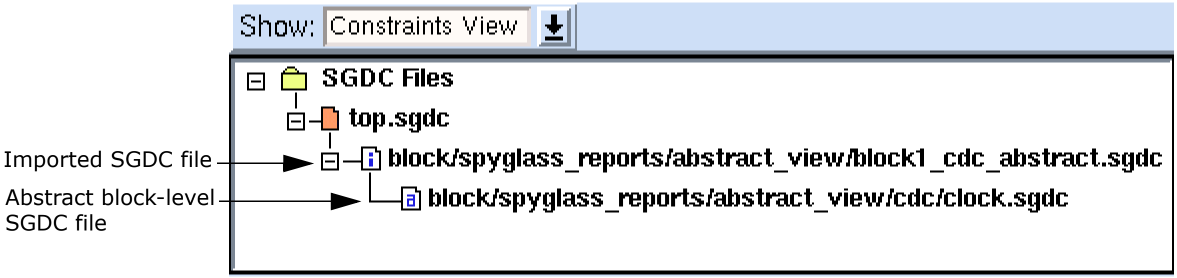

Show: Constraints View
SGDC Files
top.sgdc
block/spyglass_reports/abstract_view/block1_cdc_abstract.sgdc
block/spyglass_reports/abstract_view/cdc/clock.sgdc
Imported SGDC file
Abstract block-level SGDC file

FIGURE 16. Sample Constraints View Page

Right-Click Menu Options of Constraints View Page

When you right-click on a module in the *Constraints View* page, the following options appear in the shortcut menu:

Find File

Select this menu option to search for an SGDC file in the *Constraints View* page.

When you select this option, the *Find* dialog appears in which you can specify the required search criteria.

Edit File

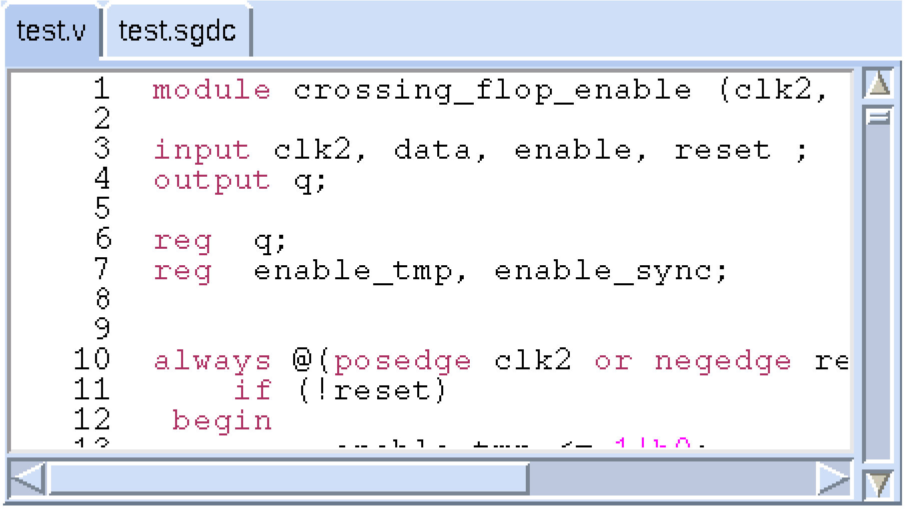
Select this menu option to edit the selected file.

When you select this option, an editor window appears in which you can perform the required updates.

HDL Viewer Pane

The *HDL Viewer* pane displays the selected file, such as source file or SGDC file under separate tabs.

The following figure shows the *HDL Viewer* pane:



```
module crossing_flop_enable (clk2,

input clk2, data, enable, reset ;
output q;

reg  q;
reg  enable_tmp, enable_sync;


always @(posedge clk2 or negedge re
    if (!reset)
 begin
         enable_tmp <= 1'b0;
```

FIGURE 17. HDL Viewer

When you select a particular pathway or gate in the schematic, the corresponding line is highlighted in the source code in the *HDL Viewer* pane.

You can display or hide *HDL Viewer* by clicking the *Show HDL Viewer* or *Hide HDL Viewer* option on the ribbon bar.

Color-Coding Scheme in HDL Viewer

A specific color-coding scheme is used for the source code displayed in the *HDL Viewer* pane. For example, keywords are displayed in the maroon color.

To disable the color scheme in *HDL Viewer*, de-select the *Show Color Coded syntax in RTL Window* option in the *HDL Navigator Pane* page of the

Preferences dialog:

NOTE: *The color-coding is visible only after the design is run. The coding of keywords differentiates between the Verilog and VHDL files and the modes (v2k, no_v2k, and so on) defined for VHDL files.*

Searching in HDL Viewer

To search for any text in the source code, select the *Edit > Search* menu option. When you select this option, the *Search* dialog appears in which you can specify the required search criteria.

Inactive Code Display

An RTL of a design unit can have the following inactive sections:

- ifdef...endif directives in a Verilog module

  These directives make specific sections of the module inactive if the corresponding condition is evaluated to false.
- translate_off/translate_on directives in a design unit

  These directives turn off sections of design on which SpyGlass analysis happens either partially or not at all.
- Comments in the source input files

Such inactive sections are highlighted in the green color in the *HDL Viewer* pane.

NOTE: *The inactive code display feature is not available for the precompiled VHDL library files.*

Interaction with Other Windows

When you double-click on an object in a source design file, that object is highlighted in various windows.

The following table specifies different types of objects that is highlighted in different windows:

Object Type	Window Actions	Highlight Details
Nets	Modular Schematic window is redrawn to show the design unit containing the probed net.	Net is highlighted.
	Net and all connected components are added to the Incremental Schematic window.	Net is highlighted across the hierarchical boundaries.
	Net is added to the Legend window list.	-
Ports	Modular Schematic window is redrawn to show the module containing the probed port.	The Port and its connected net are highlighted.
	Port and all components connected to the port’s connected net are added to the Incremental Schematic window.	Port and its connected net are highlighted across the hierarchical boundaries.
	Port is added to the Legend window list.	-
Instances	Modular Schematic window is redrawn to show the parent module containing the probed instance.	Instance is highlighted.
	Instance icon is added to the Incremental Schematic window.	Instance is highlighted
	Instance is added to the Legend window list.	-

Cross-probing of Nets To and From Schematic

For a current module, a net and all its end connections up to ports or leaf-level instances are highlighted in *The Modular Schematic Window*.

To view the schematic pathways or gates associated with a given object, open the *Modular Schematic* window and double-click the object in the *HDL Viewer*. Similarly, double-click an object in the *Modular Schematic* window and the corresponding file is opened in the *HDL Viewer*, if not already open, and the name is highlighted.

NOTE: *You can also select more than one objects for probing or remove an object from a set of probed objects in the schematic windows by clicking the relevant object while* *holding down the <Ctrl> key on the keyboard.*

SGDC to Schematic Cross-Probing

To implement SGDC to schematic cross-probing, double-click on a SGDC file appearing in the *Constraints View Page*. This displays the SGDC source file in the *HDL Viewer*.

Objects in this source file appear as hyperlinks. So when you select an object in the *HDL Viewer* and click the *Modular Schematic* or *Incremental Schematic* icons on the toolbar, the corresponding object is highlighted in the schematic window.

Limiting the Number of Objects While Cross-Probing From SGDC

Cross-probing from an SGDC file is limited to a maximum of 30 matches for a single object in the SGDC file. Out of these, only 10 matches are shown by default.

If you select an object that has more than 10 corresponding objects, the *Select Limit* dialog appears that prompts you to enter the number of objects that need to be cross-probed. In this dialog, select the *Display objects* option and enter the number of objects that you want to cross-probe in the text field. Select the *Display All objects* option to consider all objects for cross-probing.

You can also enter the number of objects that need to be cross-probed by selecting the *Select maximum number of objects that can be cross-probed from SGDC file* option from the *Misc Page* of the *Tools > Preferences* window.

NOTE: *Selecting a large number of objects for cross-probing may require more memory.*

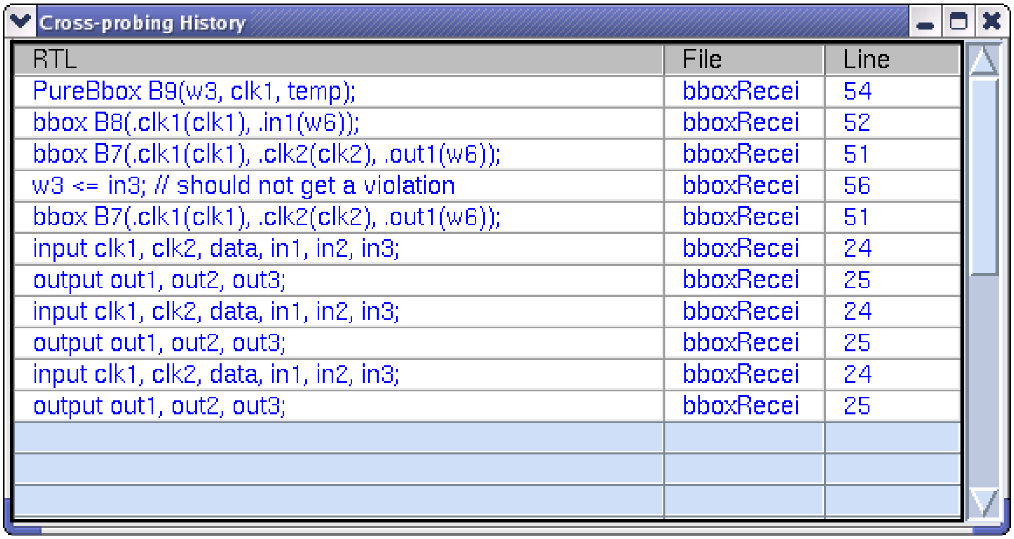
You can also view the SGDC constraints set on an object. Refer to the *Viewing SDC and SGDC Constraints Set on an Object* section for details.

Viewing Cross-Probing History

In SpyGlass, cross-probing can be performed from various sections, such as *HDL Viewer Pane*, *The File/Module/Instance/Constraints Pane*, *HDL Navigator Pane*, *The Modular Schematic Window*, *The Incremental Schematic Window*, *Spreadsheet Viewer Window*, *Waveform Viewer Window*, and various SpyGlass reports.

Atrenta Console provides the following buttons in the *HDL Viewer* that enable you to keep a track of highlighted lines for better debugging:

- Previous button (↶): This button is enabled if highlighting information is available in history. When you click this button, the previously highlighted file and line is fetched from the history and highlighted in the *HDL Viewer*. You can also view the complete history by right-clicking this button and selecting the *All History* option from the shortcut menu. Then, the *Cross-probing History* window appears that provides a list of all highlighted lines.



RTL	File	Line
PureBbox B9(w3, clk1, temp);	bboxRecei	54
bbox B8(.clk1(clk1), .in1(w6));	bboxRecei	52
bbox B7(.clk1(clk1), .clk2(clk2), .out1(w6));	bboxRecei	51
w3 <= in3; // should not get a violation	bboxRecei	56
bbox B7(.clk1(clk1), .clk2(clk2), .out1(w6));	bboxRecei	51
input clk1, clk2, data, in1, in2, in3;	bboxRecei	24
output out1, out2, out3;	bboxRecei	25
input clk1, clk2, data, in1, in2, in3;	bboxRecei	24
output out1, out2, out3;	bboxRecei	25
input clk1, clk2, data, in1, in2, in3;	bboxRecei	24
output out1, out2, out3;	bboxRecei	25

FIGURE 18. Cross-probing History

NOTE: *If you click the back button several times and then cross-probe from different sections, the new highlighted information is inserted in the current position of the history (instead of at the end). Therefore, some of the highlighted history might be lost.*

- Next button (↷): This button is enabled when a previously highlighted line is fetched from the history using the Previous button. You can also view the complete history by right-clicking the Next button and selecting the *All History...* option from the shortcut menu. Then, the *Cross-probing History* window appears that provides a list of all highlighted lines.

The Previous and Next buttons are hidden by default. To display these buttons, right-click on the navigation bar and select the *Next/Previous Probed Line* check box.

Right-Click Options in the HDL Viewer Pane

When you right-click on a signal in the *HDL Viewer* pane, the following options appear in the shortcut menu:

- *Common Options*
- *Options Visible at the Analyze Results Stage*

Common Options

Following are the common set of options that appear when you right-click on the *HDL Viewer* pane:

Edit File

Use this option to edit a file.

When you select this option, a text-editing program appear as defined by the *Specify external editor program* setting in the *Miscellaneous Page* of the *Tools > Preferences* menu option.

If you have not specified a text editor in this setting, the text editor pointed to by the EDITOR environment variable is invoked. Otherwise, Atrenta Console attempts to invoke the commonly-used text editor of your operating platform.

If you edit and save a source file listed in the *File View* page, the file name is displayed in the red color to indicate that the file has been modified since last analysis run.

Module > Show Instance List

Use this option to view the list of instances in the module. When you select the Show Instance List option, the Instances window is displayed. Refer to the *Show Instance List* section for more details.

Module > Set as Top Module

Use this option to set a module as a top module for the next run. When you select this option, the specified module is prefixed with a (T) in the Module View page. Refer to *Set Top Module* for details.

Module > Remove Top Module

Use this option to remove a module as a top module for the next run.

Module > Stop Module

Use this option to stop the module from being synthesized in the next run. When you select this option, the specified module is prefixed with a (s) in the Module View page. Refer to *Set Stop Module* for details.

Module > Remove Stop Module

Use this option so that a stopped module is synthesized in the next run.

Waive > Waive Messages of Line

Use this option to waive the messages of the selected line. When you select this option, the Waiver Editor window is displayed with an entry for the selected line. See *Tools > Waiver Editor* for more details.

Click *OK* to apply the setting and close the dialog.

Waive Rules by Module

Use this option to waive all messages corresponding to the module.

Preferences

Use this option to set the preferences, such as view color-coded syntax in the HDL Viewer and view detailed information about the drivers and loads declared for a signal. When you click this menu option, the Preferences window opens at the *HDL Navigator Page*.

Options Visible at the Analyze Results Stage

The following additional options are visible only at the *Analyze Results* stage.

Help

Use this option to view the HTML help for the selected violation in the message window. When you select this option, a browser window appears that displays information about the rule.

Signal

Use this option to view the information about a signal, such as the location where the signal is declared and the loads and drivers associated with the signal.

Instance

Use this option to view the information about an instance, such as, the scope of the instance and the location where the instance is declared.

Macro

Use this option to view the information about a macro, such as, the value of the macro and the location where the macro is defined.

Copy Object Name

Use this option to copy the current name of the module.

Properties

Use this option to view the signal properties. When you click the *Properties...* option, the Properties window is displayed that lists the signal name, the signal type, and the full hierarchical name of the signal.

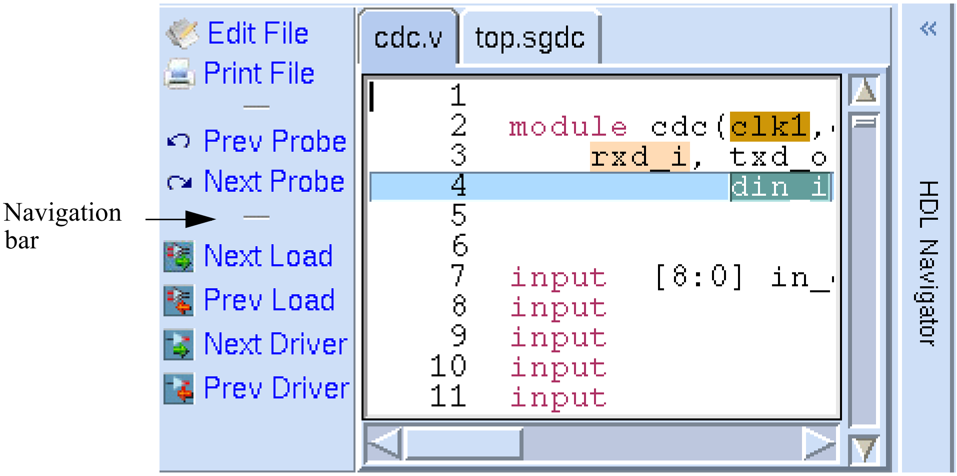
Set SGDC Constraints

Use this option to set the SGDC constraints on a signal. For more information on setting SGDC constraints, refer to the *Setting SGDC Constraints* section.

Navigation Bar

A navigation bar is used to edit and print an RTL file, probe between signals, and navigate between loads and drivers declared for a signal.

It is present towards the left side of the *HDL Viewer* pane. The following figure shows the navigation bar:



```
 1
 2  module cdc(clk1,
 3       rxd_i, txd_o
 4                din_i
 5
 6
 7  input   [8:0] in_
 8  input
 9  input
10  input
11  input
```

FIGURE 19. Navigation Bar

To navigate between the loads and drivers declared for a signal, double-click a signal to select it. Next, select from any of the following options:

- *Edit File*

  Click this option to open the file in a text editor specified by you in the *Miscellaneous Page* of the Preferences window.

- *Print File*

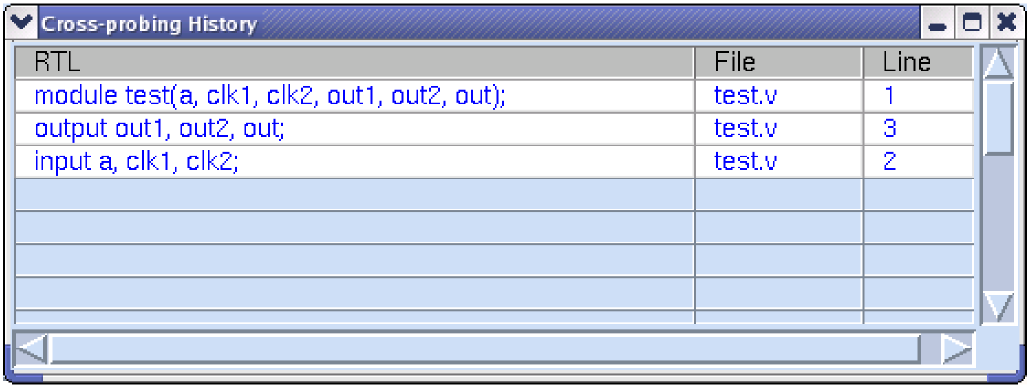
  Click this option to print the displayed RTL file. See *File > Print File* for more details.

- *Prev Probe*

  Click this option to navigate to the previous cross-probed line in the HDL Viewer.

  You can view the complete cross-probing history by right-clicking the

Prev Probe link and selecting the *All History...* shortcut menu option. Then, the Cross-probing History dialog appears as shown below.



Cross-probing History

RTL	File	Line
module test(a, clk1, clk2, out1, out2, out);	test.v	1
output out1, out2, out;	test.v	3
input a, clk1, clk2;	test.v	2

FIGURE 20. Cross-probing History

Clicking a link on the Cross-probing History dialog highlights the line in the HDL Viewer.

- *Next Probe*

  Click this option to navigate to the next cross-probed line in the HDL Viewer.

- *Next Load*

  Click this option to navigate to the next load declared for a signal.

- *Previous Load*

  Click this option to navigate to the previous load declared for a signal.

- *Next Driver*

  Click this option to navigate to the next driver declared for a signal.

- *Previous Driver*

  Click this option to navigate to the previous driver declared for a signal.

HDL Navigator Pane

When you double-click on a signal, instance, or a macro in the HDL Viewer pane, the HDL Navigation pane displays the following:

- Information for Signals
- Information for Instances
- Information for Macros

Information for Signals

The *HDL Navigator* pane displays information about loads and drivers declared for a signal, which is double-clicked in the *HDL Viewer Pane*.

The following figure shows the *HDL Navigator* pane:

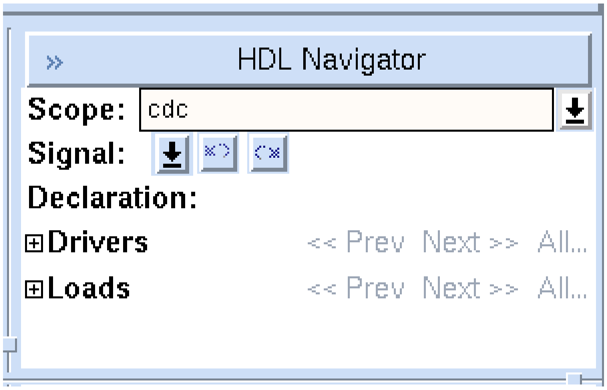


FIGURE 21. HDL Navigator

This pane appears only at the *Analyze Setup* stage.

The HDL Navigator pane displays the following information:

Scope	*Signal*	*Declaration*	*Drivers*	*Loads*

Scope

A signal may span across a design hierarchy, and therefore, may have different loads and drivers declared at different levels. You can set the scope for a signal in the *Scope* field.

By default, SpyGlass selects the most relevant scope for you. You can edit this scope depending on your requirement. If a signal has only one scope, the *Scope* field is updated automatically. If the signal has multiple scopes, you can select the relevant scope from the *Scope* drop-down list.

You can set the scope from the *Instance* view or by selecting a violation message.

Signal

This section contains a list of the signals selected in the *Source* section. You can select a signal from the drop-down list to view the drivers and loads declared for that signal.

Declaration

This section contains the file name and line number of an RTL source code where the signal is declared.

By default, the *Declaration* option is not visible in the *HDL Navigator* pane. To display this option in this pane, select the *Display > Signal Declaration* option from the *HDL Navigator Pane* of the *Preferences* dialog.

You can also invoke the *Preferences* dialog by selecting the *Preferences* option from the *Options* drop-down list.

Drivers

This section contains the drivers declared for a signal selected in the *Signal* section.

When you click on a driver, the line where the driver is declared is highlighted in the *Source* section. To know the line and line where a driver is declared in the source code, place the cursor over that driver. A balloon appears displaying the line and file information. Click the *Next* button to view the next driver location in the *Source* section.

To view all the drivers declared for a signal, click *All*. When you click All, the SpyGlass Drivers window opens that displays the list of all drivers declared for the signal. You can click on a driver in the SpyGlass Drivers window to view the location where the driver is declared in the HDL Viewer.

You can also view the driver information in a tabular format by selecting the *Show Detailed View* option from the *HDL Navigator Page* of the Preferences window.

You can show or hide the *Drivers* section in the HDL Navigator window by selecting the *Display > Drivers* option from the *HDL Navigator Page* of the Preferences window.

Loads

This section contains the loads declared for the signal selected in the *Signal* section. When you click a load in this section, the line where the load is declared is highlighted in the HDL Viewer. You can also find the file and line number of the RTL source code where the load is declared by placing the cursor over the load. Then, the file name and line number of the RTL source is displayed in a balloon window.

Click the *Next* button to view the next load location in the HDL Viewer.

To view all the loads declared for the signal click *All*. When you click All, the SpyGlass Loads window opens that displays the list of all loads declared for the signal. You can click on a load in the SpyGlass Loads window to view the location where the load is declared in the HDL Viewer.

You can also view the load information in a tabular format by selecting the *Show Detailed View* option from the *HDL Navigator Page* of the Preferences window.

You can show or hide the *Loads* section in the HDL Navigator window by selecting the *Display > Loads* option from the *HDL Navigator Page* of the Preferences window.

Properties

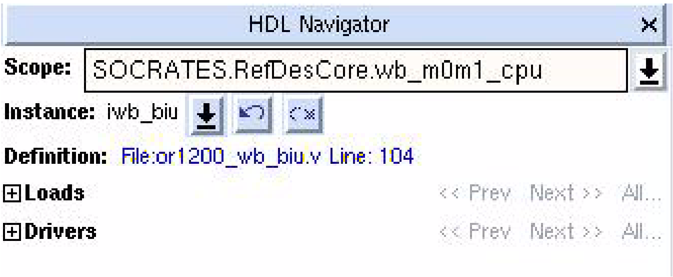
This section contains information about the signal, such as the signal name, signal type, and the full hierarchical name of the signal.

You can show or hide the *Properties* section in the HDL Navigator window by selecting the *Display > Properties* option from the *HDL Navigator Page* of the Preferences window.

Information for Instances

The *HDL Navigator* pane displays information about the instance, which is double-clicked in the *HDL Viewer Pane*.

The following figure shows the *HDL Navigator* pane:



HDL Navigator

Scope: SOCRATES.RefDesCore.wb_m0m1_cpu

Instance: iwb_biu

Definition: File:or1200_wb_biu.v Line: 104

FIGURE 22. HDL Navigator - Instance

The HDL Navigator pane displays the following information for Instances:

Scope *Instance* *Definition* *Drivers* *Loads*

Scope

An instance may span across a design hierarchy, and therefore, may have different loads and drivers declared at different levels. You can set the scope for a signal in the *Scope* field.

By default, SpyGlass selects the most relevant scope for you. You can edit this scope depending on your requirement. If an instance has only one scope, the *Scope* field is updated automatically. If an instance has multiple scopes, you can select the relevant scope from the *Scope* drop-down list.

You can set the scope from the *Instance* view or by selecting a violation message.

Instance

This section contains a list of instances selected in the *Source* section. You can select an instance from the drop-down list to view the drivers and loads declared for that instance.

Definition

This section contains the file name and line number of an RTL source code where the instance is declared.

By default, the *Definition* option is not visible in the *HDL Navigator* pane. To display this option in this pane, select the *Display > Signal Declaration* option from the *HDL Navigator Pane* of the *Preferences* dialog.

You can also invoke the *Preferences* dialog by selecting the *Preferences* option from the *Options* drop-down list.

Drivers

This section contains the drivers declared for an instance selected in the *Signal* section.

When you click on a driver, the line where the driver is declared is highlighted in the *Source* section. To know the line and line where a driver is declared in the source code, place the cursor over that driver. A balloon appears displaying the line and file information. Click the *Next* button to view the next driver location in the *Source* section.

To view all the drivers declared for an instance, click *All*. When you click All, the SpyGlass Drivers window opens that displays the list of all drivers declared for the instance. You can click on a driver in the SpyGlass Drivers window to view the location where the driver is declared in the HDL Viewer.

You can also view the driver information in a tabular format by selecting the *Show Detailed View* option from the *HDL Navigator Page* of the Preferences window.

You can show or hide the *Drivers* section in the HDL Navigator window by selecting the *Display > Drivers* option from the *HDL Navigator Page* of the Preferences window.

Loads

This section contains the loads declared for the instance selected in the *Instance* section. When you click a load in this section, the line where the load is declared is highlighted in the HDL Viewer. You can also find the file and line number of the RTL source code where the load is declared by placing the cursor over the load. Then, the file name and line number of the RTL source is displayed in a balloon window.

Click the *Next* button to view the next load location in the HDL Viewer.

To view all the loads declared for the instance click *All*. When you click All, the SpyGlass Loads window opens that displays the list of all loads declared for the instance. You can click on a load in the SpyGlass Loads window to view the location where the load is declared in the HDL Viewer.

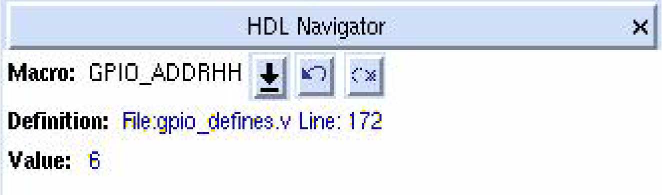
You can also view the load information in a tabular format by selecting the *Show Detailed View* option from the *HDL Navigator Page* of the Preferences window.

You can show or hide the *Loads* section in the HDL Navigator window by selecting the *Display > Loads* option from the *HDL Navigator Page* of the Preferences window.

Information for Macros

The *HDL Navigator* pane displays information about the macro, which is double-clicked in the *HDL Viewer Pane*.

The following figure shows the *HDL Navigator* pane:



Macro: GPIO_ADDRHH

Definition: File:gpio_defines.v Line: 172

Value: 6

FIGURE 23. HDL Navigator Pane - Macros

The HDL Navigator pane displays the following information for Macros:

Macro	*Definition*	*Value*

Macro

This section contains a list of macros selected in the *Source* section. You can select a macro from the drop-down list to view the corresponding definition and value for the selected macro.

Definition

This section contains the file name and line number of an RTL source code where the macro is declared. The information is displayed as a clickable link. Clicking on the link takes you to the respective macro definition in the source.

Value

This section displays the value for the selected macro.

Viewing Declarations of Signals as Loads/Drivers

By default, the search for loads and drivers stops at a module boundary. If the *Stop At Module Boundary* option is selected from the *Options* drop-down list, loads and drivers stop at the location where the signal is declared at the start of the module. However, if you deselect the *Stop At Module Boundary* option, loads and drivers also show usage of a signal outside/inside the module boundary.

The following may occur when the *Stop at Module Boundary* option is selected:

- Clicking the signals in instantiations will display the loads or drivers inside the module being instantiated if the signal is an input or output signal respective to that module (declaration line of the signals/ports).
- Clicking the input port/signal declarations in a module will display the drivers outside the module (location where that module is being instantiated).
- Clicking the output port/signal declarations in a module will display the loads outside the module.
- Clicking any signal that is an input to its module will display the drivers as the port declaration of that signal.
- Clicking any signal that is an output of its module will display the loads as the port declaration of that signal.

The following may occur when the *Stop at Module Boundary* option is not selected:

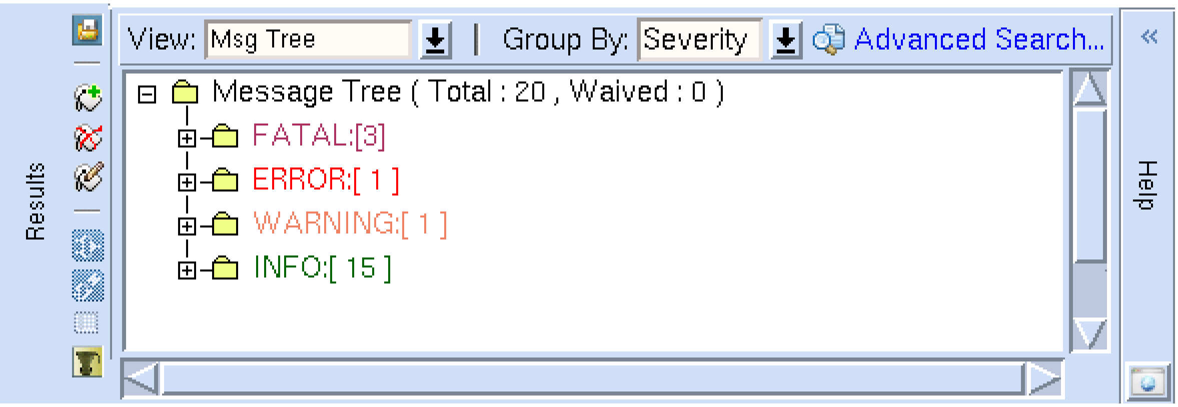
- Clicking the signals in instantiations will display the loads or drivers inside the module being instantiated if the signal is an input or output signal respective to that module (usage of the signal within the module being instantiated).
- Clicking the input port/signal declarations in a module will display the drivers outside the module (location where that module is being instantiated).
- Clicking the output port/signal declarations in a module will display the loads outside the module.
- Clicking any signal that is an input of its module will display the drivers outside the module (location where that module is being instantiated).

- Clicking any signal that is an output of its module will display the loads outside the module.

NOTE: *Clicking the input/output port/signal declarations in a module will always display the loads/drivers outside the module irrespective of whether the Stop at Module Boundary option is selected or deselected.*

Results Pane

The *Results* pane displays violation messages generated after SpyGlass analysis. The following figure displays the *Results* pane:



View: Msg Tree | Group By: Severity Advanced Search...

```
⊟ Message Tree ( Total : 20 , Waived : 0 )
   ⊞ FATAL:[3]
   ⊞ ERROR:[ 1 ]
   ⊞ WARNING:[ 1 ]
   ⊞ INFO:[ 15 ]
```

FIGURE 24. Results Pane

When you double-click on a violation message in the *Results* pane, the corresponding design unit is displayed in the *HDL Viewer Pane* and the complete message path is highlighted in *The Modular Schematic Window*. For other rule messages, only the containing design unit is displayed.

Atrenta Console displays a specific symbol prefixed with some violation messages. The following table describes the meaning of each symbol:

Symbol	Description
⊃	Indicates that the message has a schematic associated with it
T	Indicates that selected rule messages and rule descriptions have associated text data or are multi-line messages
∿	Indicates that the message has a waveform associated with it
G	Indicates that the message has an FSM associated with it

When you place the pointer over a message, a small balloon window appears that displays the following information:

- Tag applied to the message
- Goal that contains the rule corresponding to the message
- Severity of the message
- Rule name corresponding to the message.

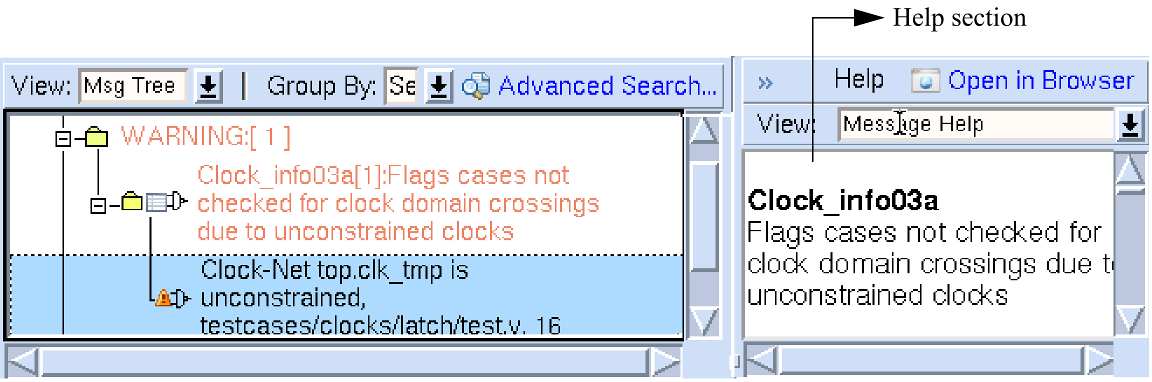
The *Results* pane contains the following drop-down lists:

- *View*: Used to select the required format, such as *Msg Tree Page*, *Msg Summary Page*, *Module Hierarchy Page*, and *Waiver Tree Page* for displaying messages.
- *Group By*: Used to select a value based on which the messages are grouped.

Message Help Section

The *Help* section in the *Results* pane displays different type of help information, such as message help and goal debug help.

The following figure displays the *Help* section:



View: Msg Tree

Group By: Se

WARNING:[1]

Clock_info03a[1]:Flags cases not checked for clock domain crossings due to unconstrained clocks

Clock-Net top.clk_tmp is unconstrained, testcases/clocks/latch/test.v, 16

View: Message Help

Clock_info03a

Flags cases not checked for clock domain crossings due t unconstrained clocks

FIGURE 25. Message Help

Depending on the type of help you want to view, select *Message Help* or *Goal Debug Help* option from the *View* drop-down list. If nothing is selected in the message tree, the *Help* section displays a *What's Next* message that provides you some basic guidance to start debugging.

Goal Debug Help

Goal debug help provides you guidance and hints to debug the messages reported after goal run.

To view the goal debug help, select the *Goal Debug Help* option from the *View* drop-down list in the *Help* window.

Message Help

Message help provides you details of the corresponding rule of selected message.

To view the message help, select the violation message and then select the *Message Help* option from the *View* drop-down list in the *Help* window.

If you want to view the rule help in a separate browser window, select the *Open in Browser* option.

Message Grouping

You can group messages in different grouping orders by selecting an appropriate option from the *Group By* drop-down list.

Viewing the Grouping Level

The text displayed in the *Group By* text-box specifies the first level of the currently selected grouping order. To view all levels of the currently selected grouping order, place the cursor over the *Group By* text-box. A tool-tip appears that specifies all levels of the currently selected grouping order. Similarly, you can view all the levels of the grouping orders listed in the *Group By* pull-down list by placing the cursor over the grouping orders listed therein.

Specifying the Default Grouping Order

You can define a default grouping order for rule messages in the .spyglass.grouping_order file which is automatically created and stored in the $HOME/.atrenta directory when you select the grouping order in Atrenta Console.

You can also store the .spyglass.grouping_order file at the following locations:

- In addition, you can store the .spyglass.grouping_order file in the SpyGlass installation directory ($SPYGLASS_HOME). Storing the .spyglass.grouping_order file in the SpyGlass installation directory

($SPYGLASS_HOME) is useful when multiple SpyGlass users are working on the same project/design and consistent usage of message grouping order is required. However, the settings specified in the $HOME/.atrenta directory override the settings specified in the SpyGlass installation directory ($SPYGLASS_HOME). Therefore, if you require consistent usage of message grouping, ensure that you do not change the grouping order.

- Any directory other than $SPYGLASS_HOME and $HOME/.atrenta. You can save the .spyglass.grouping_order file in any directory of your choice, and set the SPYGLASS_CONFIG_PATH environment variable to point to that directory. Then, the settings specified in the directory override the settings in the SpyGlass installation directory ($SPYGLASS_HOME). However, the settings specified in the $HOME/ .atrenta directory override the settings specified in the directory that is specified using the SPYGLASS_CONFIG_PATH environment variable.

The .spyglass.grouping_order file is automatically read whenever SpyGlass starts and the message grouping order specified in this file is automatically loaded.

NOTE: *Since the .spyglass.grouping_order file is automatically read and loaded whenever SpyGlass starts. Therefore, there is no need to explicitly specify this file.*

Sample .spyglass.grouping_order file

```
Severity#Severity->Rule->Message(s)
Goal# Goal->Severity->Rule->Message(s)
Tag#UserTag->Severity->Rule->Message(s)
All#Built-In->UserTag->Goal
    ->RuleGroup->Rule->Message(s)
```

Viewing the Default Grouping

In case no default message grouping order is specified (that is, the .spyglass.grouping_order file is neither present in the $HOME/.atrenta directory nor in the $SPYGLASS_HOME directory), the following SpyGlass default message grouping orders are loaded:

- Severity

  Severity->Rule->Message(s)

- Goal

  Goal->Severity->Rule->Message(s)

- Tag

  User Tag->Severity->Rule->Message(s)
- File

  File->Severity->Rule->Message(s)
- Module

  Module->Severity->Rule->Message(s)
- Severity-1

  Severity-Label->Rule->Message(s)
- Built-in

  Built-In->Severity-Class->Rule->Message(s)
- Rules (Goal Order)

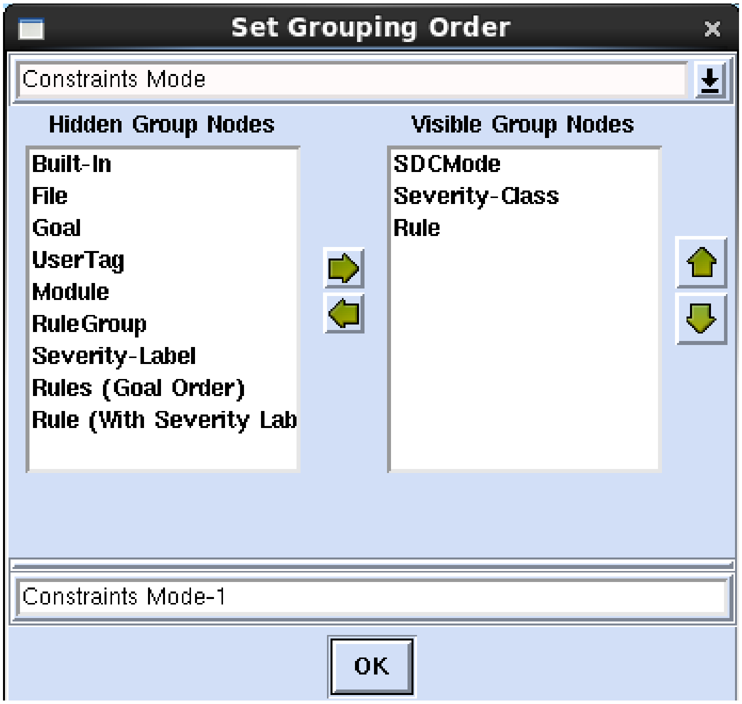
  Goal->Rules (Goal Order)->Message(s)

  In this case, rules are listed in the order in which they are specified in the goal file of a goal.
- SDCMode

  Severity->Rule->Message(s)

Customizing the Grouping Order

In addition, you can also set the message grouping order according to your preference. To do so, select the *Custom...* option from the *Group By* drop-down list in the *Msg Tree* toolbar. Then, the Set Grouping Order dialog box is displayed, as shown below:



Set Grouping Order

Constraints Mode

Hidden Group Nodes
- Built-In
- File
- Goal
- UserTag
- Module
- RuleGroup
- Severity-Label
- Rules (Goal Order)
- Rule (With Severity Lab

Visible Group Nodes
- SDCMode
- Severity-Class
- Rule

Constraints Mode-1

FIGURE 26. Set Grouping Order

In the **Set Grouping Order** dialog box, you can select the grouping criteria and create your own custom grouping. You can also change the order of the nodes, using the Up ⬆ and Down ⬇ arrows.

The order in which the grouping criteria is placed in the Visible Group Nodes column is the order in which the message tree is organized.

To save the current settings specified for the grouping order of the rule messages in the Msg Tree page for later use, specify a name for the grouping and click OK.

The Msg Tree page is updated according to the new settings.

Message Display Formats

In the *Results* pane, you can select various message display formats, such as *Msg Tree Page*, *Msg Summary Page*, *Module Hierarchy Page*, and *Waiver Tree Page*.

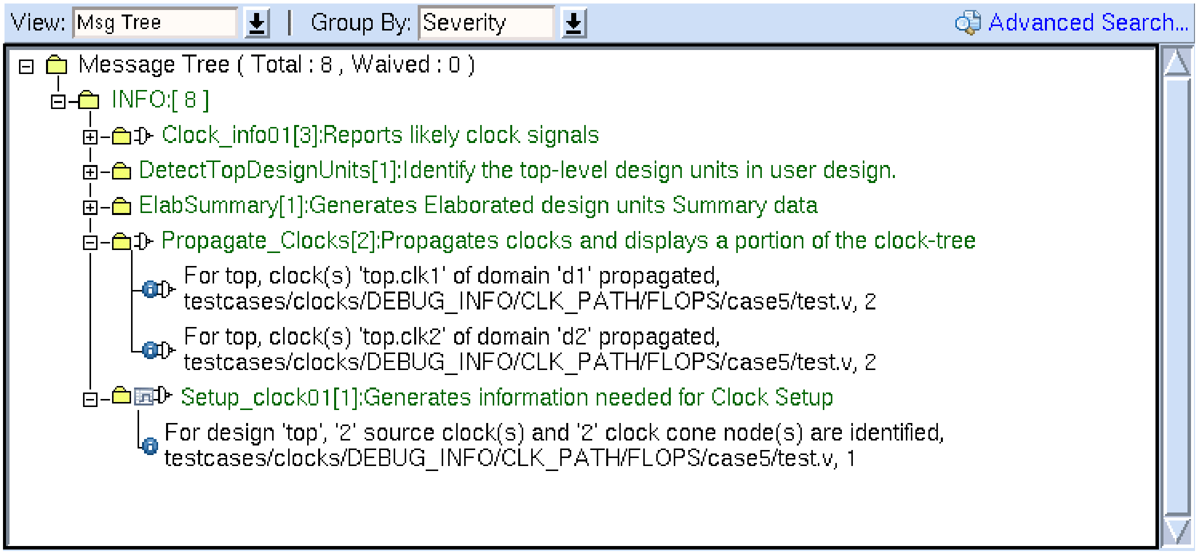
Select the required format from the *View:* drop-down list.

Msg Tree Page

The *Msg Tree* page displays:

- A hierarchical message tree sorted by a message severity (by default) or by a product.
- The count of total number of messages (waived + non-waived) and total number of waived messages.

The following figure displays the *Msg Tree* page:



```
Message Tree ( Total : 8 , Waived : 0 )
  INFO:[ 8 ]
    Clock_info01[3]:Reports likely clock signals
    DetectTopDesignUnits[1]:Identify the top-level design units in user design.
    ElabSummary[1]:Generates Elaborated design units Summary data
    Propagate_Clocks[2]:Propagates clocks and displays a portion of the clock-tree
      For top, clock(s) 'top.clk1' of domain 'd1' propagated,
      testcases/clocks/DEBUG_INFO/CLK_PATH/FLOPS/case5/test.v, 2
      For top, clock(s) 'top.clk2' of domain 'd2' propagated,
      testcases/clocks/DEBUG_INFO/CLK_PATH/FLOPS/case5/test.v, 2
    Setup_clock01[1]:Generates information needed for Clock Setup
      For design 'top', '2' source clock(s) and '2' clock cone node(s) are identified,
      testcases/clocks/DEBUG_INFO/CLK_PATH/FLOPS/case5/test.v, 1
```

FIGURE 27. Msg Tree Page

The *Msg Tree* page displays a hierarchical list of violation messages based on the goals executed. This list displays fatal messages at the top of the hierarchy followed by the messages of other severities.

You can collapse the expanded message tree appearing in the *Msg Tree* page by using the shortcut key, Ctrl -.

Grouping in Msg Tree Page

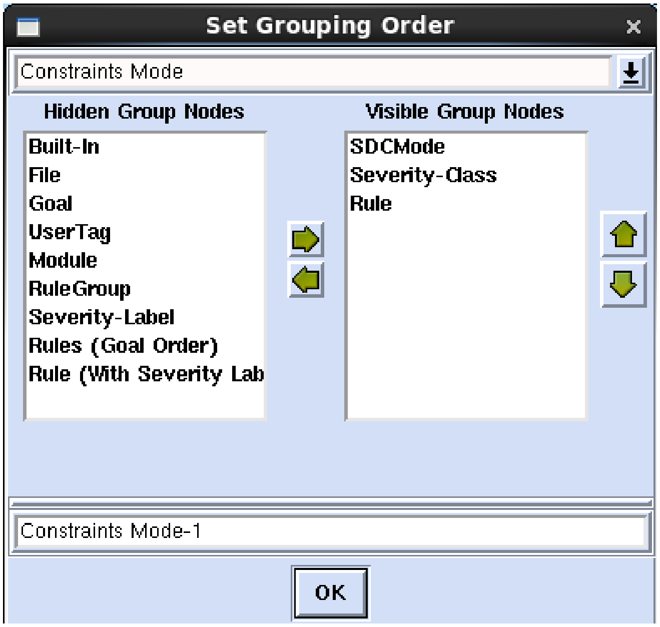
You can group the messages under different categories by selecting an appropriate option from the *Group By* drop-down list. Based on the type of grouping selected, messages are sorted accordingly as discussed in the following points:

- If messages are grouped by product, the messages are sorted in the following order:
  - ❒ Sorted by product
  - ❒ Sorted by alphabetically arranged rules
  - ❒ Sorted on the basis of placement in source files

  To view messages based on their severity under the product node, perform the following steps:

1. Select the *Custom* option from the *Group-By* drop-down list.

   The Set Grouping Order dialog box is displayed as shown in the following figure:



Constraints Mode

Hidden Group Nodes: Built-In, File, Goal, UserTag, Module, RuleGroup, Severity-Label, Rules (Goal Order), Rule (With Severity Lab

Visible Group Nodes: SDCMode, Severity-Class, Rule

Constraints Mode-1

2. Select any of the predefined options from the Set Grouping Order dialog box.
3. Specify a name for the grouping Order.
4. Click the *OK* button.

   If messages are sorted based on severity, the violation messages of rules having multiple severities are grouped under different severity labels.

   If messages are not grouped by severity, the severity of the messages can be identified based on the following severity icons:

Message Severity	Severity Icon
FATAL	●
ERROR	⬢
WARNING	⚠
INFO	ℹ

Color Scheme in the Msg Tree Page

Messages in the *Msg Tree* page are displayed according to the following default color scheme:

Message Severity	Display Color
FATAL	Dark Red
ERROR	Red
WARNING	Amber
INFO	Green

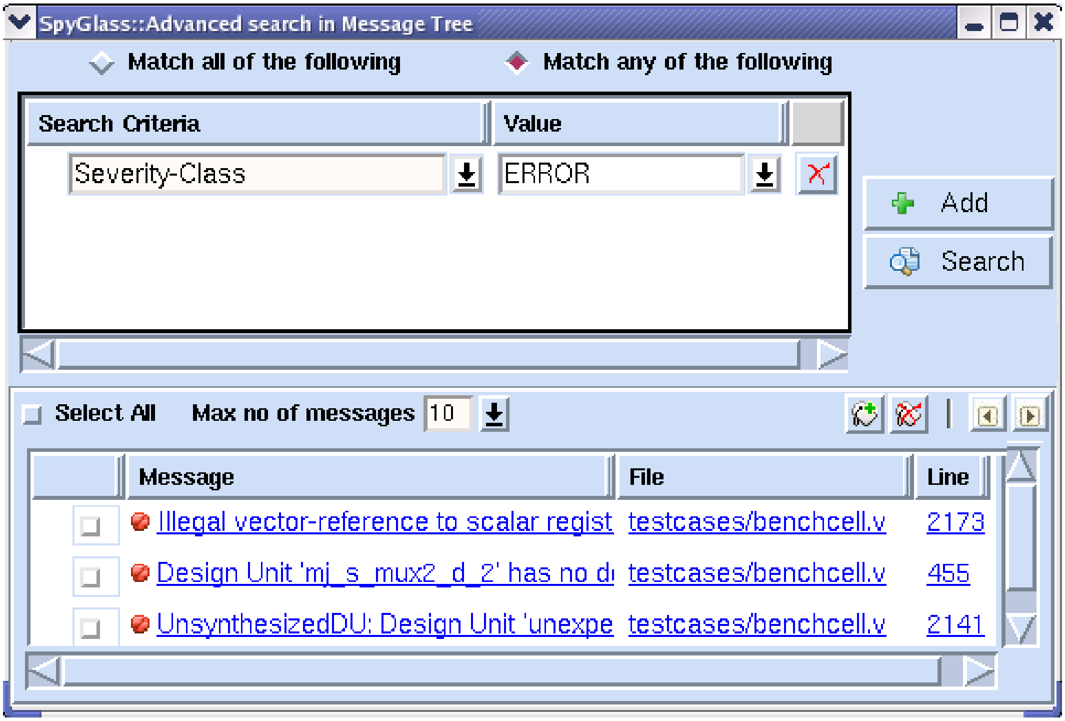
You can customize the above color scheme according to your preference by using the *Message Page* of the *Tools > Preferences* menu option. Alternatively, you can right-click on the message and select the *Preferences* option from the shortcut menu to open the *Preferences* dialog.

Advanced Search in the Msg Tree Page

You can perform an advanced search in the *Msg Tree* page to search for a specific text.

To perform an advanced search, click Advanced Search... in the *Msg Tree* page. This displays the *Advanced Search* dialog.

The following figure shows the *Advanced Search* dialog:


SpyGlass::Advanced search in Message Tree
◇ Match all of the following
◆ Match any of the following
Add
Search
□ Select All
Max no of messages 10

Search Criteria	Value
Severity-Class	ERROR

	Message	File	Line
□	Illegal vector-reference to scalar regist	testcases/benchcell.v	2173
□	Design Unit 'mj_s_mux2_d_2' has no d	testcases/benchcell.v	455
□	UnsynthesizedDU: Design Unit 'unexpe	testcases/benchcell.v	2141

FIGURE 28. Message Tree - Advanced Search

The above dialog is divided into the following two sections:

- The top-most section enables you to specify the search criteria.
- The bottom section displays results based on the specified search criteria.

Specifying the Search Criteria

Select the type of search criteria from the drop-down list in the *Search Criteria* field. Based on the selected search criteria, the *Value* drop-down list is populated with all the available values for the selected search criteria. You can then select an appropriate value from the *Value* drop-down list. For example, if you select the search criteria as *Severity-Class*, you can select the required severity class, such as *ERROR*, *WARNING*, *INFO*, etc., from the *Value* field.

If you want to specify multiple search criteria, click the *Add* button. This adds a new row in which you can specify another search criteria.

You can choose to search for the reported violation messages based on all the specified search criteria (similar to the AND operation) or any one of the specified search criteria (similar to the OR operation). To do so, select the *Match all of the following* or *Select any of the following* options, respectively.

Once you have specified the required details, click the *Search* button to view the list of violation messages matching the specified criteria.

Viewing Results

In the results section of the advance search window, each row represents the information about the found messages. This information includes the individual message text, the file name, and the line number of the file where the message is reported, and the unique hexadecimal violation id associated with each message.

You can specify the maximum number of messages displayed per page in the results section by selecting a number from the *Max no of messages* drop-down list. In addition, you can move between the pages by using the previous page and next page icons (◀ ▶).

You can apply/modify tags to the messages listed in the results section of the advance search window by selecting the check box for the messages and using the corresponding tagging icons (). See the *Tag* section for details about tagging messages.

You can also select all the messages displayed in this window by selecting the *Select All* option. You can then perform a common action on all the selected messages. To deselect all the messages, deselect the *Select All* option.

Cross-probing from the Advanced Search Window

The messages listed in the results section of the *Advanced Search* window can be cross-probed to other Atrenta Console windows. To do so, click a message row in the results section and the following cross-probes are created:

- The source file in which the message is reported is highlighted in the *File View* page.
- The source code of the file in which the message is reported is highlighted in the *HDL Viewer Pane*.
- The message is highlighted in the *Msg Tree Page*.
- The corresponding schematic information (if available, indicated by the schematic ⊼ icon) is highlighted in *The Modular Schematic Window* and *The Incremental Schematic Window*.

SpyGlass does not allow you to cross-probe to the RTL of encrypted design units. If you try to cross-probe to the RTL of such design units, SpyGlass displays a message in the RTL viewer specifying that the file is encrypted.

Right-click Menu Options in the Msg Tree View Page

When you right-click on a violation message, the following options appear in the shortcut menu:

Help

Select this option to view rule help of the selected violation message in a separate browser.

Display Message

Select this option to view the complete message text in a separate balloon window.

You can select this option if a message text is long and does not fit in the available space.

Show BlackBox Info

This option appears for black box-related violations, such as AnalyzeBBox.

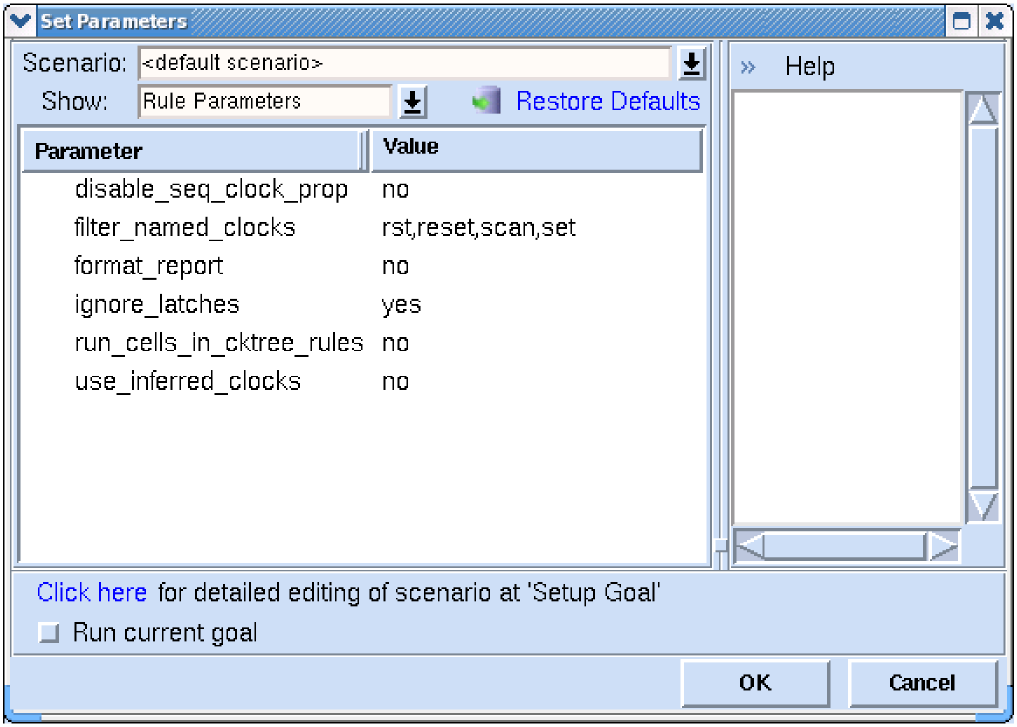
Select this option to view the detailed information about the cause and remedy of black boxes.

When you select this option, the *Black Box Viewer Window* appears.

Edit Parameters

Select this option to view or modify parameters applicable to the rule of the selected violation message.

When you select this option, the Set Parameters dialog appears, as shown in the following figure:



Scenario: <default scenario>

Show: Rule Parameters

Parameter	Value
disable_seq_clock_prop	no
filter_named_clocks	rst,reset,scan,set
format_report	no
ignore_latches	yes
run_cells_in_cktree_rules	no
use_inferred_clocks	no

Click here for detailed editing of scenario at 'Setup Goal'

□ Run current goal

FIGURE 29. Set Parameters

By default, the above dialog displays parameters for the scenario that was last run.

If you want to change parameter values of a different scenario, perform the following steps:

1. Select the required scenario from the Scenario drop-down list.

NOTE: *You can also create a new scenario by selecting the Create New Scenario option from this drop-down list. When you select this option, the Create New Scenario dialog appears in which you can specify the name of the new scenario. By default, this new scenario has the same settings as that of the last modified scenario.*

2. Update parameter values as per your requirement.
3. Select the Run current goal option if you want to save the scenario and automatically run that scenario after the next step.

4. Click the OK button.

After performing the above steps, the selected scenario is saved with the updated parameter values.

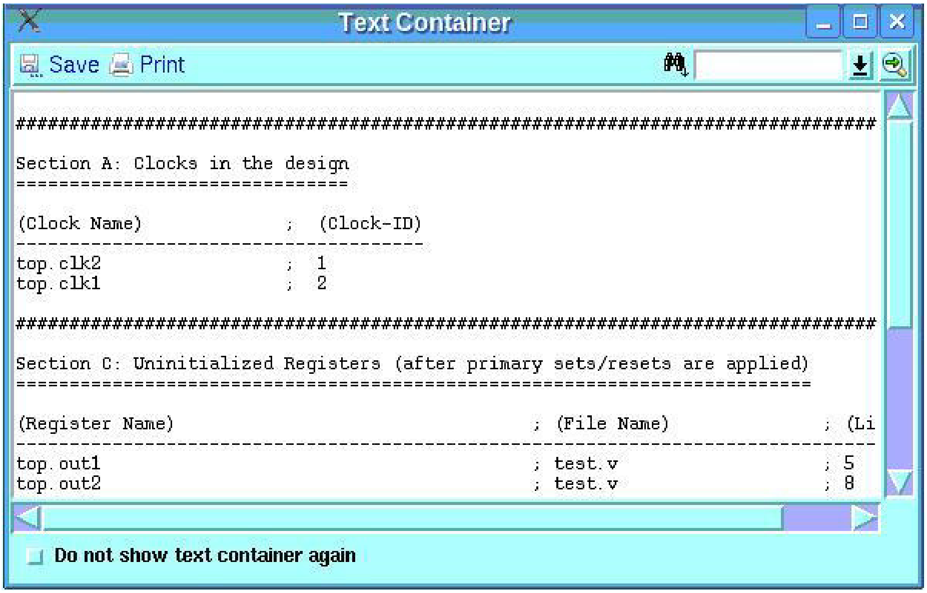
While updating parameter values of a particular scenario, if you select another scenario from the Scenario drop-down list, a Warning dialog appears that prompts you to save changes in the current scenario.

If you want to specify additional details (other than parameters) for a scenario, click the Click here link in the above dialog. This displays a page under the Setup Goal tab under which you can specify additional details, such as SGDC files for a scenario.

Text Viewer

Select this option to view the text container for a message, which is indicated by a T icon in front of the message. You can also open Text Viewer by double-clicking on the violation message.

Clicking on the Click Text Viewer option displays the following text container for the selected message:



```
################################################################################

Section A: Clocks in the design
===============================

(Clock Name)              ;  (Clock-ID)
--------------------------------------
top.clk2                  ;  1
top.clk1                  ;  2

################################################################################

Section C: Uninitialized Registers (after primary sets/resets are applied)
==========================================================================

(Register Name)                                ; (File Name)                ;  (Li
--------------------------------------------------------------------------------
top.out1                                       ; test.v                     ;  5
top.out2                                       ; test.v                     ;  8
```

Copy

Select this option to copy the message text (full text or partial text).

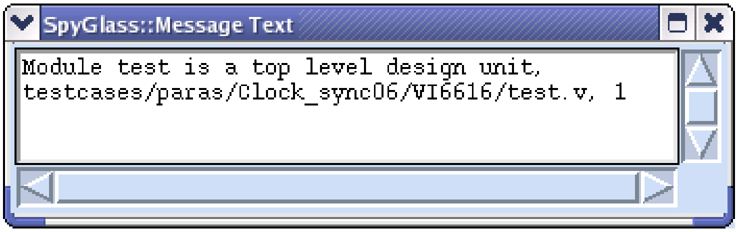
When you select this option, the following options appear in the sub-menu:

- *MsgText*: Select this option to copy complete text of the selected message. Alternatively, you can use *<Ctrl>+<C>* keys or *<Ctrl>+<Insert>* keys to copy the selected message.

  The message text is copied on the clipboard. You can then paste the text in any other application that supports pasting of text.

- *Custom*: Select this option to copy partial text of a message.

  When you select this option, The *Message Text* window dialog appears displaying the complete message text, as shown in the following figure:



```
Module test is a top level design unit,
testcases/paras/Clock_sync06/VI6616/test.v, 1
```

FIGURE 30. Message Text

In the above dialog, you can select a partial text, right-click on that text, and select the required option from the shortcut menu to perform the desired action.

For example, you can copy/paste the text, search for multiple occurrences of a selected word/character, jump to a particular line within the *Message Text* dialog, and wrap the text (word wrap or character wrap) within the *Message Text* dialog.

NOTE: *You can also select multiple messages (using <Ctrl>+Click) and perform the above functions, such as copying the text of all the selected messages.*

Save Message List

Select this option to save the contents of the *Msg Tree* page.

When you select this option, the *Save Report As* dialog appears in which you can specify the name, type, and path of the file in which you want to save the message.

Select Default Waiver File

Select this option to specify a waiver file to be used by default.

For details, refer to the *Setting a Default Waiver File* topic of *Atrenta Console User Guide*.

Waive Selected Messages

Select this option to waive the selected message.

For details on this option, refer to the *Waiving Selected Messages* topic of *Atrenta Console User Guide*.

Waive

Select this option to waive messages based on a criterion. You can specify this criterion by selecting an option from the sub-menu of the *Waive* option.

NOTE: *This option appears only if the Enable advanced waiver creation preference option is set.*

For details on this option, refer to the *Waiving Specific Type of Messages* topic of *Atrenta Console User Guide*.

Tag

Select this option to add a tag to the selected message.

When you select this option, a sub-menu appears from which you can select the required option.

These options enable you can tag rule messages either with certain predefined identifiers, such as *Investigate*, *Fixed*, *ToFix*, and *VerifiedFixed* or with your own tags.

Tagging of messages with identifiers helps you keep track of the rule violations in your design that are reported as a result of design analysis. For example, in a single analysis run, you can analyze a number of violations and assign appropriate tags based on the actions that you have performed (such as *Fixed* or *VerifiedFixed*) or intend to perform later (such as *Investigate* or *ToFix*).

This way, you need not repeatedly run the analysis to update the status of the violations after performing a set of corrective actions in your design to address those rule violations.

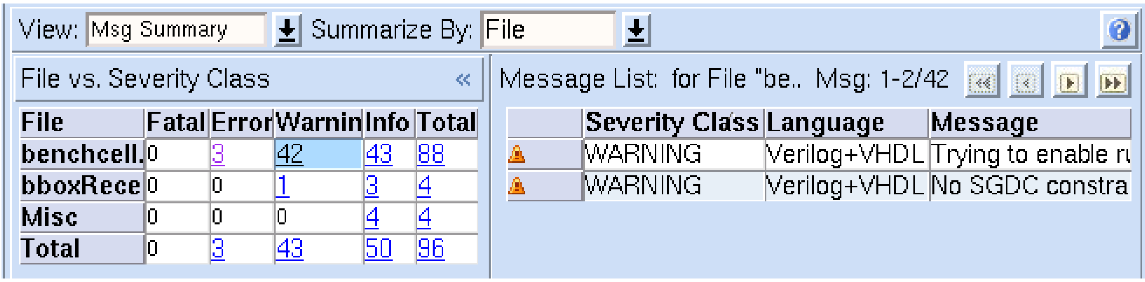
Preferences

Select this option to display the *Edit > Preferences* dialog.

Msg Summary Page

The *Msg Summary* page displays an interactive summary of reported violation messages.

The following figure shows the *Msg Summary* page:



View: Msg Summary Summarize By: File

File vs. Severity Class

File	Fatal	Error	Warnin	Info	Total
benchcell.	0	3	42	43	88
bboxRece	0	0	1	3	4
Misc	0	0	0	4	4
Total	0	3	43	50	96

Message List: for File "be.. Msg: 1-2/42

	Severity Class	Language	Message
⚠	WARNING	Verilog+VHDL	Trying to enable ru
⚠	WARNING	Verilog+VHDL	No SGDC constra

FIGURE 31. Msg Summary

To view the help of the *Msg Summary* page, click ? button. This displays the *Message Summary Help* window.

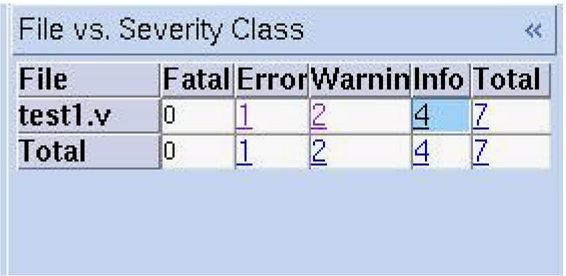
The above page is divided into the following sections:

- *Summary Matrix Section*: This section displays the total number of messages categorized in a tabular format.
- *Message List Section*: This section displays individual violation messages with their corresponding violation IDs in a tabular format.

Summary Matrix Section

The *Summary Matrix* section displays the total number of different severity messages summarized by file, module/design unit, tag, or goal.

The following figure displays the *Summary Matrix* section:



File vs. Severity Class

File	Fatal	Error	Warnin	Info	Total
test1.v	0	1	2	4	7
Total	0	1	2	4	7

FIGURE 32. Summary Matrix

In the above section:

- Each row displays the number of messages summarized by a file, module/design unit, tag, or goal as selected in the *Summarize By* drop-down list.

  By default, messages are summarized either by files or by goals (if any goals are selected).

- Each column displays the number of messages summarized by severity class.

  Therefore, each cell in the *Summary Matrix* section displays the total number of messages of a particular severity class reported in a particular file/module/design unit (as the case may be).

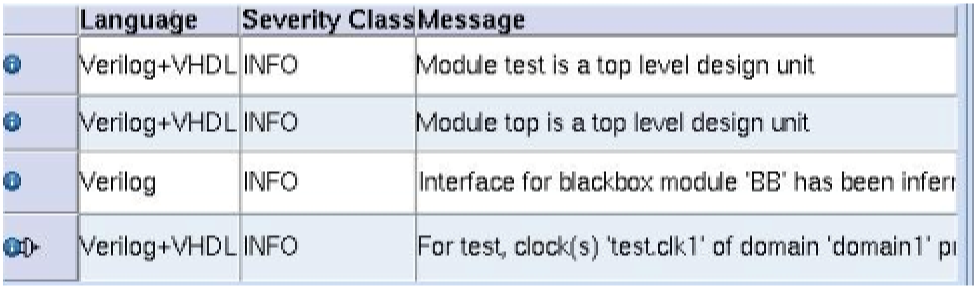
Placing the pointer on a cell displays a balloon window that contains information about the messages contained in that cell.

NOTE: *You can resize each row and column of the Summary Matrix section by dragging the edges of the row/column.*

When you click on a cell that has a non-zero entry, Atrenta Console updates the *Message List Section* accordingly with the appropriate violation messages.

Message List Section

The *Message List* section displays a list of messages in a tabular format, as shown in the following figure:



	Language	Severity Class	Message
ⓘ	Verilog+VHDL	INFO	Module test is a top level design unit
ⓘ	Verilog+VHDL	INFO	Module top is a top level design unit
ⓘ	Verilog	INFO	Interface for blackbox module 'BB' has been inferr
	Verilog+VHDL	INFO	For test, clock(s) 'test.clk1' of domain 'domain1' p

FIGURE 33. Message List

Depending upon the cell clicked in the *Summary Matrix Section*, the above section is automatically updated. Each row in this section displays the details of a particular violation message.

The violation messages are displayed based on severity. If a rule has multiple severities, the related rule messages are displayed separately with the severity.

NOTE: *The information about the currently selected cell of the Summary Matrix section is also displayed in the Msg Summary Toolbar.*

Viewing Message Information

If the cursor is placed over a message row in the Message List section, a small balloon window is displayed with the following information about the message:

- The tag applied to the message
- The goal that contains the rule corresponding to the displayed message
- The severity of the message
- The rule name corresponding to the displayed message

Viewing Rule Help

To view detailed information about a rule message, click the row containing the violation message in the *Message List* section. The help related to the rule is displayed in the *Help* section located on the right side of the *Message List* section.

Right-Click Menu Options in the Msg Summary Page

When you right-click on a row in the *Message List Section*, the following options appear in the shortcut menu:

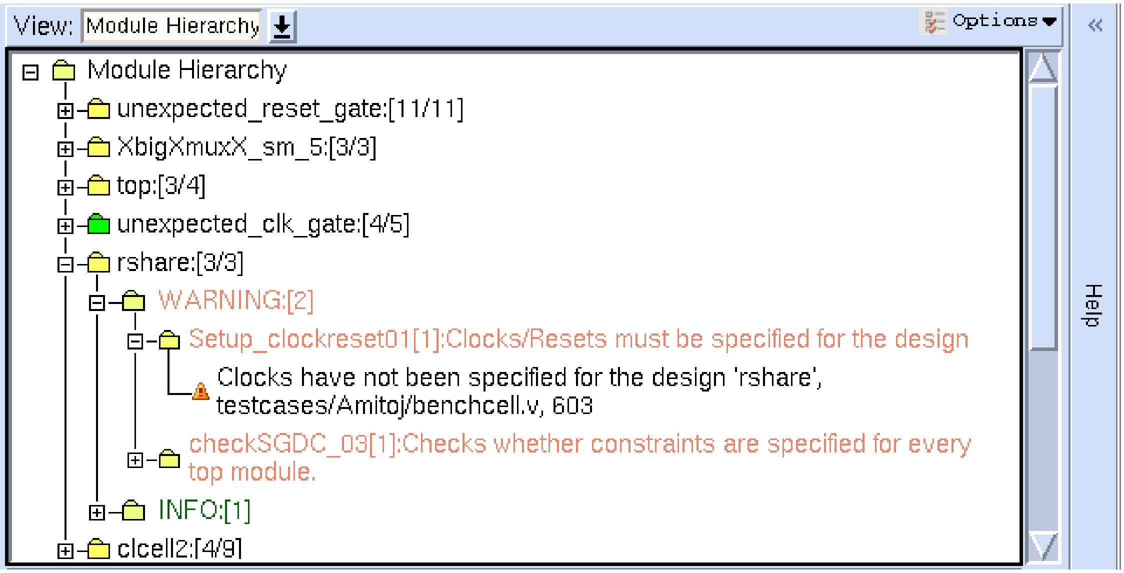
Option	**Description**
Display Message	For details, see *Display Message*.
Show Blackbox Info	For details, see *Show BlackBox Info*.
Edit Parameters	For details, see *Edit Parameters*.
Configure Columns	For details, see *Configure Columns*.
Configure Page Size	Select this option to specify the total number of rows to be displayed at a time. When you select this option, the *Configure Page Size* dialog appears in which you can specify the required number of rows to be displayed at a time. You can use the pagination buttons (⏮ ◀ ▶ ⏭) to move across the pages in the *Message List* section. NOTE: The pagination buttons are not displayed if the number of messages is less than or equal to the current page size.
Copy	Select this option to copy the selected cell or row. Based on your requirement, you can select *Copy > Copy Cell(s)* or *Copy >Copy Row(s)* option from the shortcut menu.
Save Message List	For details, see *Save Message List*.
Waive Selected Messages	For details, see *Waive Selected Messages*.
Tag	For details, see *Tag*.
Preferences	For details, see *Preferences*.

Module Hierarchy Page

The *Module Hierarchy* page displays a tree-view of the module hierarchy of your design. Child modules, if present, are displayed directly under the parent module in the module hierarchy.

Violation messages (grouped by severity) are displayed directly under each module.

The following figure displays the *Module Hierarchy* page:



```
Module Hierarchy
  unexpected_reset_gate:[11/11]
  XbigXmuxX_sm_5:[3/3]
  top:[3/4]
  unexpected_clk_gate:[4/5]
  rshare:[3/3]
    WARNING:[2]
      Setup_clockreset01[1]:Clocks/Resets must be specified for the design
        Clocks have not been specified for the design 'rshare',
        testcases/Amitoj/benchcell.v, 603
      checkSGDC_03[1]:Checks whether constraints are specified for every
      top module.
    INFO:[1]
  clcell2:[4/9]
```

FIGURE 34. Module Hierarchy

Viewing messages in the above format is useful in the following cases:

- When you want to identify a location where problems may exist through the hierarchy
- When sub-blocks are owned by different RTL designers

NOTE: *If a module is instantiated at different levels, messages related to that module are displayed at all places where the module is instantiated. Therefore, the count of total number of violations added through the hierarchy will be higher than the count of individual, flat display of messages. This difference could be very large if there are messages for low-level cells that are re-used through a design. For example, messages from instantiated technology library cells.*

If you place the cursor over a module, a balloon window appears that displays the following information:

- Total number of messages in a module
- Total number of messages in the module hierarchy
- Total number of instances of that module

Right-Click Menu Options in the Module Hierarchy Page

When you right-click on a module in the *Module Hierarchy* page, the following options appear in the right-click menu:

Save Message List

Select this option to save messages present in a module in a text or HTML format.

When you select this option, a sub-menu appears containing two options, *Html Format* and *Text Format*.

When you select the required option from the sub-menu, the *Save Report As* dialog appears in which you can specify the required details, such as name, type, and path of the file in which you want to save the message.

Show Instance List

Select this option to view a list of instances of the selected module.

For details on this option, see *Show Instance List*.

Preferences

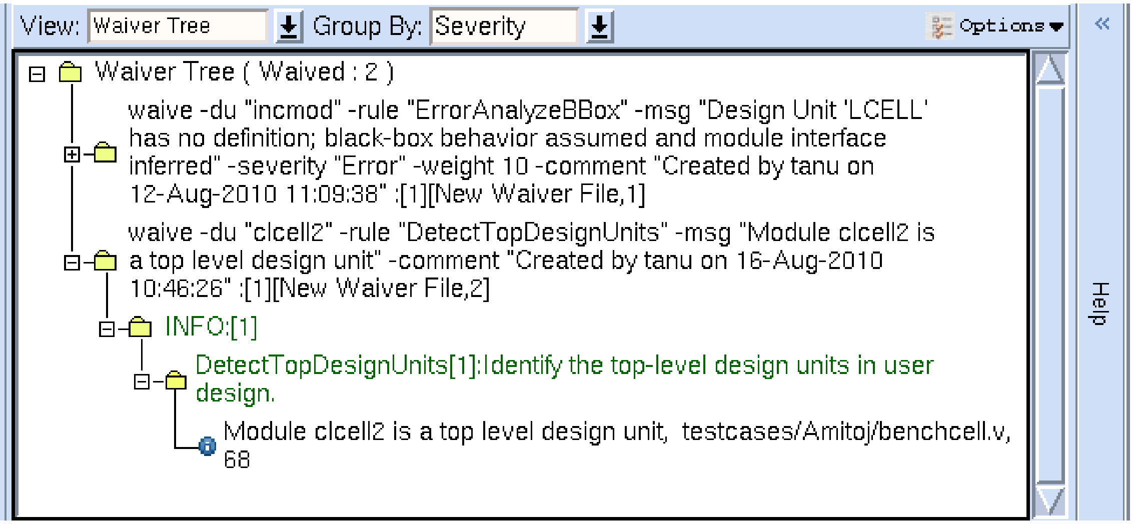
Select this option to display the *Preferences* dialog.

Waiver Tree Page

The *Waiver Tree* page displays the following details:

- A waiver tree containing messages waived specified by the waive constraints
- Comments applied to each of these messages
- Count of total number of waived messages

The following figure displays the *Waiver Tree* page:



Waiver Tree (Waived : 2)

waive -du "incmod" -rule "ErrorAnalyzeBBox" -msg "Design Unit 'LCELL' has no definition; black-box behavior assumed and module interface inferred" -severity "Error" -weight 10 -comment "Created by tanu on 12-Aug-2010 11:09:38" :[1][New Waiver File,1]

waive -du "clcell2" -rule "DetectTopDesignUnits" -msg "Module clcell2 is a top level design unit" -comment "Created by tanu on 16-Aug-2010 10:46:26" :[1][New Waiver File,2]

INFO:[1]

DetectTopDesignUnits[1]:Identify the top-level design units in user design.

Module clcell2 is a top level design unit, testcases/Amitoj/benchcell.v, 68

FIGURE 35. Waiver Tree

By default, messages in the above page are first grouped by waiver expressions and then by severity.

To group messages based on severity first and then waiver expressions, select Options, and then de-select the *Group by Waiver Expression first* option.

You can also set miscellaneous preferences by selecting *Preferences* option from the *Options* drop-down list, which displays the *Miscellaneous Page* of the *Preferences* dialog.

Viewing Waiver Messages in the Incremental Mode

You can view waived messages in the incremental mode so that you can compare the messages waived in the last run with the run.

To view waived messages in the incremental mode, select the Incremental Mode option in the GUI. When you select this option, the following nodes appear in the Waiver tree page:

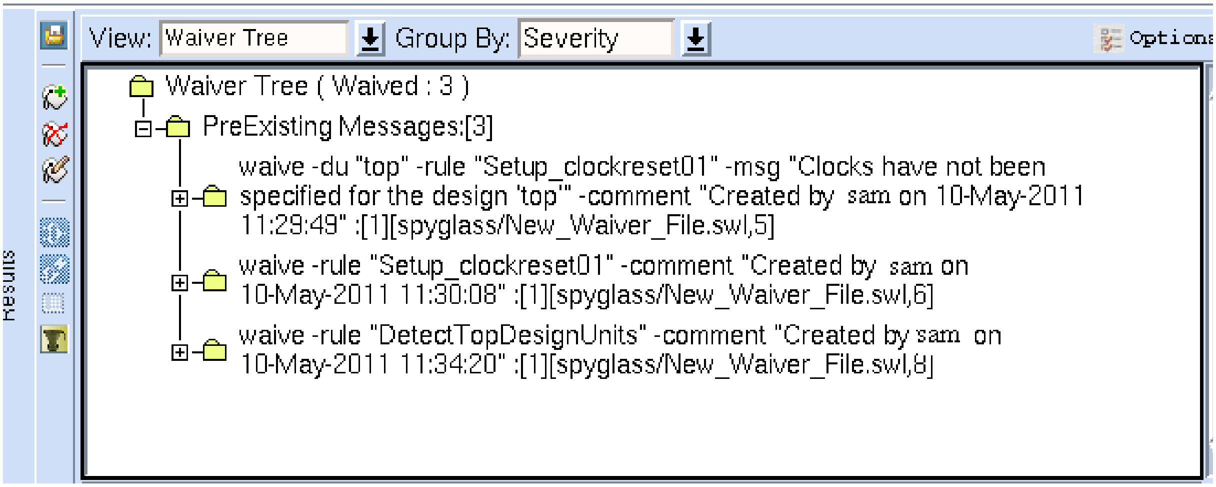
- New Messages

  This node contains messages that are waived in the current run.

- Pre-existing messages

  This node contains messages that were already waived in the last run as well are waived in the current run.

For example, the following figure shows the Pre-existing messages node:



```
Waiver Tree ( Waived : 3 )
  PreExisting Messages:[3]
    waive -du "top" -rule "Setup_clockreset01" -msg "Clocks have not been specified for the design 'top'" -comment "Created by sam on 10-May-2011 11:29:49" :[1][spyglass/New_Waiver_File.swl,5]
    waive -rule "Setup_clockreset01" -comment "Created by sam on 10-May-2011 11:30:08" :[1][spyglass/New_Waiver_File.swl,6]
    waive -rule "DetectTopDesignUnits" -comment "Created by sam on 10-May-2011 11:34:20" :[1][spyglass/New_Waiver_File.swl,8]
```

FIGURE 36. Viewing Waiver Messages in the Incremental Mode

Right-Click Menu Options in the Waiver Tree Page

When you right-click on a waiver expression displayed in the *Waiver Tree* page, the following options appear in the shortcut menu:

Edit

Select this option to edit waiver settings.

When you select this option, the *Waiver Editor Window* appears in which you can make the required updates.

Disable

Select this option to remove the selected waiver from the *Waiver Tree* page.

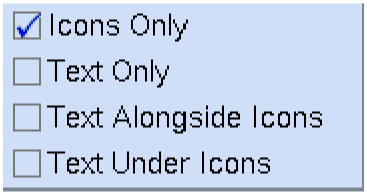
When you select this option, the selected waiver appears disabled in the spreadsheet of the *Waiver Editor window*. That is, its entry still remains in the *Waiver Editor window*.

When you right-click on a violation message in the *Waiver Tree* page, the following options appear in the shortcut menu:

Option	Description
Display Message	For details, see *Display Message*.
Save Message List	Select this option to save the contents of the Waiver Tree page. When you select this option, the *Save Waiver Tree As* dialog appears in which you can specify the name, type, and path of the file in which you want to save the data.
Copy	For details, see *Copy*.
Tag	For details, see *Tag*.
Preferences	Select this option to display the *Preferences* dialog.

Right-Click Options of the Results Pane

When you right-click on the icons displayed on the right-most side of the Results pane, the following options appear:



☑ Icons Only
☐ Text Only
☐ Text Alongside Icons
☐ Text Under Icons

FIGURE 37. Results Pane - Right-Click Options

Icons Only

Select this option to view only icons for various options, as shown in the following figure:

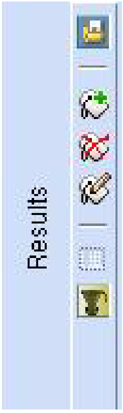


FIGURE 38. Results Pane - Icons Only

Text Only

Select this option to view only text for various options, as shown in the following figure:

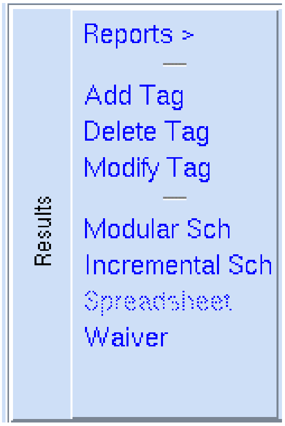


FIGURE 39. Results Pane - Text Only

Text Alongside Icons

Select this option to view text along with the corresponding icons for various options, as shown in the following figure:

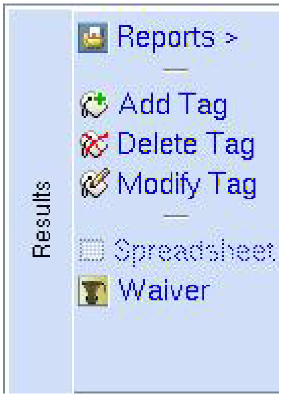


FIGURE 40. Results Pane - Text Alongside Icons

Text Under Icons

Select this option to view text under the corresponding icons for various options, as shown in the following figure:

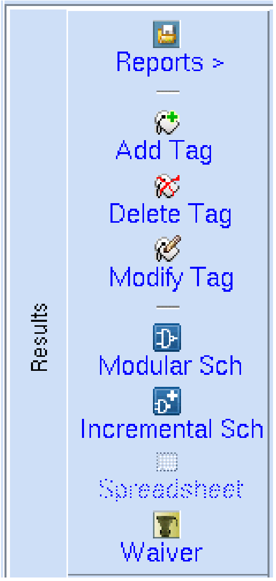


FIGURE 41. Results Pane - Text Under Icons

Interaction with Other Windows

You can probe a message in most pages of the Message window by double-clicking the message. Then, the related objects are highlighted in other windows that are open, as follows:

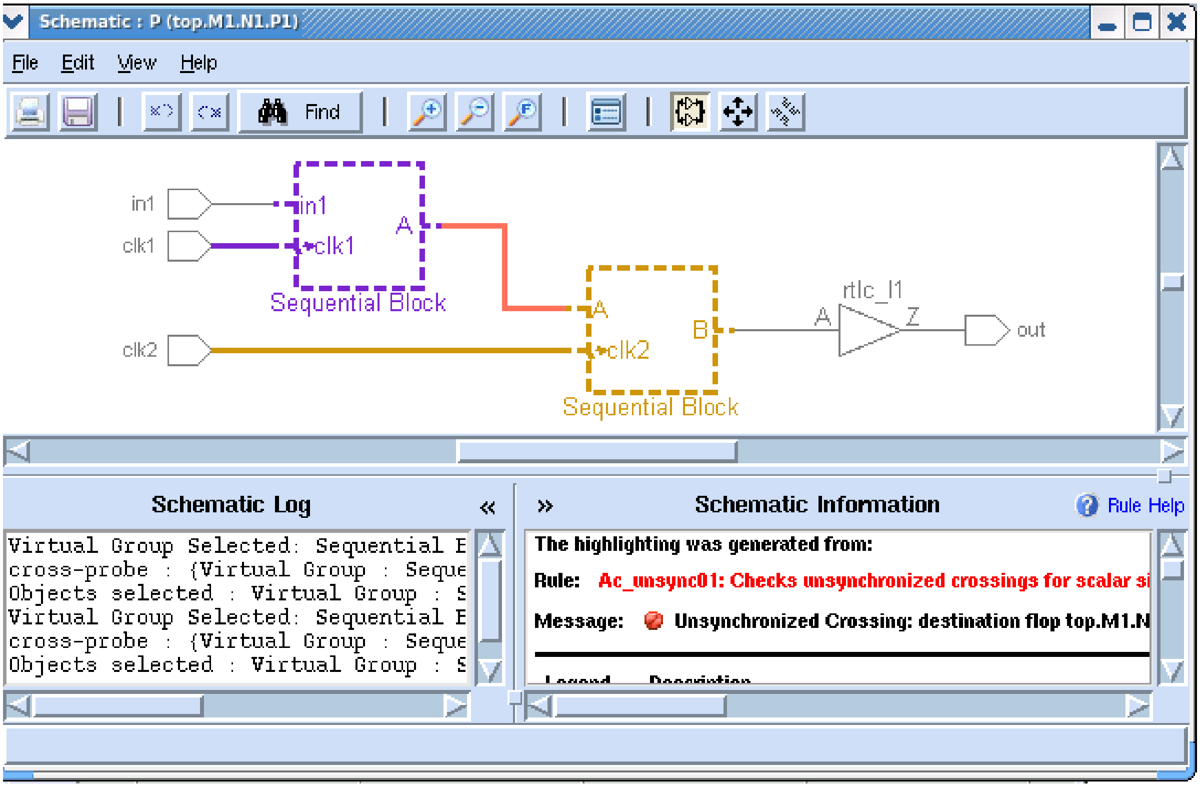
Object Type	Window Actions	Highlight Details
Message	The source file containing the message is loaded in the HDL Viewer and the related source code line is highlighted.	The source code line is highlighted.
	Modular Schematic window is redrawn to show the design unit containing the message.	Components involved in the message are highlighted.
	Components involved in the message are added to the Incremental Schematic window.	Components involved in the message are highlighted across the hierarchical boundaries.
	One or more entries are added to the Legend window list depending on the number of colors associated with the message (as set in the product).	-

The Modular Schematic Window

The *Modular Schematic* window displays a hierarchical schematic layout of the modules analyzed by SpyGlass. This window displays data based on the violation message selected or a module selected from the *Module View Page*.

To display the *Modular Schematic* window, click the *Modular Sch* link in the *Results* pane.

The following figure shows the *Modular Schematic* window:



```
Virtual Group Selected: Sequential E
cross-probe : {Virtual Group : Seque
Objects selected : Virtual Group : S
Virtual Group Selected: Sequential E
cross-probe : {Virtual Group : Seque
Objects selected : Virtual Group : S
```

The highlighting was generated from:

Rule: **Ac_unsync01: Checks unsynchronized crossings for scalar si**

Message: **Unsynchronized Crossing: destination flop top.M1.N**

Legend **Description**

FIGURE 42. The Modular Schematic Window

The *Modular Schematic* window is divided in to the following sections:

The schematic display	*Schematic Information*	*Schematic Log*
Toolbar	*Menu bar*	

The schematic display

This section displays all the components (across hierarchical boundaries) pertaining to the selected message or a probed component.

You can select multiple messages at a time and view all the components pertaining to the selected messages.

When you try to view a flattened design unit with very large number of objects (instances, nets, and ports) in the main schematic window, a warning is displayed and the schematic window is closed automatically. Please note that this issue relates to flattened design unit size and is not related to the overall design size.

Set the value of the AUTOENABLE_HUGE_SCHEMATIC_DISPLAY configuration key to yes. Once changed to yes, a warning is still displayed but the user can continue the action that may result in a long wait, hanging of Atrenta Console, or out of memory situation. This would depend on design size and available memory on the system.

Menu bar

Refer to the *Incremental Schematic Window Menu Bar* topic.

Toolbar

The toolbar contains buttons, such as *Print*, *Find*, *Undo*, *Redo*, *Zoom In*, *Zoom Out*, *Zoom Fit*, and *Preferences*.

Schematic Log

This section shows messages based on the actions performed in the schematic.

Schematic Information

This section shows the following information:

- The meaning of colors and text attributes applied on the objects in the schematic.
- The rule name, rule summary, and message of the selected violation.

You can view the complete rule help by clicking Rule Help. This displays the *Rule Help* window containing the complete rule help.

NOTE: *The information appearing in this section applies to the first selected message, that is, the primary message. It does not apply to the secondary messages.*

Using the Back Annotation Support

The *Modular Schematic* window enables back-annotation between the schematic and the source code.

For example, when you select a gate in the schematic, SpyGlass highlights the corresponding line in the RTL code (shown in *HDL Viewer Pane*) that generated that gate.

Black boxes appear in a different color in the schematic to distinguish them from other objects.

Creating RTL Groups

An RTL group is a group of a logic, such as:

- *Always/Process Blocks in the RTL Code*
- *Vectored Instances in the RTL Code*
- *Registers and Combinational Logic in the RTL Code*

Creating RTL groups in the *Modular Schematic* window makes it easier to navigate in the schematic and trace a particular logic.

RTL grouping is particularly useful in viewing a module that has many leaf-level instances.

Use the following toolbar buttons in the schematic to work with RTL groups:

Button	Description
⊳	Click this button to enable the RTL grouping.
✥	Click this button to expand all groups and their subgroups in the current module. Other modules remain unaffected.
✣	Click this button to collapse all expanded subgroups and objects in the current module to view consolidated groups. Other modules remain unaffected

The filtered schematic view (Incremental Schematic or IS), which is commonly used for debugging rule violation messages, does not currently utilize the grouping capabilities in order that information does not get hidden while debugging specific schedule instances. Moreover, gate-level (structural) schematics also do not utilize the grouping capabilities

Always/Process Blocks in the RTL Code

The following RTL blocks in a module definition are grouped in the modular schematic view:

- process blocks (used in VHDL)
- always blocks (used in Verilog)

These blocks are further distinguished as sequential and combo blocks in the schematic view, as shown in the following figure:

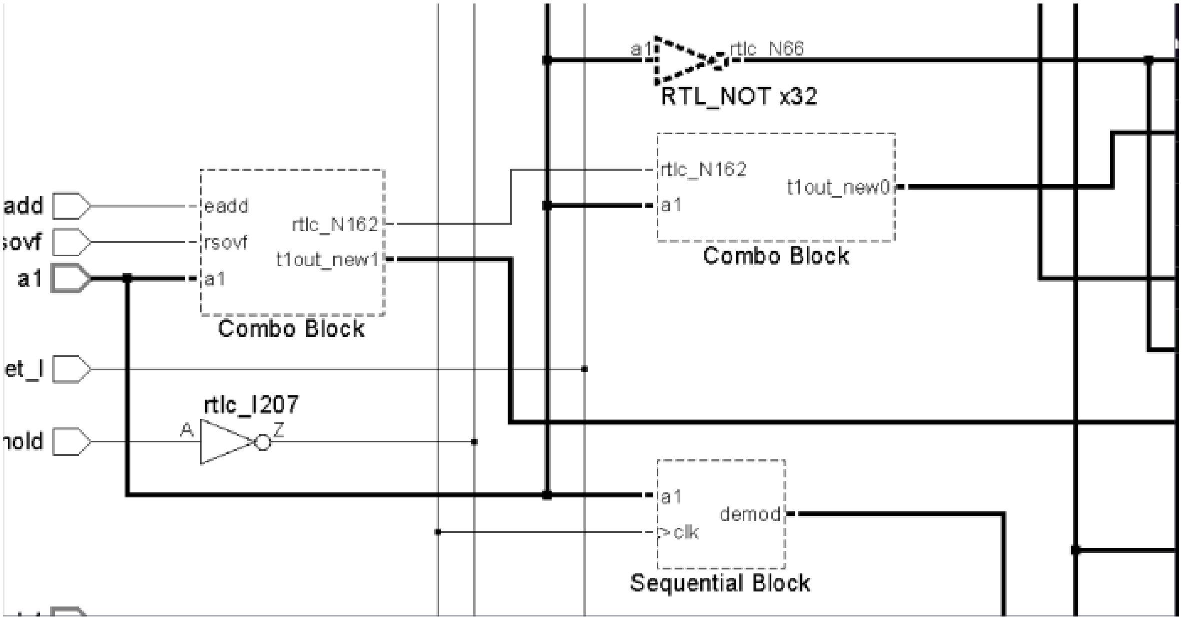


FIGURE 43.

In the above view:

- Each of these blocks is shown in the schematic display as a rectangle with dashed boundary. The boundaries of instances for which you can dive down their hierarchy are shown in a dash-dot pattern.

- The sequential blocks and the combo blocks are named as *Sequential Block* and *Combo Blocks*, respectively in the view.
- These blocks appear as single entities. To view the objects and sub-groups contained in the group, you can expand these entities by double-clicking them.

Vectored Instances in the RTL Code

The *Modular Schematic* window displays the vectored or arrayed instances in the RTL source code as single components with appropriate bus pins. Vectors share the same master module for all the instances in the code, and therefore, the names of the vector groups are derived from the master modules of the instances and the number of instances in that group.

For example, RTL_MUX x32 means that the group contains 32 instances of RTL_MUX.

The vector groups are generally shown in the schematic display as a rectangle with a thicker dotted boundary than normal groups or instances (analogous to net bundles being thicker than single nets). If a vector group has the same input/output mapping as that of the constituent instances, the vector group is shown with the same symbol as that of the instances in the schematic view.

For example, a vector group containing a buffer with a bus input and a bus output is shown by a buffer symbol having the same bus as input/output.

The following illustration shows a vector group representation in the Modular Schematic window. Note that the vector group has the same mapping as that of the constituent instances and therefore it is represented using the same symbol as that of the instances.

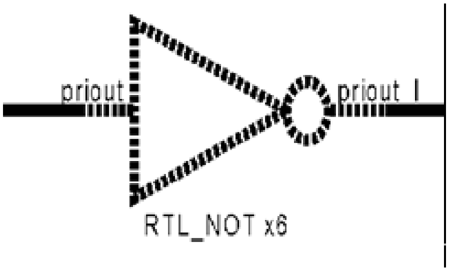


FIGURE 44. Vector Group Representation

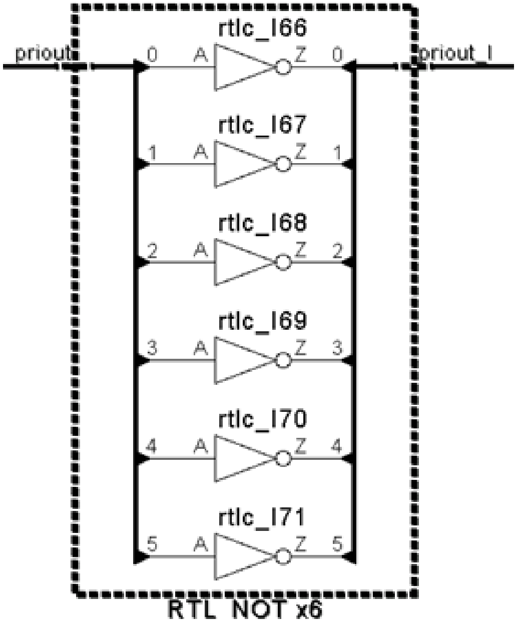


FIGURE 45. Vector Group Representation (Expanded)

Registers and Combinational Logic in the RTL Code

Consider the following figure:

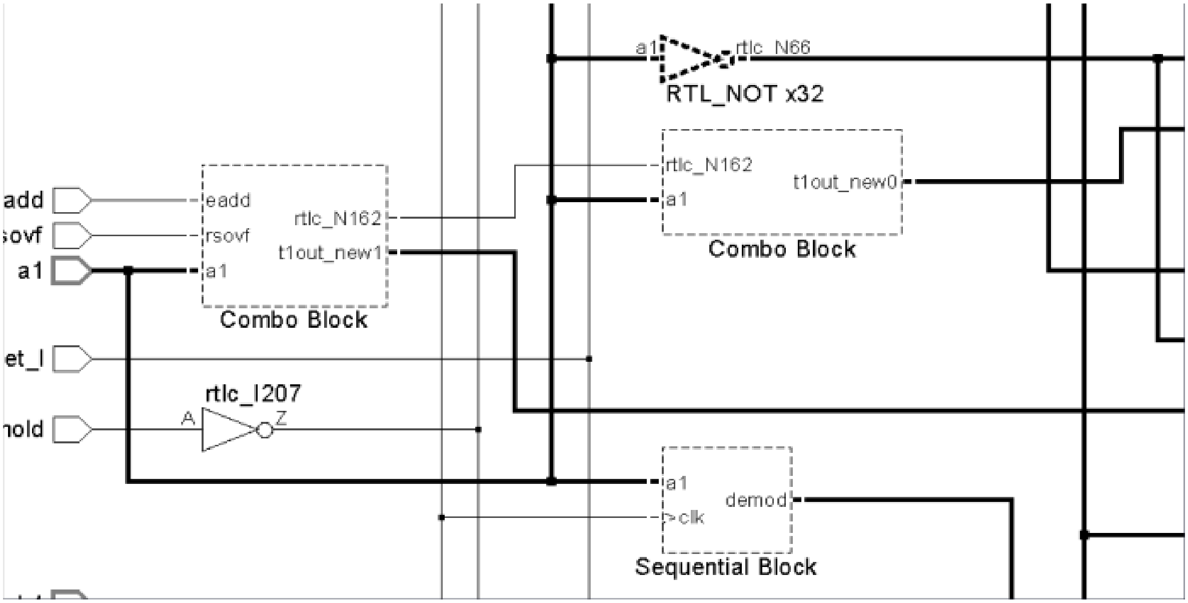


FIGURE 46. Vector Group Representation (Expanded)

The figure displays a group having purely combinational logic as *Combo Block* in the design. You can easily distinguish the combinational logic blocks from the sequential blocks that are shown as *Sequential Block*).

Interaction with Other Windows

When you click an object in the *Modular Schematic* window, SpyGlass cross-probes that object in other windows, as described in the following table:

Object Type	Window Actions	Highlight Details
Nets	HDL Viewer has the source file containing the net displayed.	The line where the net is created or inferred is grooved. If the net name is present in the RTL line, the net name is also highlighted.
	Net and all connected components are added to the Incremental Schematic window.	Net is highlighted across the hierarchical boundaries.
	Net is added to the Legend window list.	-
Ports	HDL Viewer has the source file containing the port displayed.	The line where the port is created is grooved and the port name is highlighted.
	Port and all components connected to the port's connected net are added to the Incremental Schematic window.	Port and its connected net are highlighted across the hierarchical boundaries.
	The net connected to the port is added to the Legend window list.	-
Instances	HDL Viewer has the source file containing the instance displayed.	The line where the instance is created or inferred is grooved. If the instance name is present in the RTL line, the instance name is also highlighted.
	Instance icon is added to the Incremental Schematic window.	Instance is highlighted.
	Instance is added to the Legend window list.	-

Object Type	Window Actions	Highlight Details
Pins	HDL Viewer has the source file containing the pin displayed.	The line where the pin is created or inferred is grooved. If the name of the net connected to the pin is present in the RTL line, the net name is also highlighted.
	Pin and all components connected to the pin's connected net are added to the Incremental Schematic window.	Pin and its connected net are highlighted across the hierarchical boundaries.
	The net connected to the pin is added to the Legend window list.	-

NOTE: *When you probe an element of a precompiled VHDL library cell in the Modular Schematic window or the Incremental Schematic window, SpyGlass performs a textual search for the element in the corresponding VHDL library cell source file and displays it.*

Using the Cursor

You can use the cursor in different ways to perform different actions in the schematic.

- *Single-Clicking*
- *Double-Clicking*
- *Click-Drag Combinations*
- *Right-Clicking*

Single-Clicking

You can perform the following actions in the *Modular Schematic* window by through single-clicking:

- Single-click an un-probed schematic component to remove all other probes and to start probing for that component. See *Interaction with Other Windows* for more details.

- Repeatedly single-click on an object to traverse all occurrences of the name of that object in a design by using simple text search of an object name.
- <Ctrl>+single-click on an un-probed object to probe it.
- <Ctrl>+single-click on a probed object to deselect its probe without affecting other probes.
- Select multiple objects by using <Ctrl> + single click.

  You can select only up to 32 objects in the *Modular Schematic* window. If you select more than 32 objects, a warning dialog appears prompting you to deselect some of the selected objects.

By default, single-click probe is disabled. Select the *Enable Single-click Probe* option in the *Schematic* page of the *Preferences* dialog.

Double-Clicking

When you double-click on a schematic component in the *Modular Schematic* window, the module, gate, or net appears (drills-down) to the schematic of that component (if available). In addition, the section of code describing the component in the source file is highlighted in the HDL Viewer.

If the component is a module that has its own description file, the source file is opened in the *Source* section and is highlighted in the *File/Design/Constraints/Instances* page.

Click-Drag Combinations

Keeping the mouse button pressed, when you drag the cursor in a particular direction, the modular schematic view changes accordingly.

You can drag the cursor in any of the following directions to see different views:

Northwest-to-Southeast

Dragging in this direction zooms in the selected area in the schematic.

On releasing the mouse button, the zoomed-in area fills the entire schematic view.

Northeast-to-Southwest

Dragging in this direction and releasing the mouse button displays the details of current module in the entire schematic view. This is called zoom to fit.

Southeast-to-Northwest

Dragging in this direction results in going up the hierarchical tree to display the schematic of the parent module.

If the parent module had already been displayed previously, it will inherit the zoom settings of the prior visit. If the parent module had not been previously viewed, it will open in the zoom to fit mode.

Southwest-to-Northeast

Dragging in this direction decreases the magnification of the schematic view.

The farther you drag the cursor, the magnification reduces accordingly.

A short drag in this direction results in the viewing area to zoom out by a small multiplier, while dragging from one corner of the screen to the other will cause the viewing area to zoom out by a large multiplier.

Right-Clicking

Use the right-click menu options in the *Modular Schematic* window to perform the following actions.

Viewing Object Name	*Finding Objects*	*Changing the Color of an Instance*
Viewing Object Properties	*Loading or Appending to the Incremental Schematic Window*	*Viewing RTL Source Code of an Object*
Viewing Subgroups	*Hiding Subgroups*	*Viewing Case Analysis Settings on an Object*
Viewing SDC and SGDC Constraints Set on an Object	*Viewing Connected Nets*	*Viewing Library Cell Instance Information*

Viewing SpyGlass-Generated Cell Macro Definitions	*Cross-Referencing to Power Data*	*Viewing Debug Data*
Show Activity and Probability Annotation	*Generating Constraints*	*Viewing Liberty Files*

Viewing Object Name

When you right-click on an object in the schematic, the name of that object appears as the first option of the shortcut menu.

Finding Objects

To find an object in the schematic, right-click in the schematic and select the *Find* option from the shortcut menu.

The *Find Dialog* appears, as shown in the following figure:

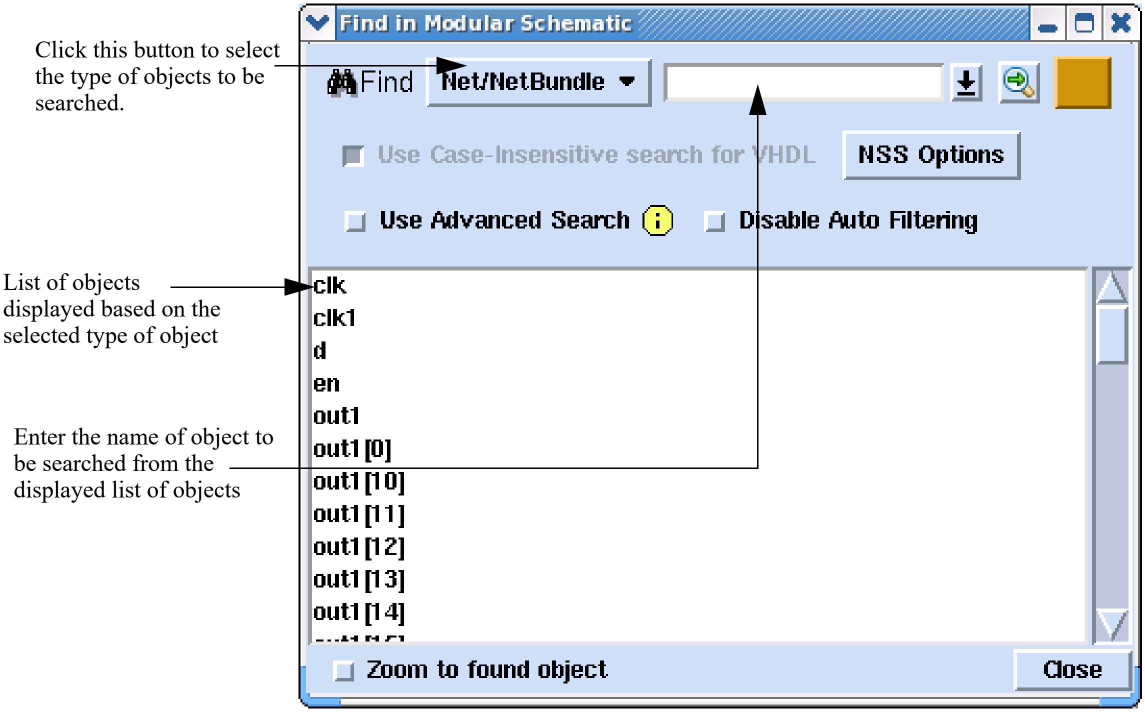

Click this button to select the type of objects to be searched.
List of objects displayed based on the selected type of object
Enter the name of object to be searched from the displayed list of objects
Find in Modular Schematic
Find
Net/NetBundle
Use Case-Insensitive search for VHDL
NSS Options
Use Advanced Search
Disable Auto Filtering
clk
clk1
d
en
out1
out1 [0]
out1 [10]
out1 [11]
out1 [12]
out1 [13]
out1 [14]
Zoom to found object
Close

FIGURE 47. Modular Schematic Window - Find Dialog

In the above dialog, select the object type to be searched after clicking the button adjacent to Find , as shown in the following figure:

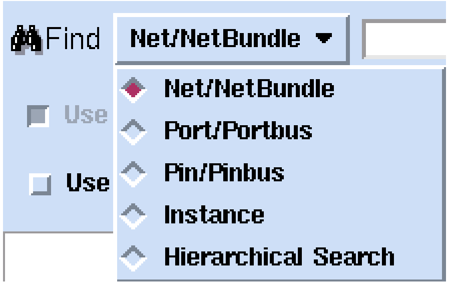


FIGURE 48. Find - Select Object Type

By default, *Net/NetBundle* is selected as the object type.

Once you find the required object in the displayed list in the *Find* dialog, select that object and then click the find icon (🔍). When you click this icon, the selected object is:

- Highlighted in the *Modular Schematic* window.

  If you have selected the *Zoom to found object* option in the *Find* dialog, the tool zooms-in the size of the object in the schematic.
- Highlighted in the *HDL Viewer Pane*.
- Added to the *Legend Window*, if open.

In addition, you can perform the following actions in the *Find* dialog:

Using Automatic Filtering	*Using Advanced Search*
Changing the Highlight Color	*Viewing Source and Sinks*
Searching Objects by Their Hierarchical Names	

Using Automatic Filtering

Automatic filtering is the process in which SpyGlass progressively refreshes the displayed list of object names in the *Find* dialog as you enter characters in the textbox in this dialog.

To disable this feature, select the *Disable Auto Filtering* option.

NOTE: *The search is case-insensitive. Therefore, typing* `foo` *will find and show* `foo`*,* `fOo1`*,* `FOO`*,* `Foo23`*, etc.*

Using Advanced Search

In this type of search, you can specify wildcard expressions for searching objects.

To implement advanced search, perform the following steps:

1. Select the *Use Advanced Search* option in the *Find* dialog.
2. Specify a regular expression containing wildcard characters in the textbox.

Changing the Highlight Color

To change the color in which the found object should be highlighted in the schematic, perform the following steps:

1. Click the color indicator icon (🟥) that shows the currently selected highlight color.

   The *Choose Color* dialog appears.
2. Select a color from the *Choose Color* dialog.
3. Click the *OK* button to close the *Choose Color* dialog.

Viewing Source and Sinks

To view sinks and sources for the searched object, click the *NSS Options* button in the *Find* dialog. When you click this button, the following drop-down list appears:

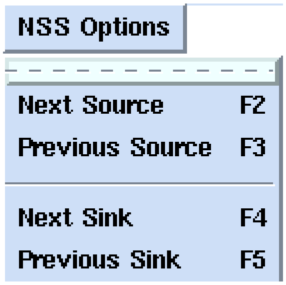
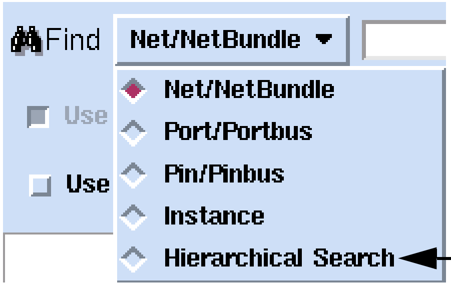


FIGURE 49. NSS Options

Select an option from the above list.

Searching Objects by Their Hierarchical Names

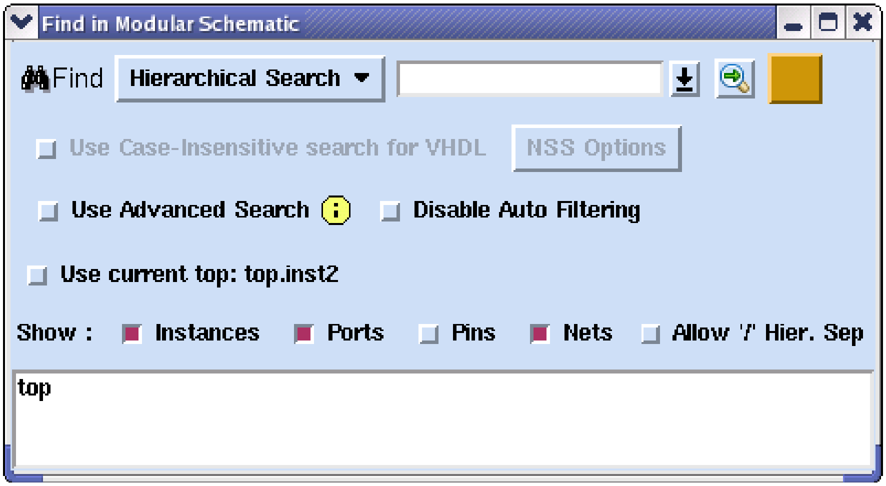
To search objects, such as instances, nets, pins, and ports by their hierarchical names, select the *Hierarchical Search* option from the drop-down list, as shown below:



Select this option to implement hierarchical search

FIGURE 50. Hierarchical Search

When you select this option, additional search options appear in the *Find* dialog, as shown in the following figure:



List of top-level design units displayed here, by default

FIGURE 51. Find in Modular Schematic

In the above dialog, you can perform the following actions:

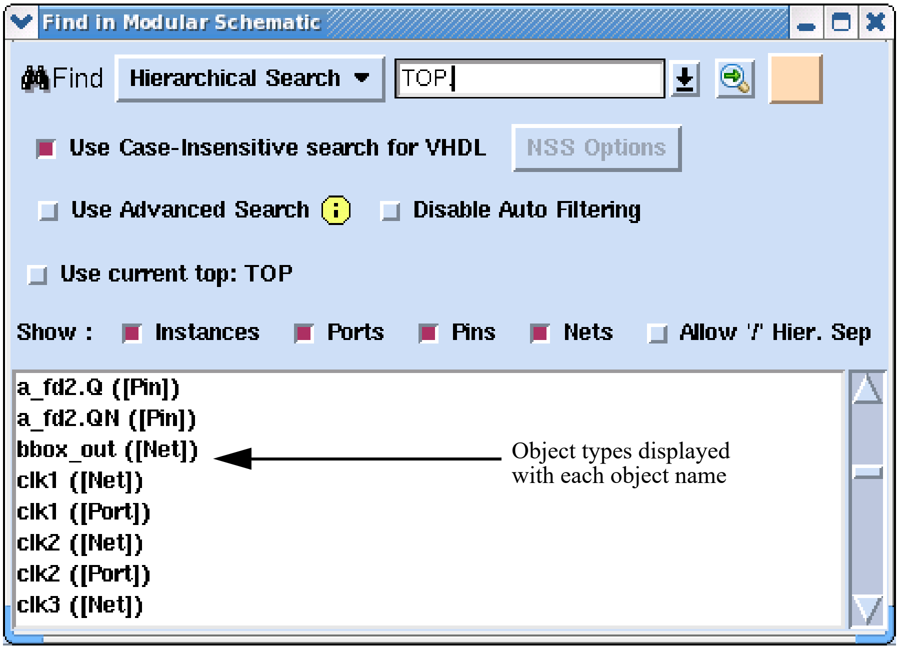
- Specify a search string in the textbox.

  As you start specifying the starting characters of a valid hierarchical name (using the dot hierarchy separator), the list of objects is progressively refreshed in this dialog.

- Dive down the hierarchy of an object displayed after the search.

To dive down the hierarchy of an object, double-click on the object name in the dialog. This action displays the list of objects within the hierarchy of the object double-clicked.

In the displayed list, object type (net, net bundle, port, pin, port bus etc.) appears along with each object name, as shown in the following figure:



Find: Hierarchical Search ▾ TOP

☒ Use Case-Insensitive search for VHDL

☐ Use Advanced Search ☐ Disable Auto Filtering

☐ Use current top: TOP

Show : ☒ Instances ☒ Ports ☒ Pins ☒ Nets ☐ Allow '/' Hier. Sep

a_fd2.Q ([Pin])
a_fd2.QN ([Pin])
bbox_out ([Net])
clk1 ([Net])
clk1 ([Port])
clk2 ([Net])
clk2 ([Port])
clk3 ([Net])

Object types displayed with each object name

FIGURE 52. Find in Modular Schematic - Results

- Filter the list of objects based on object types, such as instances, ports, pins, nets, or a combination of these types.

  To perform such filtering, select appropriate check boxes, such as *Instances*, *Ports*, *Pins*, and *Nets* in the *Find* dialog.

- Perform case-insensitive search

  You can perform case-insensitive search for finding objects by hierarchical names in the VHDL and mixed-language modes. To implement this search, select the *Use Case-Insensitive search for VHDL* option.

This option is enabled (only in VHDL and mixed-language modes) when you select the *Hierarchical Instance* search option.

NOTE: *The Use Case-Insensitive search for VHDL option is selected by default in the VHDL/ mixed-language modes. Case insensitive search is performed only inside the VHDL modules. However, this functionality is not available for Verilog modules in the mixed-language mode.*

Changing the Color of an Instance

To change the color of an instance shown in the schematic, perform the following steps:

1. Right-click on the object and select the *Colors* option from the shortcut menu.

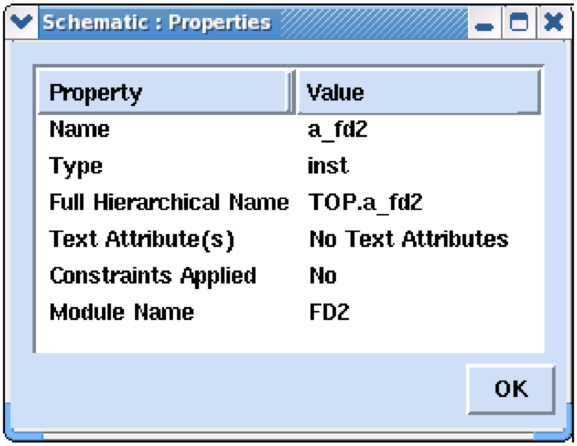
   The *Choose color* dialog appears.
2. Select a color from the *Choose color* dialog.
3. Click the OK button to close the *Choose color* dialog.

After performing the above steps, the color of the selected instance is updated in all windows.

Viewing Object Properties

To view the basic properties of an object displayed in the schematic, right-click on the object and select *Properties* option from the shortcut menu.

The *Properties* dialog appears, as shown below:



Property	Value
Name	a_fd2
Type	inst
Full Hierarchical Name	TOP.a_fd2
Text Attribute(s)	No Text Attributes
Constraints Applied	No
Module Name	FD2

FIGURE 53. The Properties Dialog

The above dialog lists object properties, such as object name, object type, complete hierarchical object name, and text attributes attached to the object.

NOTE: *For net bus, port bus, or pin bus, the text attributes are shown in a tabular form, listing only those bits that have some text attributes attached.*

Copying Object Properties

To copy the text of an object property, select a property from this dialog and perform any of the following actions:

- Select the *<Ctrl>+<C>* key combination.
- Right-click on a property, and select the required option (Copy Text or Copy Custom) from the shortcut menu.

Loading or Appending to the Incremental Schematic Window

You can load or append a port or an instance component appearing in the *Modular Schematic* window to *The Incremental Schematic Window*.

To perform such operation, select a port or instance and select the following options from the shortcut menu:

- *Load to IS*

Selecting this option clears *The Incremental Schematic Window* and loads the selected component only.

- *Append to IS*

  Selecting this option adds the selected component to *The Incremental Schematic Window*. In this case, existing components remain as it is.

  When a component is added to the *Incremental Schematic* window, only the component icon is shown without connections (for ports) and without pins (for instances). In this case:

  - Double-click the port component to add the connected net and all its end connections (ports or instances).
  - Double-click an instance component to add the instance pins.

NOTE: *When more than one instance of different hierarchies is added or cross-probed to the Incremental Schematic window, parent hierarchies of the instances (up to their common ancestor) also get added to the Incremental Schematic. By default, this feature is turned on and can be turned off by clearing the check box for the Maintain Relative Hierarchies for Instances in IS option from the Schematic Page of the Tools > Preferences menu option.*

Viewing RTL Source Code of an Object

To view the RTL source code of an object, right-click the object and select *Cross-Reference to RTL* option from the shortcut menu. The source code corresponding to the object is highlighted in *HDL Viewer Pane*.

NOTE: *When you right-click on a signal in the Modular Schematic window, this option may appear as Cross-Reference to RTL/Waveform Viewer (instead of Cross-Reference to RTL). This occurs if the selected violation message has a Waveform Viewer associated with it and that signal also appears in that Waveform Viewer.*

Viewing Subgroups

To expand all subgroups of the selected group, right-click on that group and select the *Expand All Sub Groups* option from the shortcut menu. This displays subgroups of the selected group in the schematic window.

To view or hide the subgroups within the selected group, click the *Enable Groups* button ( ) or *Disable Groups* button ( ), respectively, on the schematic toolbar.

Hiding Subgroups

To hide all subgroups of the selected group, right-click on that group and select the *Collapse All Sub Groups* option from the shortcut menu. This hides the subgroups of the selected group in the schematic window.

To view or hide the subgroups within the selected group, click the *Enable Groups* button ( ) or *Disable Groups* button ( ), respectively, on the schematic toolbar.

Viewing Case Analysis Settings on an Object

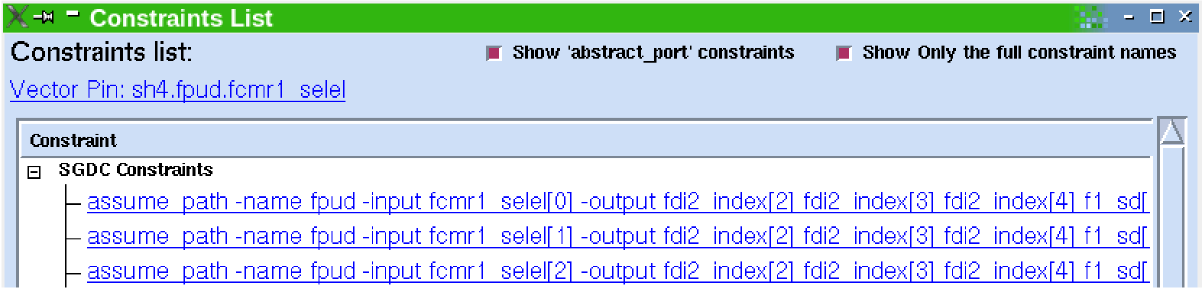
If you have specified case analysis settings on an object by using the `set_case_analysis/testmode` constraints, the object is tagged with a forced value in the schematic.

To view the source of these settings, right-click on the object and select the *Cross-Reference Case Analysis* option from the shortcut menu. This displays the SGDC/ SDC file containing the first found case analysis setting in the *HDL Viewer Pane*. In addition, the line containing the constraint is highlighted in this pane.

Viewing SDC and SGDC Constraints Set on an Object

To view SDC/SGDC constraints set on an object, such as port, net, instance, or terminal, right-click on that object and select *Cross-Reference to Constraints* option from the shortcut menu.

This displays the *Constraints List* dialog showing constraints (SDC/SGDC) set on the object. The following figure shows the *Constraints List* dialog:



Constraints list:

☑ Show 'abstract_port' constraints ☑ Show Only the full constraint names

Vector Pin: sh4.fpud.fcmr1_selel

Constraint

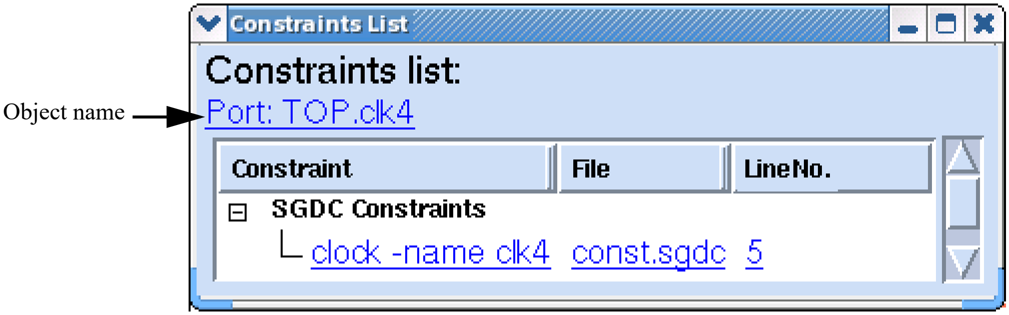
⊟ SGDC Constraints

- assume_path -name fpud -input fcmr1_selel[0] -output fdi2_index[2] fdi2_index[3] fdi2_index[4] f1_sd[
- assume_path -name fpud -input fcmr1_selel[1] -output fdi2_index[2] fdi2_index[3] fdi2_index[4] f1_sd[
- assume_path -name fpud -input fcmr1_selel[2] -output fdi2_index[2] fdi2_index[3] fdi2_index[4] f1_sd[

FIGURE 54. Constraints List-Default View

In the above window, the Constraints column specifies the name of the constraint.

You can view the details of the constraints, such as, file name by switching on the **Show only the full constraint names toggle** switch.



Constraints list:
Port: TOP.clk4

Constraint	File	LineNo.
⊟ SGDC Constraints		
└ clock -name clk4	const.sgdc	5

FIGURE 55. Constraints List

In the above dialog:

- The *Constraint* column specifies the name of the constraint.
- The *File* column specifies the name of the SDC/SGDC file.
- The *Line No.* column specifies the line number in the constraints file where the constraint is defined.
- When you click the name of the object, that object is focused in the *Modular Schematic* window.
- When you click on a row corresponding to a constraint, the corresponding line in the constraints file where the selected constraint is defined is highlighted in *HDL Viewer Pane*.

NOTE: *Please note the following points:*

- *If constraints of a particular type, such as SGDC are not set for an object, they do not appear in the Constraints List dialog.*
- *If no constraints are set on an object, the Cross-Reference to Constraints option appears disabled.*
- *When you right-click a net and select the Cross-Reference to Constraints option from the shortcut menu, you can also see the constraints set on ports of connected nets in addition to the constraints set on the net.*

Viewing Connected Nets

To view nets connected to each bit of a pin bus, port bus, or net bundle, right-click on such an object and select the *List connected nets* option (for pin bus and port bus) or *List Nets* option (for net bundle) from the shortcut menu.

This displays a window listing all nets connected to the pin bus, port bus, or net bundle along with their logic values, if available. For unconnected bits, it is indicated that the bit is hanging.

Viewing Library Cell Instance Information

For instances of SpyGlass-compiled gate library cells, you can view interface and functionality inferred by SpyGlass Library Compiler while compiling gates library.

Right-click on such an instance and select *Cell Information*. A text box appears that lists the cell name, cell type, cell source library (.sglib file), inferred pin interface, and inferred functionality. You can print the displayed information by clicking the *Print* button.

Viewing SpyGlass-Generated Cell Macro Definitions

During synthesis, SpyGlass adds cells from its internal library for inferred objects, such as MUX and select boxes. You can view the corresponding macro definition by right-clicking the macro instance in the Modular Schematic window and then selecting *Show Macro Definition...* from the shortcut menu. A window appears displaying the corresponding macro definition (from the <your-inst-dir>/SPYGLASS_HOME/auxi/target_libs/generic/macro_lib.v* directory), if available.

Cross-Referencing to Power Data

While debugging power-related violations through schematic, you may want to cross-probe to the UPF file in which isolation/level shifter strategies are defined for design elements.

To enable you locate design elements on which such strategies are defined, Atrenta Console displays certain symbols adjacent to these elements in the schematic. For example, notice ▶ symbols representing isolation logic in the following schematic:

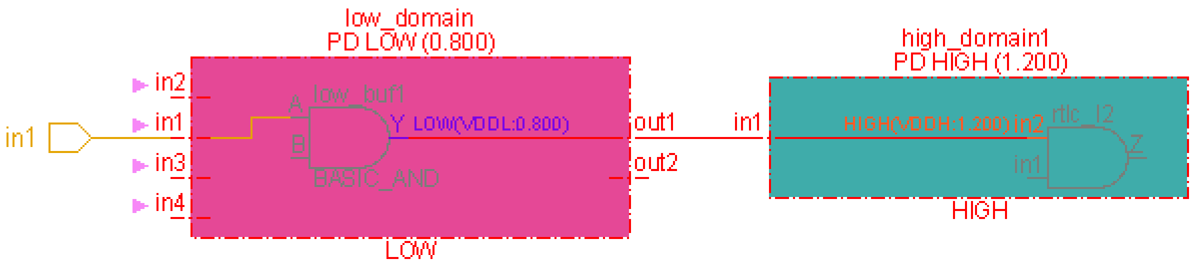


FIGURE 56. Cross-Referencing to Power Data

In the above schematic, the ▶ symbols indicate that isolation logic is applied on the `in1`, `in2`, `in3`, and `in4` pins.

NOTE: *Different symbols appear to represent a particular type of strategy. For example, ▶ appears to represent an isolation logic and ┌ appears to represent a level shifter logic. These symbols appear based on the type of the selected rule violation. For example, if both level shifter and isolation logic is applied on an element, and you a select a rule violation that reports information related to only isolation logic, the ▶ symbol appears adjacent to that element.*

Once you locate such design elements, you can directly cross-probe to the UPF file by right-clicking on an element and selecting the *Cross-Reference to Power Data* option from the shortcut menu.

When you select this option, a sub-menu appears displaying the type of strategy, such as *Isolation* and *Level shifter* that is applied on the selected element. On clicking the *Isolation* or *Level shifter* option from the sub-menu, names of strategies appear in another sub-menu.

For example, consider the following figure:

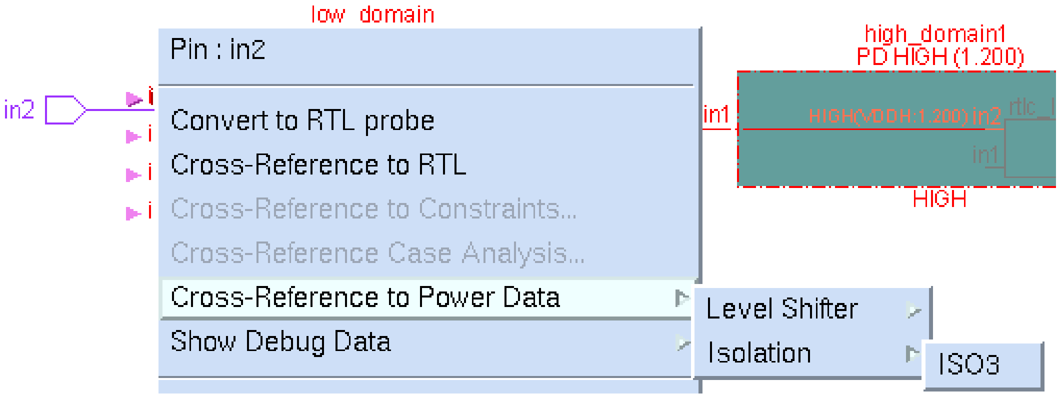


FIGURE 57. Isolation or Level Shifter Options

In the above example, when you click the isolation logic name (`ISO3`) displayed as a sub-menu of the *Isolation* option, Atrenta Console cross-probes to the UPF file where this isolation logic is defined.

Viewing Debug Data

To view the debug data, right-click the object and select any of the following options:

- *Show Debug Data-> DFT*: Select this option to view SpyGlass DFT solution debug data.

NOTE: *SpyGlass supports the back annotation feature for SpyGlass DFT solution.*

- *Show Debug Data-> Power Est*: Select this option to view power data.
- *Show Debug Data-> Clock-reset*: Select this option to view SpyGlass CDC solution debug data.

Show Activity and Probability Annotation

Use this option to annotate activity/probability data on the selected instance.

NOTE: *By default, this option is disabled. This option is enabled only when you run a goal of the SpyGlass Power Family solution.*

For more information, refer to the *Viewing Simulation Data in Schematic* section of the *Power Family Rules Reference Guide*.

Generating Constraints

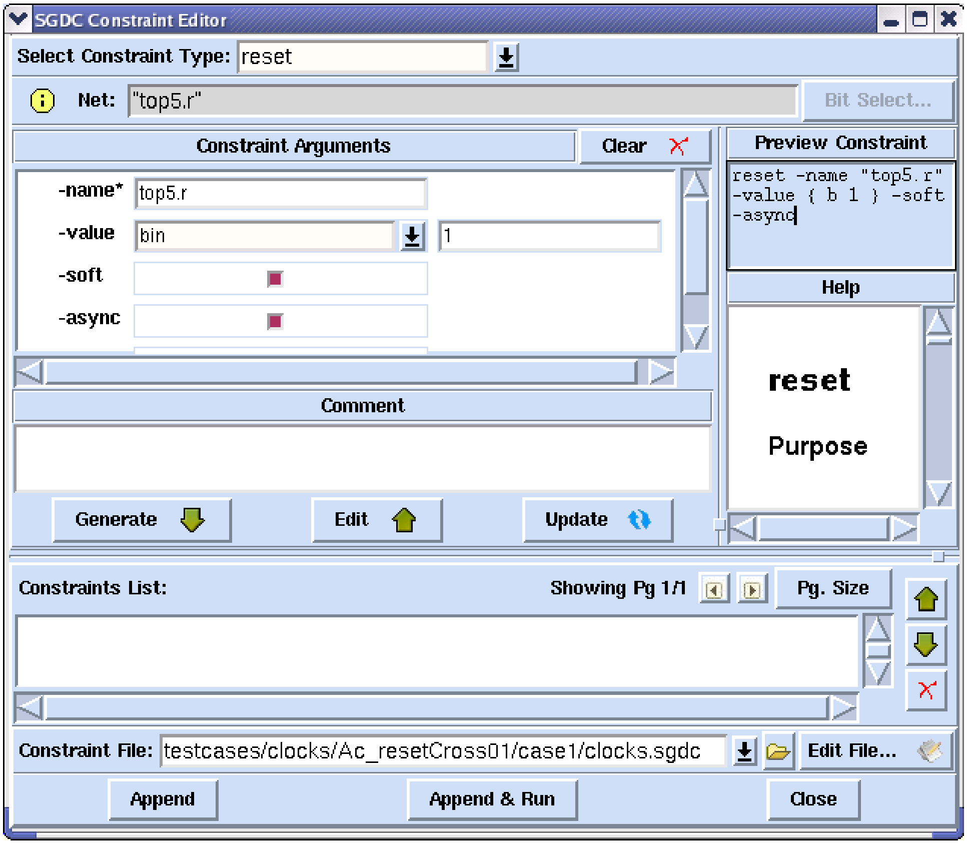
To generate a constraint on an object, such as port, net, and terminal, perform the following steps:

1. Right-click on an object in the schematic, and select the *Set SGDC Constraint* option from the shortcut menu.

   When you select this option, the *SGDC Constraint Editor* window appears.

NOTE: *When you* ***right-click*** *on an instance of a module, the set SGDC constraint is applied for* ***all*** *the instances of a module.*

The following figure displays the *SGDC Constraint Editor* window:



Select Constraint Type: reset

Net: "top5.r"

-name* top5.r

-value bin 1

-soft

-async

Preview Constraint

```
reset -name "top5.r"
-value { b 1 } -soft
-async
```

Constraint File: testcases/clocks/Ac_resetCross01/case1/clocks.sgdc

FIGURE 58. SGDC Constraints Editor

2. Select the constraint you want to generate from the *Select Constraint Type:* drop-down list.

   When you select a constraint, the *Constraint Arguments* section displays various fields for different arguments applicable to that constraint. For example, if you select the reset constraint, the *Constraint Arguments* section displays various fields of different arguments of the reset constraint, such as -name, -value, -soft, etc. In addition, the help of that constraint is displayed in the *Help* section.

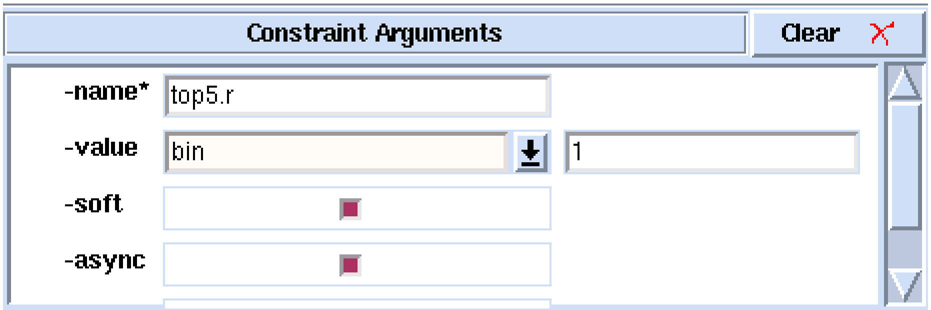
   You can enter the name of the objects in the from, through, or to fields or right click any field and select the last selected object from the *Select Object from MS/Select Object from IS* shortcut menu options.

NOTE: *The* **Set SGDC Constraint** *option is enabled for the instances, except internal instances. When you specify the* `cdc_false_path` *constraint in the* **Select Constraint Type** *field, for such instances, in the* **SGDC Constraint Editor** *window, the name of the master module is populated in the name field.*

NOTE: *You can also add the cdc_false_path constraints for the Ac_unsync01 rule through the spreadsheet viewer.*

3. Specify the argument details in the *Constraint Arguments* section.

   You must specify values for all the mandatory arguments. Such argument names are suffixed with an asterisk (*), as shown in the following figure:



-name* top5.r

-value bin 1

-soft ■

-async ■

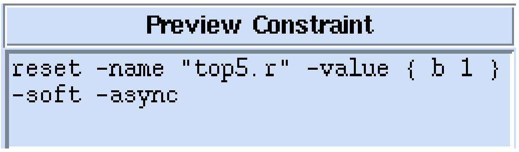
FIGURE 59. Constraint Arguments

In the above case, `-name` is the mandatory argument.

Please note the following points:

- ❒ While specifying argument values, you can simultaneously see the preview of the constraint command being generated in the *Preview Constraint* section.

  An example is shown in the following figure:

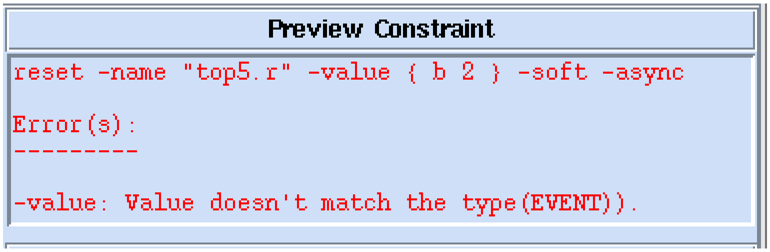


```
reset -name "top5.r" -value { b 1 }
-soft -async
```

FIGURE 60. Preview Constraint

- ❒ If you specify an invalid value in the -value textbox for a particular constraint, Atrenta Console reports an appropriate error message in the *Preview Constraint* section and highlights the text in this section in red.

  For example, if you select the value type as bin, but specify 2 in the -value textbox, Atrenta Console dumps the following information in the *Preview Constraint* section:

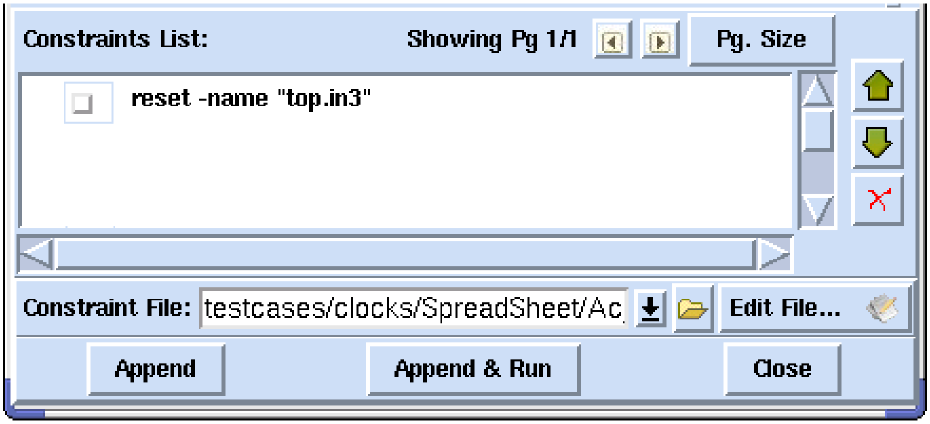


```
reset -name "top5.r" -value { b 2 } -soft -async

Error(s):
----------

-value: Value doesn't match the type(EVENT)).
```

FIGURE 61. Preview Constraint Error Message

4. Click the *Generate* button to generate the specified constraint for the selected object in the schematic.

   The generated constraint is displayed in the *Constraints List* section, as shown in the following figure:



□ reset -name "top.in3"

Constraint File: testcases/clocks/SpreadSheet/Ac

FIGURE 62. Constraints List

NOTE: *If the number of constraints added exceeds the page size, click the (◀ ▶) buttons to go to previous or next pages.*

5. Specify the name of the constraint file in the *Constraint File* textbox. Alternatively, click ⬇ to select the required SGDC file from the drop-down list. If this list does not contain the required file, click 📂 and browse to the directory containing the required file.
6. Click the *Append* button to add the newly generated constraints to the specified SGDC file.
7. Click the *Append & Run* button to apply your settings and run SpyGlass analysis with the new commands.

NOTE: *You need not specify the current design in the SGDC Constraint Editor window. SpyGlass automatically inserts the appropriate current design whenever required.*

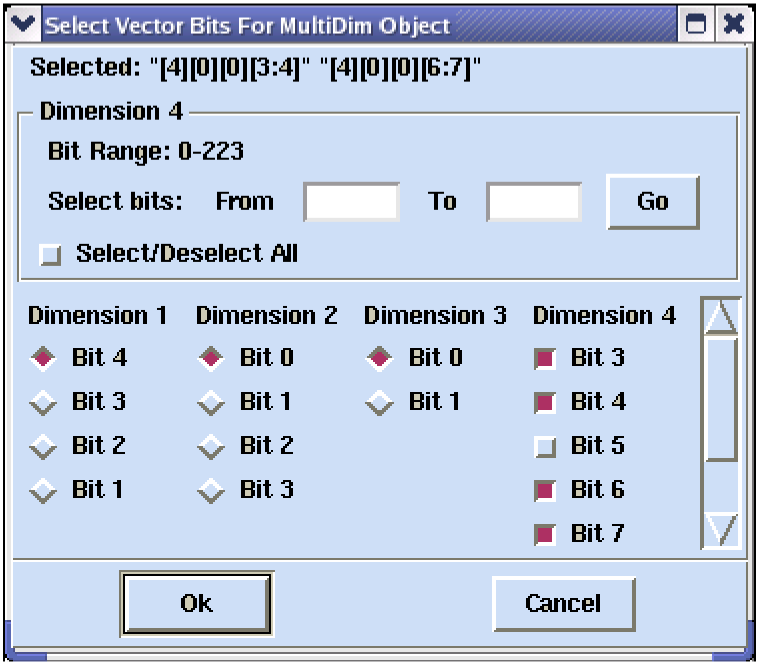
8. Click the *Close* button to close the *SGDC Constraint Editor* window.

Buttons Used in the SGDC Constraint Editor Window

The *SGDC Constraint Editor* window also contains the following buttons:

- *Bit Select*: Enables you select multiple bits for a constraint.

  Consider a scenario in which you want to specify the bit range for a four-dimensional vector net. In this case, when you click the *Bit Select* button, the *Select Vector Bits for MultiDim Object* dialog is displayed, as shown in the following figure:



Select Vector Bits For MultiDim Object

Selected: "[4][0][0][3:4]" "[4][0][0][6:7]"

Dimension 4

Bit Range: 0-223

Select bits: From □ To □

□ Select/Deselect All

Dimension 1	Dimension 2	Dimension 3	Dimension 4
◆ Bit 4	◆ Bit 0	◆ Bit 0	■ Bit 3
◇ Bit 3	◇ Bit 1	◇ Bit 1	■ Bit 4
◇ Bit 2	◇ Bit 2		□ Bit 5
◇ Bit 1	◇ Bit 3		■ Bit 6
			■ Bit 7

FIGURE 63. Select Vector Bits for MultiDim Object

In this dialog, you can specify the bit range by selecting an appropriate bit in each dimension, except the last dimension. Hence, in this case, you can select one bit in each field, *Dimension 1*, *Dimension 2*, and *Dimension 3*. The last dimension (*Dimension 4* in this case) enables you to specify the part select for the vector signal. You can specify the part select by selecting multiple bits in the last dimension. All the selected bits are simultaneously displayed in the *Selected* field.

Once you select the bit range, click the *Ok* button. The specified bit range is appended to the constraint name in the *name* field. You can also specify the bit range by entering the values in the *From* and *To* fields and clicking the *Go* button. When you click the *Go* button, the options for the specified bit range are automatically selected. You need to click the *Ok* button to append the bit range to the constraint name.

- *Clear*: Clears the argument values specified in the *Constraint Arguments* section.
- *Edit*: Enables you to edit the values specified for the selected constraint
- *Update*: Updates the constraint selected in the *Constraints List* section with the details specified in the *Constraint Arguments* section.

NOTE: *When you update the values of a constraint(s), you need to append the constraint(s) to the SGDC file again.*

- ⇧ : Enables you to move the selected constraint up in the list.
- ⇩ : Enables you to move the selected constraint down the list.
- ✗ : Deletes the selected constraint.

Viewing Liberty Files

The schematic displays cells defined in .lib or .sglib files specified by using the *gateslib* or *sglib* commands, respectively.

To view the definition of such cells, right-click on a cell and select the *Open Liberty File* option from the shortcut menu.

This displays the liberty file (.lib file) in a text editor window showing the definition of the selected cell.

NOTE: *SpyGlass opens the text editor specified by the Specify external editor program option in the Preferences dialog.*

The text editor always opens the .lib file. So if the selected cell is specified in a .sglib file, SpyGlass opens its .lib version from which that .sglib is generated.

Modular Schematic Window Menu Bar

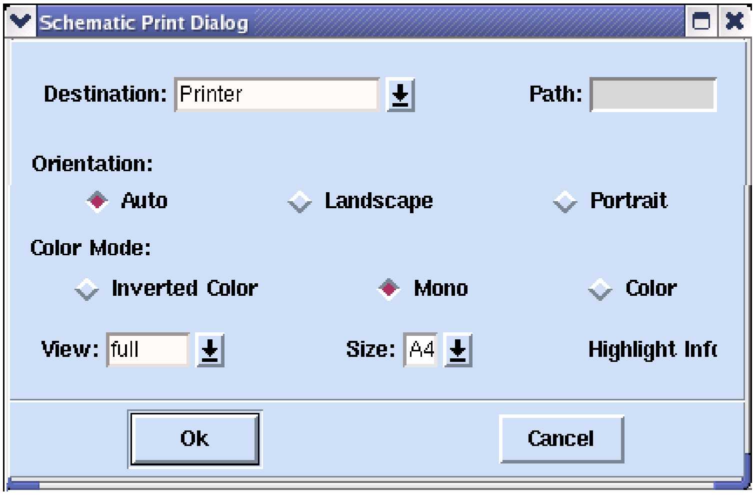
The Modular Schematic window menu bar has the following options:

File > Print	*File > Save As*	*File > Close*
Edit > Undo	*Edit > Redo*	*Edit > Clear All Items*
Edit > Find...	*Edit > Show Case Analysis*	*Edit > Clear All Case Analysis*
Edit > Preferences	*View > Show Schematic Legend*	*View > Show Classic Legend Window*
View > Zoom > In	*View > Zoom > Out*	*View > Zoom > Fit*
View > Pan > Left	*View > Pan > Right*	*View > Pan > Up*
View > Pan > Down	*View > Go To Parent Hierarchy*	

File > Print

Use this menu option to print the schematic or save it as a Postscript file.

When you select this menu option, the Schematic Print dialog appears:



Schematic Print Dialog

Destination: Printer

Path:

Orientation:

◆ Auto ◇ Landscape ◇ Portrait

Color Mode:

◇ Inverted Color ◆ Mono ◇ Color

View: full

Size: A4

Highlight Infc

FIGURE 64. Modular Schematic Window - Schematic Print Dialog

Select or enter as follows:

- *Destination*

  Select any of the following option from the drop-down list:

Option	Description
Printer	Prints the schematic
Postscript File	Saves the schematic as a postscript file
SVG File	Saves the schematic in an SVG (Scalable Vector Graphics) format
EPS File	Saves the schematic in an EPS (Encapsulated PostScript) format

- *Path*

  If you have selected to save the schematic as a Postscript file, enter the full path name of the destination file. Alternatively, click *Browse...* and navigate through the directory structure to set the destination file name. Click Save in the Save As dialog to set the file name.

NOTE: *Clicking Save in Save As dialog does not save the schematic to a postscript file; it only sets the file name. You still need to click OK in the Print Dialog to save the schematic to a postscript file.*

- *Orientation*

  Select *Landscape* to print or save the schematic in a landscape orientation, select *Portrait* to print or save the schematic in a portrait orientation, or select *Auto* to let SpyGlass decide the best-fit orientation.

- *Color Mode*

  Select *Inverted Color* to print or save the schematic in a reverse color mode, select *Mono* to print or save the schematic in monochrome mode, or select *Color* to print or save the schematic in a color mode.

- *View*

  Select *full* to print or save the full schematic or select *current* to print or save the currently displayed view of the schematic.

- *Size*

  Select the target paper size from the supported paper sizes.

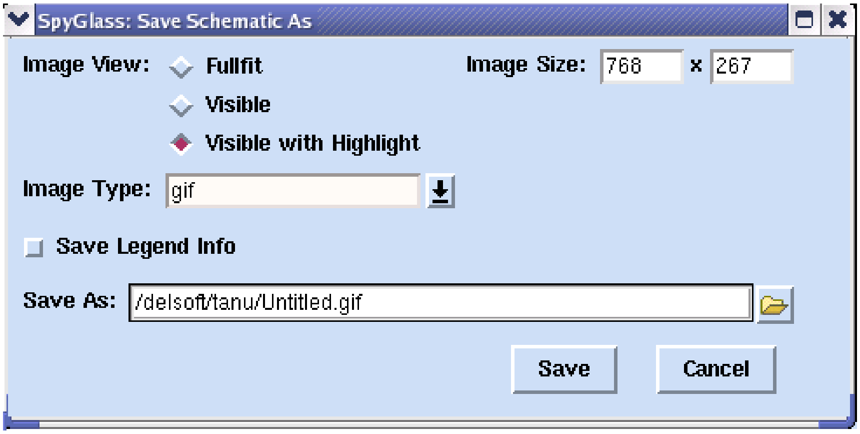
- *Highlight Info*

  If you have selected to print or save the schematic in Color mode, you can set this option to print the highlighted objects with their respective colors.

You can also invoke the Schematic Print dialog by using the *<Ctrl>+P>* key combination on your keyboard or by clicking the (🖨) button on the Modular Schematic toolbar located below the menu bar.

File > Save As

Use this menu option to save the schematic view as an image. When you select this menu option, the Save Schematic As dialog appears as shown in the following figure:



SpyGlass: Save Schematic As

Image View: ○ Fullfit ○ Visible ● Visible with Highlight

Image Size: 768 x 267

Image Type: gif

□ Save Legend Info

Save As: /delsoft/tanu/Untitled.gif

FIGURE 65. Modular Schematic Window - Save Schematic As

Select or enter as follows:

- *Fullfit*

  Select this option to save the complete schematic view as an image.

- *Visible*

  Select this option to save only the visible portion of the schematic as an image.

- *Visible with Highlight*

  Select this option to save the visible portion of the schematic along with the highlighted information as an image.

- *Image Size*

  Enter the size in which you want save the image in the provided text fields.

- *Image Type*

  Click the *Image Type* drop-down list and select the format in which you want to save the image. The available formats are *bmp*, *gif*, *jpeg*, and *png*.

- *Save As*

  Enter the location where you want to save the image in the *Save As* text field. Alternatively, click (📂) and browse to the location where you want to save the image.

Click *Save* to save the image.

You can also open the Save Schematic As dialog by clicking (💾) on the Schematic toolbar.

File > Close

Use this menu option to close the Modular Schematic window.

You can also close the Modular Schematic window by pressing the *<Ctrl>+Q>* key combination on your keyboard.

Edit > Undo

Use this menu option to undo the last probe action in the Modular Schematic window.

You can also undo the last probe action by pressing the *<Ctrl>+<Z>* key combination on your keyboard or by clicking the (↶) button on the Modular Schematic toolbar.

Clicking the (↶) button repeatedly leads to the object from where the probe was initiated.

Edit > Redo

Use this menu option to redo a probe action that was previously undone in the Modular Schematic window.

You can also redo a probe action by pressing the <Ctrl>+<Y> key combination on the keyboard or by clicking the (↷) button on the Modular Schematic toolbar.

NOTE: *The Redo option is enabled only when you have undone a probe action once.*

Edit > Clear All Items

Use this menu option to clear all selections in the Modular Schematic window.

You can also clear all selections by pressing the *<Shift>+C* key combination on your keyboard.

Edit > Find...

Use this menu option to find a net, port, or instance in the schematic.

See *Finding Objects* for more details.

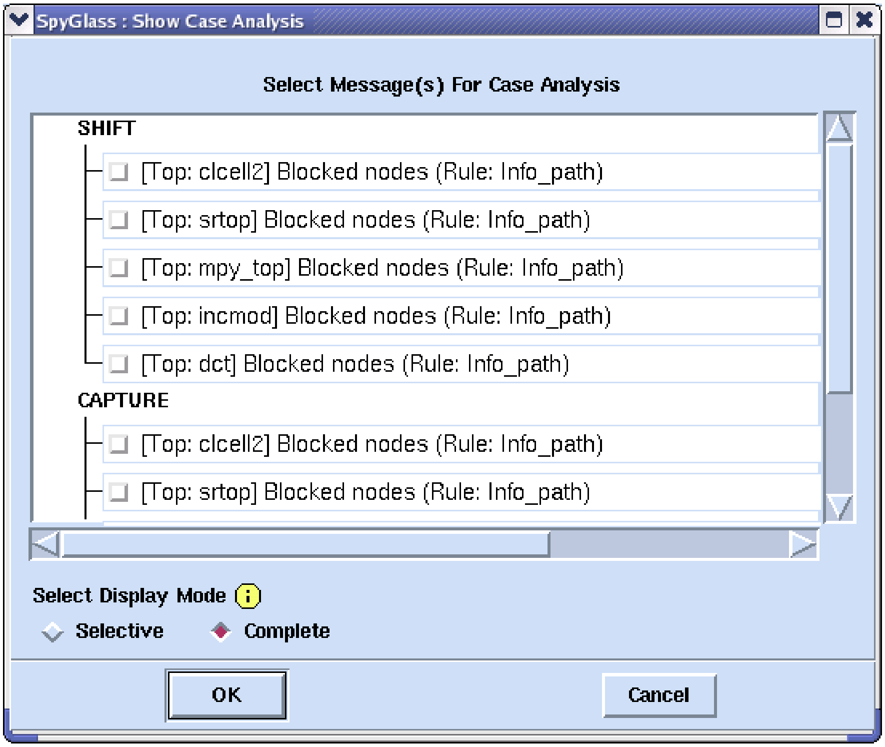
You can also invoke the Find Dialog by pressing the *<Ctrl>+<F>* key combination on the keyboard or by right-clicking anywhere in the Modular Schematic Window and selecting Find... You can also invoke the Find dialog by clicking the (Find) button on the Modular Schematic toolbar.

Edit > Show Case Analysis

Use this menu option to annotate the related show case analysis message (if any) as an auxiliary message over the currently displayed message.

When you select this menu option, a sub-menu appears. Select an appropriate option from this sub-menu based on the type of violations you want to view.

When you select the required option from the sub-menu, the *Show Case Analysis* dialog appears, as shown in the following figure:



SpyGlass : Show Case Analysis

Select Message(s) For Case Analysis

SHIFT

- □ [Top: clcell2] Blocked nodes (Rule: Info_path)
- □ [Top: srtop] Blocked nodes (Rule: Info_path)
- □ [Top: mpy_top] Blocked nodes (Rule: Info_path)
- □ [Top: incmod] Blocked nodes (Rule: Info_path)
- □ [Top: dct] Blocked nodes (Rule: Info_path)

CAPTURE

- □ [Top: clcell2] Blocked nodes (Rule: Info_path)
- □ [Top: srtop] Blocked nodes (Rule: Info_path)

Select Display Mode

○ Selective ● Complete

FIGURE 66. Modular Schematic Window - Show Case Analysis

In the above dialog, you can select the message(s) to be annotated.

NOTE: *You cannot select the SHIFT, CAPTURE, and CAPTURE_ATSPEED mode options together. For example, if you have selected the SHIFT mode options and you try to select the CAPTURE options also, the Conflicting Case Analysis dialog appears prompting you to clear the previously selected options (SHIFT mode options) to enable the selection of CAPTURE mode options.*

You can also press the <*Ctrl*>+*A* key combination on the keyboard to invoke the Show Case Analysis dialog.

Edit > Clear All Case Analysis

Use this menu option to clear all case analysis annotations in the current view.

You can also press the *<Shift>+A* key combination on the keyboard to clear all case analysis annotations.

Edit > Preferences

Use this option to invoke the Preferences dialog. See *Tools > Preferences* for more details.

You can also invoke the Preferences dialog by pressing the *<Ctrl>+<R>* key combination on the keyboard or by clicking the (▤) button on the Modular Schematic toolbar.

View > Show Schematic Legend

Use this option to invoke the Schematic Legend window. This window displays the meaning of highlighted colors and the text attributes applied on the objects in that schematic window.

Once the *Schematic Legend* window is displayed, this menu option changes to *Hide Schematic Legend*. You can select this option to hide the *Schematic Legend* window.

View > Show Classic Legend Window

Use this menu option to invoke The Legend Window.

NOTE: *This menu option is visible only if the Enable Classic Legend option is selected in the Preferences dialog. For details on the Preferences dialog, refer to the section, Tools > Preferences.*

View > Zoom > In

Use this menu option to zoom in the current schematic view.

You can also zoom in by pressing the *<Z>* key on the keyboard or by clicking the (🔍) button on the Modular Schematic window.

View > Zoom > Out

Use this menu option to zoom out from the current schematic view.

You can also zoom out by pressing the <*O*> key on the keyboard or by clicking the (🔍) button on the Modular Schematic toolbar.

View > Zoom > Fit

Use this menu option to display the complete schematic of the current design unit in the Modular Schematic window.

You can also zoom fit by pressing the <*F*> key on the keyboard or by clicking the (🔍) button on the Modular Schematic toolbar.

View > Pan > Left

Use this menu option to pan left in the current schematic view.

You can also pan left by pressing the left arrow key on the keyboard.

View > Pan > Right

Use this menu option to pan right in the current schematic view.

You can also pan right by pressing the right arrow key on the keyboard.

View > Pan > Up

Use this menu option to pan up in the current schematic view.

You can also pan up by pressing the up arrow key on the keyboard.

View > Pan > Down

Use this menu option to pan down in the current schematic view.

You can also pan down by pressing the down arrow key on the keyboard.

View > Go To Parent Hierarchy

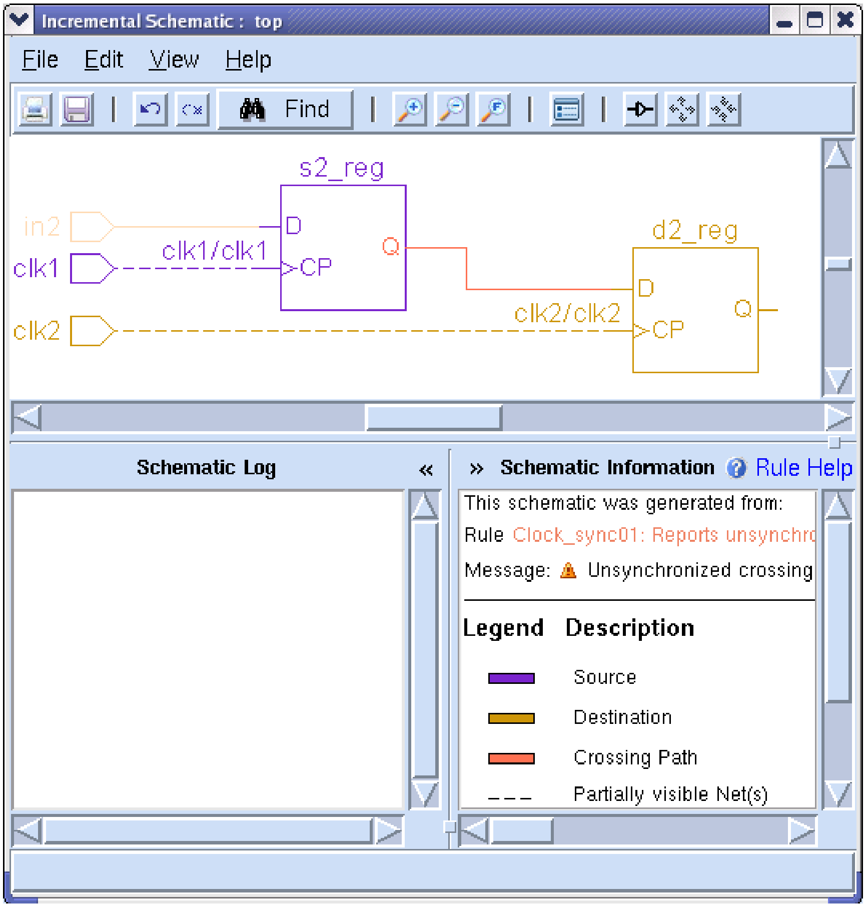
Use this menu option to view the parent hierarchy of the design unit currently displayed in the Modular Schematic window.

You can also view the parent hierarchy by pressing the <*U*> key on the keyboard.

The Incremental Schematic Window

Incremental Schematic window displays the selected portions of (flattened) design schematic across hierarchical boundaries.

To display the *Incremental Schematic* window, double-click on the violation message and click Incremental Sch .in the *Results* pane.



This schematic was generated from:

Rule Clock_sync01: Reports unsynchro

Message: ⚠ Unsynchronized crossing

Legend	Description
🟪	Source
🟨	Destination
🟧	Crossing Path
– – –	Partially visible Net(s)

FIGURE 67. Incremental Schematic Window

The *Incremental Schematic* window is divided into the following sections:

The schematic display

This section displays all components (across hierarchical boundaries) involved in a selected message or a probed component. You can select more than one message at a time and view all components involved in the selected messages.

Menu bar

Refer to the *Incremental Schematic Window Menu Bar* topic.

Toolbar

Toolbar contains various buttons, such as *Print*, *Find*, *Undo*, *Redo*, *Zoom In*, *Zoom Out*, *Zoom Fit*, and *Preferences*.

Schematic Log

This section prints appropriate messages based on the actions performed in the schematic.

Schematic Information

This section displays the meaning of highlighted colors and the text attributes applied on the objects in the schematic window. In addition, if a violation message is selected, it also displays the rule name, short summary of the rule, and the violation message.

You can view the complete rule help by selecting ? Rule Help. This displays the *Rule Help* window containing the complete rule help.

NOTE: *The information visible in this section applies to the first selected message, that is, the primary message. It does not apply to secondary messages.*

Interaction with Other Windows

Interaction of the *Incremental Schematic* window with other windows is same as it is in the *Modular Schematic* window except that the *HDL Viewer* and the *Modular Schematic* windows are updated when a schematic component of another module is selected.

Creating IS Probes

In the Incremental Schematic window, an IS probe is created when:

- A component is loaded from the Modular Schematic window to the Incremental Schematic window. See *Loading or Appending to the Incremental Schematic Window* for details.
- A component in the Incremental Schematic window is explored further by double-clicking (see *Double-Clicking*) or by tracing input or output cones (see *Tracing Cones*).

When an instance is cross-probed into the Incremental Schematic from any other window, such as from RTL, Instance Browser, Power Browser, or Design Tree, the parent hierarchies of the instances are also added to the Incremental Schematic. By default, this feature is enabled and can be disabled by clearing the check box for the *Maintain Relative Hierarchies for Instances in IS* option from the *Schematic Page* of the *Tools > Preferences* menu option.

NOTE: *Atrenta Console does not allow you to cross-probe to the RTL files of encrypted design units. If you try to cross-probe to the RTL of such design units, SpyGlass displays a message in the RTL viewer specifying that the file is encrypted.*

Clearing IS Probes

To clear IS probes, right click anywhere in the *Incremental Schematic* window and select the *Clear All IS Probes* option from the shortcut menu.

Using the Cursor

You can change view in the Incremental Schematic window in the following different ways:

Single-Clicking

Refer to the *Single-Clicking* section in the Modular Schematic window.

Double-Clicking

Double-clicking on different objects displays different results in the schematic, as discussed in the following points:

- Double-clicking on a port or pin schematic component

  When you double-click a port or pin schematic component in the Incremental Schematic window, the schematic is expanded to show the connected net and all its end connections (instances or ports).

  When only one component is connected to the IS Probed port or pin, the connected component is displayed with a solid line connection. However, if there are multiple components connected to the IS Probed port or pin, the first found component is displayed with a dotted line connection. Double-click the dotted line connection and the next found component connected to the IS Probed port or pin is displayed. If it is the only other component connected to the IS Probed port or pin, the component is displayed with a solid line connection. Otherwise, the component is displayed with a dotted line connection. Continue double-clicking the dotted line connection until all connected components are added with solid line connections.

- Double-clicking on an instance schematic component

  When you double-click an instance schematic component in the Incremental Schematic window, the schematic is expanded to show the pins of that instance.

- Double-clicking on a pinbus schematic component

  When you double-click a pin bus schematic component in the Incremental Schematic window, the schematic is expanded to show the probed signals and objects connected to these probed signals.

  Set the *Expand all signals of the Pinbus on Double Clicking* option of the *Preferences* dialog to have the schematic expanded to show all connected nets and the objects connected to these connected nets whenever you double-click a pin bus.

- Double-clicking on an input or output pin/port and partial net

  Based on whether the *Show Full Connectivity on double click* option is selected in the *Schematic Page* of the *Preferences* dialog, the connectivity of input and output port/pin and partial net is displayed accordingly, as discussed in the following table:

Action performed	**Show Full connectivity on double click option selected**	**Show Full connectivity on double click option NOT selected**
Double-click on input pin/port	All the instances driving that pin get loaded in the schematic.	Only one instance driving that pin is loaded. Double-click on that pin/port again to load another instance driving that pin.
Double-click on output pin/port	All the instances driven by that pin get loaded.	Only one instance driven by that pin is loaded. Double-click on that pin/port again to load another instance driven by that pin.
Double-click on partial net	All the loads and drivers of that net, that is its entire connectivity within that module is loaded.	Single load or a single driver is loaded for that instance. Double-click on that net again to load another load/driver of that net.

Please note the following points:

- Double-clicking works as a toggle. For example, when you double-click an instance that is currently showing only the pins involved in the probe, the schematic is expanded to show the pins of that instance. Now, double-clicking again will bring back the original view. Corresponding messages are displayed in the Log section of the Incremental Schematic window.
- When one or more pins of an instance have case analysis settings and you have selected the *View > Show Case Analysis* menu option, double-clicking on the instance for the first time expands the schematic to show the currently showing pins and those pins with case analysis settings. Double-clicking again expands the schematic to display all pins of the instance and their case analysis settings, if any. Double-clicking the third time returns the schematic to the original display. Corresponding messages are displayed in the Log section of the Incremental Schematic window.

Click-Drag Combinations

Refer to the *Click-Drag Combinations* section in the Modular Schematic window.

Right-Clicking

Use the right-click menu options in the *Incremental Schematic* window to perform the following actions:

Viewing the Object Name

For details on this option, see *Viewing Object Name*.

RTL Probing

RTL probe (or normal probe) is a process in which when you click an object in the schematic, the following actions occur:

- The object is cross-probed to other windows, such as *HDL Viewer Pane*.
- The corresponding entry is added to the *Legend Window*.

However, in some cases, you may need to locate an object first in the *Incremental Schematic* window before creating RTL probe for that object. In such cases, perform the following steps:

1. Create IS probe for pins (by double-clicking the pins) to add the objects connected to the pins in the *Incremental Schematic* window.

   For details, see *Creating IS Probes*.

NOTE: *IS probes do not update other Atrenta Console windows.*

2. When the desired object is found in the *Incremental Schematic* window after multiple IS probes, convert the IS probe into an RTL probe by right-clicking the desired object and selecting the *Convert to RTL probe* option from the shortcut menu.

After performing the above steps, the object is cross-probed across other Atrenta Console windows and the corresponding entry is added to the Legend window.

NOTE: *Please note the following points:*

- *If you clear all IS probes then only RTL probes are retained in the Incremental Schematic window. See Clearing IS Probes.*

🗎 *When you right-click on a signal in the Incremental Schematic window, this option may appear as Convert to RTL probe/Waveform Viewer (instead of Convert to RTL probe). This occurs if the selected violation message has a Waveform Viewer associated with it and that signal appears in the Waveform Viewer.*

🗎 *Atrenta Console does not allow you to cross-probe to the RTL of encrypted design units. If you try to cross-probe to RTL of such design units, SpyGlass displays a message in the RTL viewer specifying that the file is encrypted.*

Viewing the RTL Source Code of an Object

To view the RTL source code of an object, right-click on the object and select the *Cross-Reference to RTL* option from the shortcut menu. The source code corresponding to the object is highlighted in *HDL Viewer Pane*.

NOTE: *When you right-click on a signal in the Incremental Schematic window, this option may appear as Cross-Reference to RTL/Waveform Viewer (instead of Cross-Reference to RTL). This occurs if the selected violation message has a Waveform Viewer associated with it and that signal appears in that Waveform Viewer.*

Viewing Case Analysis Settings on an Object

For details on this option, see *Viewing Case Analysis Settings on an Object*.

Viewing SDC/SGDC Constraints Set on an Object

For details on this option, see *Viewing SDC and SGDC Constraints Set on an Object*.

Cross-Referencing to Power Data

For details on this option, see *Cross-Referencing to Power Data*.

Viewing Debug Data

To view debug data, right-click on an object and select any of the following options based on your requirement:

- *Show Debug Data-> DFT* option to view SpyGlass DFT solution debug data.

NOTE: *SpyGlass supports the back annotation feature for SpyGlass DFT solution.*

- *Show Debug Data-> Power Est* option to view power data

- *Show Debug Data-> Clock-reset* option to view SpyGlass CDC solution data

Viewing Object Properties

For details on this option, see *Viewing Object Properties*.

Show Activity and Probability Annotation

Use this option to annotate activity/probability data on the selected instance.

NOTE: *By default, this option is disabled. This option is enabled only when you run a goal of the SpyGlass Power Family solution.*

For more information, refer to the *Viewing Simulation Data in Schematic* section of the *Power Family Rules Reference Guide*.

Viewing/Tracing Connected Nets

To view/trace the nets connected to each bit of a pin bus, net bundle, or port bus, right-click on the net and select the *List/Trace Connected Nets* option from the shortcut menu.

A dialog appears listing all nets connected to the bits of a pin bus, net bundle, and port bus. For unconnected bits, it is indicated that the bit is hanging.

The *List/Trace Connected Nets* dialog automatically appears if there are no probed signals and you double-click a pin bus, net bundle, or port bus.

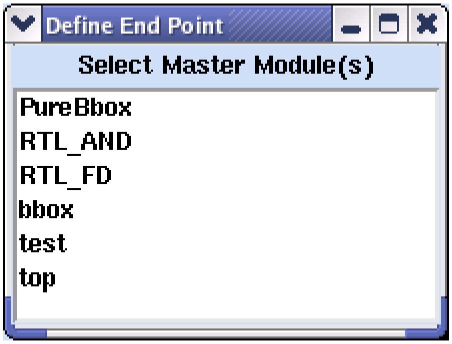
You can also set the *Show List/Trace Connected Nets Dialog on Double Clicking* option in the *Schematic Page* of the *Tools > Preferences* menu option to have the *List/Trace Connected Nets* dialog appear whenever you double-click a pin bus, net bundle, or port bus.

Tracing Cones

You can trace the input cone (for input or inout pins) or output cone (for output or inout pins) of any pin in the schematic portion displayed in the *Incremental Schematic* window.

To trace the cone, right-click on a pin and select any of the options from the shortcut menu:

- *Show Input Cone > To Primary Inputs* to view all components in the fan-in cone of the selected pin up to primary inputs.
- *Show Input Cone > To Flops* to view all components in the fan-in cone of the selected pin up to flip-flops.
- *Show Input Cone > To Flops/Latches* to view all components in the fan-in cone of the selected pin up to flip-flops and latches.
- *Show Input Cone > Define End Point...* to set the end points (instances) up to which you want to view the fan-in components. Then, the *Define End Point* dialog appears listing all the master modules instantiated in the design:



Define End Point

Select Master Module(s)

PureBbox
RTL_AND
RTL_FD
bbox
test
top

FIGURE 68. Define Endpoint Dialog

Select the master module(s) and then click *Show Cone*. The *Incremental Schematic* window is updated to show all components in the fan-in of the selected pin up the instances of the selected master module(s). You can search for an instance by entering its name in the *Find Text* field and clicking *Search*. As you type the starting letters of a valid module name, the results are updated accordingly.

- Select *Show Input Cone > To Module Boundary* to display the input cone till the boundary of the current design unit in which the cone is being seen.

The procedure for tracing output cones for output or inout pins is same using the *Show Output Cone* option in the shortcut menu.

You can undo the last Show Cone actions using the *Undo Show Cone* option in the shortcut menu.

Highlighting Path for the Selected Objects

Perform the following actions to highlight the path between the selected objects:

1. Select and right-click on an object in the *Incremental Schematic.*
2. Click Show Path from the right-click menu.

   A sub-menu is displayed as shown in the following figure:

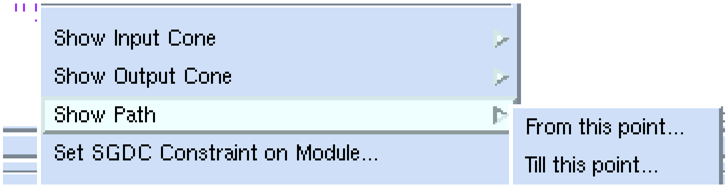


FIGURE 69. Show Path

3. Click **From this point** to make the selected object as the source object.
4. Click **Till this point** to make the selected object as the destination object.
5. Click **Go** on the **Show Path** toolbar to highlight the path for the selected objects.

NOTE: *You specify hierarchical names of instance/pin or port as an input to the* ***From*** *and* ***To*** *fields. However, you can not specify net names.*

For more information on highlighting the paths for the selected objects, see *Highlighting Path Between Two Reference Points*.

Setting SGDC Constraints

For details on this option, see *Generating Constraints*.

Searching the Incremental Schematic Window

For details on this option, see *Finding Objects*.

Viewing Library Cell Instance Information

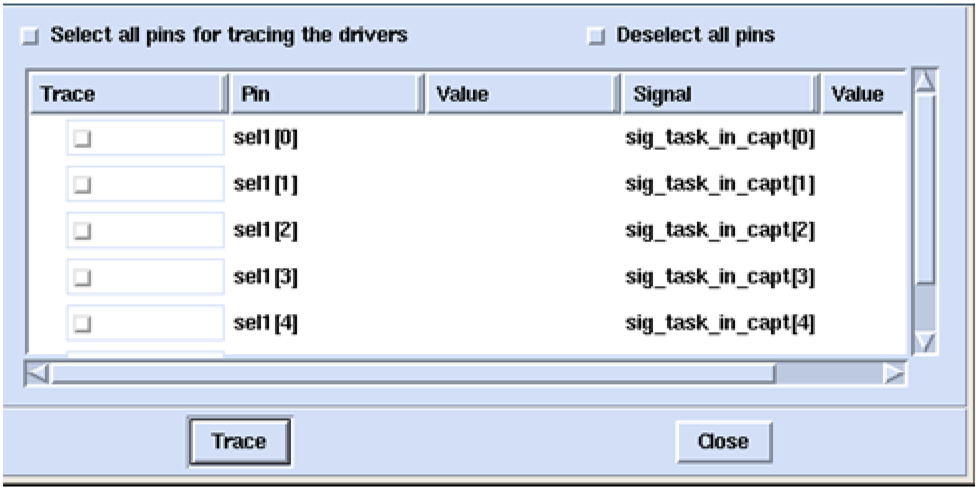
For details on this option, see *Viewing Library Cell Instance Information*.

Tracing the Driver of an Input Pin

In large designs, the driver of an input pin can be present quite far from the pin. In such cases, to view the driver of such input pins, right-click on the required pin and select the *Trace to Driver* option from the shortcut menu.

When you select this option, the pin driver is shown at the centre of the schematic.

In case if you select a pin that is a bus, a dialog appears in which all signals associated with that pin are shown. The following figure shows such dialog:



☐ Select all pins for tracing the drivers ☐ Deselect all pins

Trace	Pin	Value	Signal	Value
☐	sel1 [0]		sig_task_in_capt[0]	
☐	sel1 [1]		sig_task_in_capt[1]	
☐	sel1 [2]		sig_task_in_capt[2]	
☐	sel1 [3]		sig_task_in_capt[3]	
☐	sel1 [4]		sig_task_in_capt[4]	

FIGURE 70. Trace to Driver Option

In the above dialog, select the required pin to bring its driver into view.

If you select multiple pins, driver of the first selected pin is shown in the schematic.

If a pin has multiple drivers, the first driver appears in the schematic.

Viewing Liberty Files

For details on this option, see *Viewing Liberty Files*.

Highlighting Path Between Two Reference Points

To highlight the path between two reference points, for debugging, in the Incremental Schematic, click the **Path** button in the Incremental Schematic toolbar as shown in the *Figure 71*:

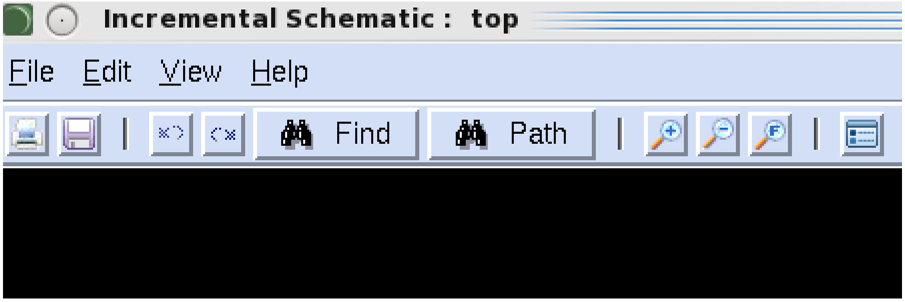
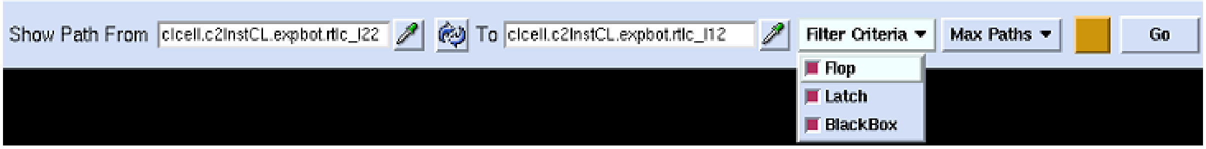


FIGURE 71. Path Button

Clicking the Path button displays the expanded Show Path toolbar as shown in *Figure 72*:


Show Path From clcell.c2InstCL.expbot.rtlc_l22

To clcell.c2InstCL.expbot.rtlc_l12

FIGURE 72. Show Path Toolbar

The Show Path toolbar has two editable fields, namely, **From** and **To**, to input the hierarchical names of the objects. You can either type in the object name or use the dropper button to pick the name of the object from Incremental Schematic.

NOTE: *You specify hierarchical names of instance/pin or port as an input to the* ***From*** *and* ***To*** *fields. However, you can not specify net names.*

However, if no path is found for the selected objects, an error message is displayed.

Also, if wrong hierarchial name is provided for the object, an error message is displayed.

This feature is violation independent. That is, the path for the specified objects is searched throughout the synthesized design, independent of whether the object is part of the violation or not.

You can perform the following actions from the Show Path toolbar:

- *Specify the Filter Criteria*
- *Display One or All Paths*
- *Change Color of the Displayed Path*

Specify the Filter Criteria

You can select one or more of the following options from the **Filter Criteria** drop-down list to neglect the paths containing the specified option from being displayed:

- Flop
- Latch
- BlackBox

Therefore, if you have selected Flop in the Filter Criteria drop-down list, then any path that encounters a flop will not be explored further and will not be displayed. Same behavior is applicable to other options in the Filter Criteria drop-down list.

Alternatively, to highlight the path between the selected objects using the right-click menu, see *Highlighting Path for the Selected Objects*.

Display One or All Paths

You can display single or all paths available in the design for the selected objects using the **Max Paths** drop-down list.

Select **One** from the drop-down list to display only one path for the selected objects.

Select **All** from the drop-down list to display all the paths available for the selected objects.

NOTE: *Selecting All to search for all the paths can be time-consuming for larger designs.*

Change Color of the Displayed Path

Click the Color button adjacent to the Max Paths drown-down list to display the **Show Color** dialog box. You can select any color from the color palette to highlight the selected path with that color.

Incremental Schematic Window Menu Bar

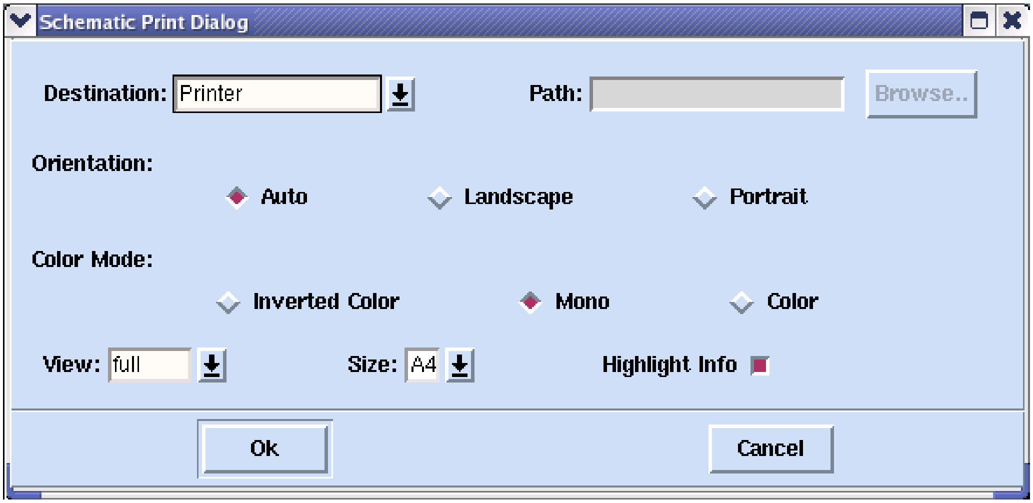
The Incremental Schematic window menu bar has the following options:

File > Print	*File > Close*	*Edit > Undo*
Edit > Redo	*Edit > Clear All Items*	*Edit > Find...*
Edit > Show Case Analysis	*Edit > Clear All Case Analysis*	*Edit > Preferences*
View > Show Schematic Legend	*View > Show Classic Legend Window*	*View > Zoom > In*
View > Zoom > Out	*View > Zoom > Fit*	*View > Pan Left*
View > Pan > Right	*View > Pan > Up*	*View > Pan > Down*
View > Pan > Down		

File > Print

Use this menu option to print the schematic or save it as a Postscript file.

When you select this menu option, the Schematic Print dialog appears:



Schematic Print Dialog

Destination: Printer

Path:

Orientation: ● Auto ○ Landscape ○ Portrait

Color Mode: ○ Inverted Color ● Mono ○ Color

View: full

Size: A4

Highlight Info ■

FIGURE 73. Incremental Schematic Window - Schematic Print Dialog

Select or enter as follows:

- *Destination*

  Select any of the following option from the drop-down list:

Option	Description
Printer	Prints the schematic
Postscript File	Saves the schematic as a postscript file
SVG File	Saves the schematic in an SVG (Scalable Vector Graphics) format
EPS File	Saves the schematic in an EPS (Encapsulated PostScript) format

- *Path*

  If you save the schematic as a postscript file, enter a full path name of the destination file. Alternatively, click *Browse...* and navigate through the directory structure to set the destination file name. Click Save in the Save As dialog to set the file name.

NOTE: *Clicking Save in the Save As dialog does not save the schematic to a postscript file; it only sets the file name. You still need to click Ok in the Print Dialog to save the schematic to a postscript file.*

- *Orientation*

  Select *Landscape* to print or save the schematic in a landscape orientation, select *Portrait* to print or save the schematic in a portrait orientation, or select *Auto* to let SpyGlass decide the best-fit orientation.

- *Color Mode*

  Select *Inverted Color* to print or save the schematic in a reverse color mode, select *Mono* to print or save the schematic in monochrome mode, or select *Color* to print or save the schematic in a color mode.

- *View*

  Select *full* to print or save the full schematic or select *current* to print or save the currently displayed view of the schematic.

- *Size*

  Select the target paper size from the supported paper sizes.

- *Highlight Info*

  If you have selected to print or save the schematic in Color mode, you can set this option to print the highlighted objects with their respective colors.

You can also invoke the Schematic Print dialog by using the *<Ctrl>+P* key combination on your keyboard or by clicking the (🖶) button on the Incremental Schematic toolbar located below the menu bar.

File > Close

Use this menu option to close the Incremental Schematic window.

You can also close the Incremental Schematic window by pressing the *<Ctrl>+<Q>* key combination on the keyboard.

Edit > Undo

Use this menu option to undo the last probe action in the Incremental Schematic window.

You can also undo the last probe action by pressing the *<Ctrl>+<Z>* key combination on your keyboard or by clicking the (↶) button on the Incremental Schematic toolbar.

Clicking the (↶) button repeatedly leads to the object from where the probe was initiated.

Edit > Redo

Use this menu option to redo a probe action that was previously undone in the Incremental Schematic window.

You can also redo a probe action by pressing the <Ctrl>+<Y> key combination on the keyboard or by clicking the (↷) button on the Incremental Schematic toolbar.

NOTE: *The Redo option is enabled only when you have undone a probe action once.*

Edit > Clear All Items

Use this menu option to clear the Incremental Schematic window.

You can also clear the Incremental Schematic window by pressing the

<Shift+<C> key combination on the keyboard or by right-clicking anywhere in the window to get the shortcut menu and select *Clear Window* to remove all displayed components.

Edit > Find...

Use this menu option to find a net, port, or instance in the schematic.

See *Searching the Incremental Schematic Window* for more details.

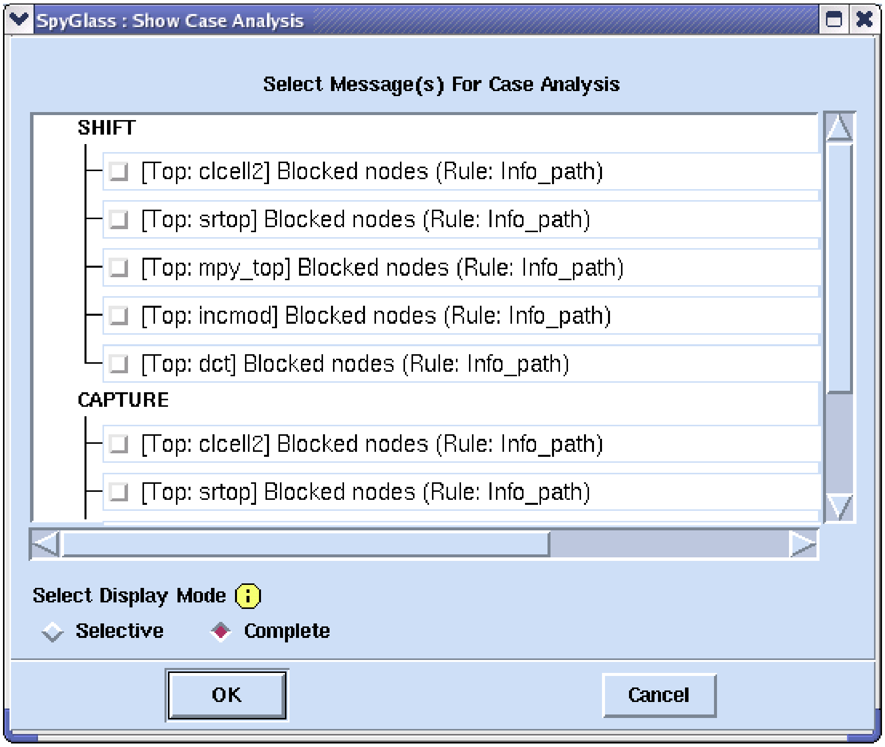
You can also invoke the Find Dialog by pressing the *<Ctrl>+<F>* key combination on the keyboard or by right-clicking anywhere in the Incremental Schematic Window and selecting Find... You can also invoke the Find dialog by clicking the (Find) button on the Modular Schematic toolbar.

Edit > Show Case Analysis

Use this menu option to annotate the related show case analysis message (if any) as an auxiliary message over the currently displayed message.

When you select this menu option, a sub-menu appears. Select an appropriate option from this sub-menu based on the type of violations you want to view.

When you select the required option from the sub-menu, the *Show Case Analysis* dialog appears, as shown in the following figure:



SpyGlass : Show Case Analysis

Select Message(s) For Case Analysis

SHIFT
- □ [Top: clcell2] Blocked nodes (Rule: Info_path)
- □ [Top: srtop] Blocked nodes (Rule: Info_path)
- □ [Top: mpy_top] Blocked nodes (Rule: Info_path)
- □ [Top: incmod] Blocked nodes (Rule: Info_path)
- □ [Top: dct] Blocked nodes (Rule: Info_path)

CAPTURE
- □ [Top: clcell2] Blocked nodes (Rule: Info_path)
- □ [Top: srtop] Blocked nodes (Rule: Info_path)

Select Display Mode

○ Selective ● Complete

FIGURE 74. Incremental Schematic Window - Show Case Analysis

In the above dialog, you can select the message(s) to be annotated.

NOTE: *You cannot select the SHIFT and CAPTURE mode options together. For example, if you have selected the SHIFT mode options and you try to select the CAPTURE options also, the Conflicting Case Analysis dialog appears prompting you to clear the previously selected options (SHIFT mode options) to enable the selection of CAPTURE mode options.*

You can also press the <*Ctrl*>+*A* key combination on the keyboard to invoke the dialog.

Edit > Clear All Case Analysis

Use this menu option to clear all case analysis annotations in the current view.

You can also press the *<Shift>+A* key combination on the keyboard to clear all case analysis annotations.

Edit > Preferences

Use this option to invoke the Preferences dialog. See *Tools > Preferences* for more details.

You can also invoke the Preferences dialog by pressing the *<Ctrl>+<R>* key combination on the keyboard or by clicking the ( ) button on the Modular Schematic toolbar.

View > Show Schematic Legend

Use this option to invoke the Schematic Legend window. This window displays the meaning of highlighted colors and the text attributes applied on the objects in that schematic window.

Once the *Schematic Legend* window is displayed, this menu option changes to *Hide Schematic Legend*. You can select this option to hide the *Schematic Legend* window.

View > Show Classic Legend Window

Use this menu option to invoke the *Legend Window*.

NOTE: *This menu option is visible only if the Enable Classic Legend option is selected in the Preferences dialog. For details on the Preferences dialog, refer to the section, Tools > Preferences.*

View > Zoom > In

Use this menu option to zoom in the current schematic view.

You can also zoom in by pressing the *<Z>* key on the keyboard or by clicking the ( ) button on the Incremental Schematic window

View > Zoom > Out

Use this menu option to zoom out from the current schematic view.

You can also zoom out by pressing the <*O*> key on the keyboard or by clicking the () button on the Incremental Schematic toolbar.

View > Zoom > Fit

Use this menu option to display the complete schematic of the current design unit in the Incremental Schematic window.

You can also zoom fit by pressing the <*F*> key on the keyboard or by clicking the (🔍) button on the Incremental Schematic toolbar.

View > Pan Left

Use this menu option to pan left in the current schematic view.

You can also pan left by pressing the left arrow key on the keyboard.

View > Pan > Right

Use this menu option to pan right in the current schematic view.

You can also pan right by pressing the right arrow key on the keyboard.

View > Pan > Up

Use this menu option to pan up in the current schematic view.

You can also pan up by pressing the up arrow key on the keyboard.

View > Pan > Down

Use this menu option to pan down in the current schematic view.

You can also pan down by pressing the down arrow key on the keyboard.

The Session Log Window

The Session Log window displays log information while the SpyGlass analysis is running. This log reports when SpyGlass enters and completes the different phases of its analysis.

The Session Log window also displays error messages that result when SpyGlass aborts its run.

The log data is appended in the Session Log window for each run.

To save log data displayed in the Session Log window as a text file, right-click on the Session-Log window and select the *Save...* option from the shortcut menu. When you select this option, the Save As dialog appears in which you can specify the file name by which you want to save the data in the *File name* text field and click *Save*.

To clear the session log data, right-click on the Session Log window and select the *Clear Session Log* shortcut menu option.

After the SpyGlass analysis is completed, the log information is saved as the <prj-name>.out file in the current directory where <prj-name> is the name of the project file. When you open a project file in Atrenta Console, the contents of the corresponding <prj-name>.out file (if found) are displayed on the Session Log window. If the corresponding <prj-name>.out file is not found, the SpyGlass version and other information is listed in the Session Log window.

Waveform Viewer Window

To debug complex bugs in your design, viewing message, RTL, schematic, and FSM may not be sufficient to find the exact cause of bugs. For such cases, you can use the Waveform Viewer window.

Waveform Viewer is an analysis tool that enables you to find the root cause of a functional bug in the design. This tool illustrates a sequence of events leading to the functional problem. These events are illustrated from an initial state of a design/sub-design to the time when the bug appears. These events can be generated as a set of simulation vectors in the VCD format. Each event or time frame in VCD corresponds to an edge of a clock relevant to the violation. The waveform viewer launched for a failure contains this VCD content.

The presence of a waveform is indicated by the waveform icon, , in front of the violation message. To view the waveform for the violation, click the button in the toolbar. This displays the *Waveform Viewer* window, as shown in the following figure:

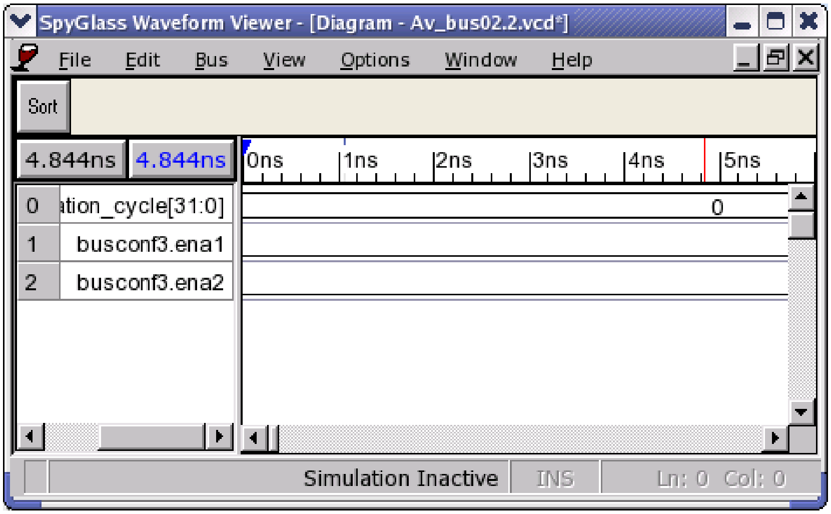


FIGURE 75. Waveform Viewer Window

Initially, SpyGlass loads a small set of signals in the waveform viewer. These signals are believed to be a good start for the debugging process.

As the debugging progresses, you can gradually load more signals in any of the following ways:

- By using cross-probing capabilities

  By looking at loaded signals, RTL code, and schematic, if you can identify relevant signals, values of which may unveil the cause of a bug, click on such signals under RTL/schematic to load the waveform of the signal and analyze the transitions.

  For details, refer to the *Cross-Probing in Waveform Viewer* topic.

- By selecting the fan-in option from the right-click menu displayed for the selected signal in the waveform viewer

  This action loads signals in the fan-in cone of the selected signal for which a waveform is available. This is limited to one signal in any path in the fan-in cone (the first signal visited in the path for which a waveform is available). By default, a waveform is only generated for registers and primary inputs in the cone of influence of a violation. You can use the watchpoint constraint to generate waveform for intermediate signals.

- By selecting the fan-in cone option from the right-click menu displayed for the selected signal in the waveform viewer

  This action reports total number of signals in the cone headed by selected signal as well as all the signals names. Signals for which waveforms are available are highlighted; by clicking on these signals, you can load the corresponding waveform.

Following are main features and capabilities provided by SpyGlass for easy debugging of a functional violation:

- Signals ordering

  The signals appear in topological ordering by default. You can reorder signals in alphabetical order and revert using provided buttons added for this purpose.

- Moving signals around

  You can select a signal(s) and move it to a different location in the waveform viewer. To do this, select the signal(s) that you want to move and place the cursor at the signal boundary. Next, drag the signal(s) to the required position. The automatic ordering feature can be used to restore the original ordering or alphabetical ordering.

- Saving and loading context

  A filter file can be saved which will include all signals visualized at the time of saving. This filter can be loaded at any time during the same debug session or in an ulterior debug session.

- An internal signal, autoverify_state[0:31], is always generated that changes value each time the design transitions from one state to another state (at least one register changes value). When large numbers of signals are loaded into the waveform viewer, this signal can be used to track any changes caused by a previous transition by adding a vertical marker on the next autoverify_state transition and scrolling through all signals in the waveform viewer.
- By default, single selection mode is used in waveform viewer. However, you can select/deselect multiple signals by using Ctrl+click.
- Cross-probing a signal for which an escaped name is specified

  You can specify an escaping style, Verilog or VHDL, to display escaped names in the waveform viewer. To specify this style, select the *View -> Set Signal Escaping Style* menu option. A sub-menu appears displaying the *Verilog* and *VHDL* options. Select the required option to set an escaping style.

  By default, the escaping style is set to *Verilog*.

  When you select the escaping mode as *VHDL*, some extra backslashes appear before the file names appearing in the *Waveform Viewer* window.

Given all the above capabilities and user knowledge of the design, the root cause of a bug can be identified. However, if more analysis is required, the generated VCD file can be analyzed under your preferred simulation environment and more simulation can be performed to analyze further the context of the failure. Alternatively, you may add new assertions and analyze them to understand further the cause of a given failure.

While techniques to identify the cause of a violation are the same, no matter if that violation is due to a RTL bug, erroneous assertion, or missing constraints, you may focus on identifying the category of the violation to find a quick answer to this question. For this purpose, instead of analyzing the failure through tracing signals in an intelligent way (selecting signals believed relevant from RTL/ schematic) or in topological order, you may focus on some set of signals that are prune to cause constraint-related violations:

- Reset: A reset not recognized in a design will participate in falsifying an assertion. By a quick look at the reset signal waveform, you may determine that a reset is making an assertion fail. To fix this problem provide the reset as a constraint in the SpyGlass Auto Verify Design Constraints file.

- Clock: A wrong clock waveform can cause a violation; furthermore, wrong relative clocks frequencies in multi-clock system can cause a violation. By examining the clocks waveforms, you may be able to find such problems. To fix clock definition related problem you can provide the correct definition of the clock in the constraint file.
- Other special ports: Ports such as test mode should be properly constrained. By looking into the waveform signals, the failure may be attributed to such signals.
- Initial state: A wrong initial state may cause a false violation. Specifically, if registers are left free (not assigned to neither a "0" nor "1"), SpyGlass Auto Verify solution will assign them to appropriate values to make assertions fail. By looking into registers file report ".reg" file and/or the waveform one can quickly determine if a violation is caused by such un-initialized registers.

Cross-Probing in Waveform Viewer

SpyGlass provides the following cross-probing capabilities for Waveform Viewer:

- RTL to waveform probing

  Clicking on a signal in the RTL code results in loading (if it is not already loaded) and highlight of the corresponding waveform, if a waveform for the signal is available. The waveform for the given signal is placed in its position according to a predefined ordering (alphabetical or topological). This action is performed if a waveform window is already open. Note that the same action (right-click on RTL signal) also highlights the corresponding schematic wire if a schematic window is already open.
- Schematic to waveform probing

  Clicking on a signal in the schematic window loads and highlights the corresponding waveform. This is similar to RTL to waveform probing.
- Waveform to RTL and schematic probing

  Clicking on a signal in the waveform viewer results in corresponding signals highlight in the RTL code and schematic viewer.

Right-Click Menu Options of the Waveform Viewer Window

When you right-click on any signal in the *Waveform Viewer* window, SpyGlass displays a shortcut menu. Various options of this shortcut menu are described in the following table:

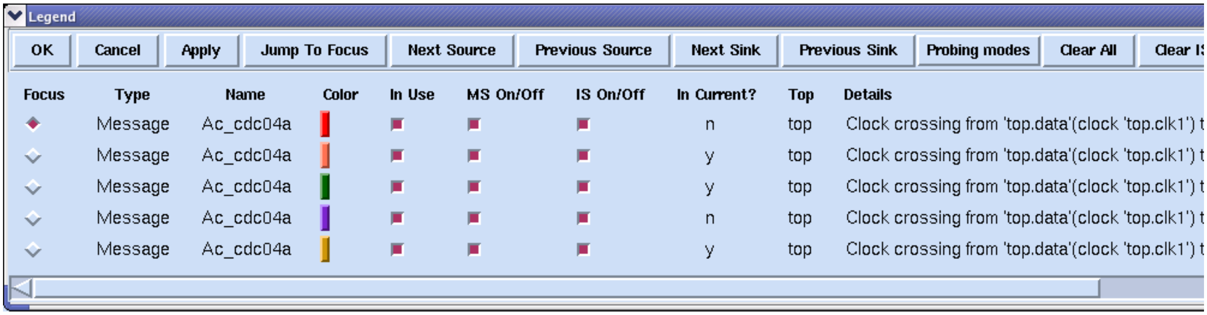
Option	**Description**
Hide Selected Signal(s)	Hides the selected signal(s)
Show Only These Signals	Displays only the selected signals in the Waveform Viewer window and hides all the other signals
Expand Bus <-> Collapse Bus	Expands all the collapsed buses to its individual signals. Similarly, collapses all the signals into a bus.
Change Signal Color	Changes the color of the selected signal. When you select this menu option, SpyGlass displays the Color dialog from which you can select the required color.
Change Signal Line Type	Changes the appearance of the line (normal, dashed, dotted, etc.) of the selected signal. When you select this menu option, SpyGlass displays a sub-menu from which you can select the desired line type (such as Normal, Dash, Dot, Dash-Dot, etc.)
Change Signal Radix	Changes the radix of the selected signal. When you select this menu option, SpyGlass displays a sub-menu from which you can select the required value (such as hex, bin, dec, signed, real, etc.)
Fan-in	Loads the selected signal in the fan-in cone of the selected signal for which a waveform is available. The fan-in signals for the selected signal are highlighted in the Schematic Viewer/RTL viewer.
Fan-in Cone	Displays a complete set of signals in the fan-in cone

Option	**Description**
Fan-out	Loads the selected signal in the fan-out cone of the selected signal for which a waveform is available. The fan-out signals for the selected signal are highlighted in the Schematic Viewer/RTL viewer.
Fan-out Cone	Displays a complete set of signals in the fan-out cone
Save	Saves the current set of signals as the current filter
Save As	Saves the current set of signals as a new filter

Legend Window

The *Legend Window* provides a consolidated list of all current highlights from all the sources.

To invoke the *Legend Window, select View >Show Classic Legend Window* menu option of the schematic window. The *Legend Window* appears, as shown in the following figure:



Focus	Type	Name	Color	In Use	MS On/Off	IS On/Off	In Current?	Top	Details
◆	Message	Ac_cdc04a	🟥	■	■	■	n	top	Clock crossing from 'top.data'(clock 'top.clk1') t
◇	Message	Ac_cdc04a	🟧	■	■	■	y	top	Clock crossing from 'top.data'(clock 'top.clk1') t
◇	Message	Ac_cdc04a	🟩	■	■	■	y	top	Clock crossing from 'top.data'(clock 'top.clk1') t
◇	Message	Ac_cdc04a	🟪	■	■	■	n	top	Clock crossing from 'top.data'(clock 'top.clk1') t
◇	Message	Ac_cdc04a	🟨	■	■	■	y	top	Clock crossing from 'top.data'(clock 'top.clk1') t

FIGURE 76. Legend Window

The *Legend Window* displays various details corresponding to the selected probe or message in the HDL Viewer, schematic window, or message window.

Probing Modes

The *Legend Window* has the following probing modes that can be selected from the *Probing* modes button:

- *Normal (Single Color)*

  Selected probes or messages are highlighted with only one color (the highlight color set in the *Schematic Page* of the *Tools > Preferences* menu option).

  All un-probed instances are shown in blue color and all un-probed nets are shown in yellow color in both schematic windows.

- *Grey (Single Color)*

  Selected probes or messages are highlighted with only one color (the highlight color set in the *Schematic Page* of the *Tools > Preferences* menu option).

All un-probed instances and nets are shown with grey mode color (set in the *Schematic Page* of the *Tools > Preferences* menu option).

- *Grey (Multiple Colors)*

  Selected probes or messages are highlighted with different colors automatically selected from the color palette.

  All un-probed instances and nets are shown with grey mode color (set in the *Schematic Page* of the *Tools > Preferences* menu option).

Probe or Message Details

For each selected probe or message, the following details are shown in the *Legend Window*:

- *Focus*

  This option sets the selected probe or message in focus.

  Use *Jump To Focus* button to view the selected probe or message.

  Use *Next Source*, *Previous Source*, *Next Sink*, or *Previous Sink* to traverse the probe or message group.

- *Type*

  This indicator shows the annotation type as *Message*, *Message-Add*, *Probe/Net*, *Probe/Inst*, or *Clock Domain* for main message, additional messages, nets or ports, instances, and clock domains respectively.

- *Name*

  This indicator shows the net, port, instance, clock domain name for probes and clock domains and the rule name for messages.

  For bus probes, an additional selector is available that lists each individual bus bit and the whole bus. For example, for a 4-bit bus, the selector has *[0]*, *[1]*, *[2]*, *[3]*, and *all*. You can then work with any bus bit (by selecting one of the bits) or work with the whole bus (by selecting *all*).

  For messages, move the cursor over the rule name to view the rule’s short description.

- *Color*

  This indicator shows the current color set for the message or probe.

To change the color, click the color indicator and select the new color for the probe or message.

- *In Use*

  Use this option to clear permanently the annotation for a probe or message. Then, the probe or message is not highlighted in various windows and is removed from the Legend window.

  To permanently remove the annotation of a probe or message, clear this check box and then click *Apply* or *OK*.

  If you close and reopen the Legend window during the current session, the probes or messages with the *In Use* check box clear are not displayed.

- *MS On/Off*

  Use this option to temporarily clear the annotation for a probe or message in the Modular Schematic window and HDL Viewer. Then, the probe or message is not highlighted but is still shown in the Legend window.

  To temporarily remove the message or probe, clear this check box and then click *Apply* or *OK*.

  If you close and reopen the Legend window during the current session, the probes or messages with the *MS On/Off* check box clear are still displayed for you to highlight them again in the Modular Schematic window and HDL Viewer.

- *IS On/Off*

  Use this option to temporarily clear the annotation for a probe or message (and related components) in the Incremental Schematic window. Then, the probe or message is not highlighted in the Incremental Schematic window but is still shown in the Legend window.

  To temporarily remove the message or probe, clear this check box and then click *Apply* or *OK*.

  If you close and reopen the Legend window during the current session, the probes or messages with the *IS On/Off* check box clear are still displayed for you to highlight them again in the Incremental Schematic window.

- *In Current*

  This indicator shows *y* if the probe or message is in the hierarchy currently shown in the schematic windows. Otherwise, this indicator shows *n*.

- *Top*

  This indicator shows the name of the top module containing the selected probe.

- *Details*

  This description shows the sinks and sources of the net and ports, gate type for instances, clock signals for clock domains, and rule details for messages.

  To see the complete details, bring the cursor over the details entry.

Viewing Sinks and Sources

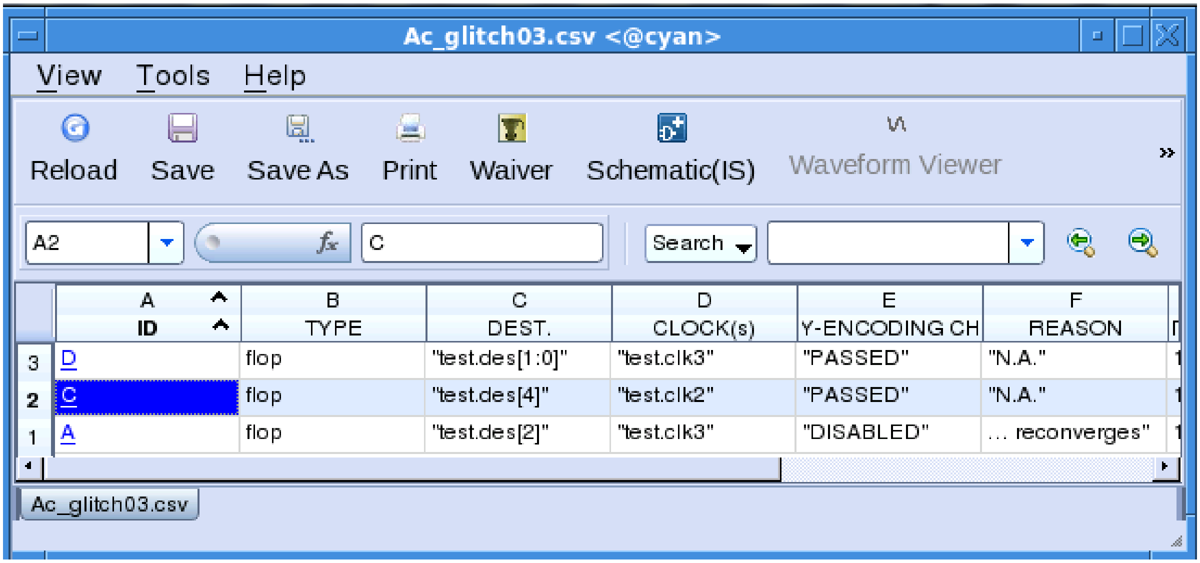
The *Legend Window* also allows you to view the sinks and sources of any net or port probe.

Select the net or port and then click *Next Sink* to view the first sink of the net or port. Use the *Next Sink* and *Previous Sink* to view next or previous sink of the net or port if the net has multiple sinks. Similarly, click *Next Source* to view the first source of a net or a port. Use the *Next Source* and *Previous Source* to view next or previous source of the net or port if the net or port has multiple sources.

Spreadsheet Viewer Window

The *Spreadsheet Viewer* window contains all the violation messages reported by a particular rule in a tabular format. The header of such rules is prefixed with the spreadsheet icon (▤).

To open a spreadsheet, select the rule header (under which all the violation messages of that rule are listed) and select the *Spreadsheet* link in the *Results* pane. This displays the *Spreadsheet Viewer* window, as shown in the following figure:



A2

C

	A ID	B TYPE	C DEST.	D CLOCK(s)	E Y-ENCODING CH	F REASON
3	D	flop	"test.des[1:0]"	"test.clk3"	"PASSED"	"N.A."
2	C	flop	"test.des[4]"	"test.clk2"	"PASSED"	"N.A."
1	A	flop	"test.des[2]"	"test.clk3"	"DISABLED"	... reconverges"

FIGURE 77. Spreadsheet Viewer Window

The above window displays the total number of violation messages in the status bar.

You can cross-probe from the *Spreadsheet Viewer* window to other windows/panes, such as *Schematic Window*, *HDL Viewer*, and File/Design Constraints/Instance views.

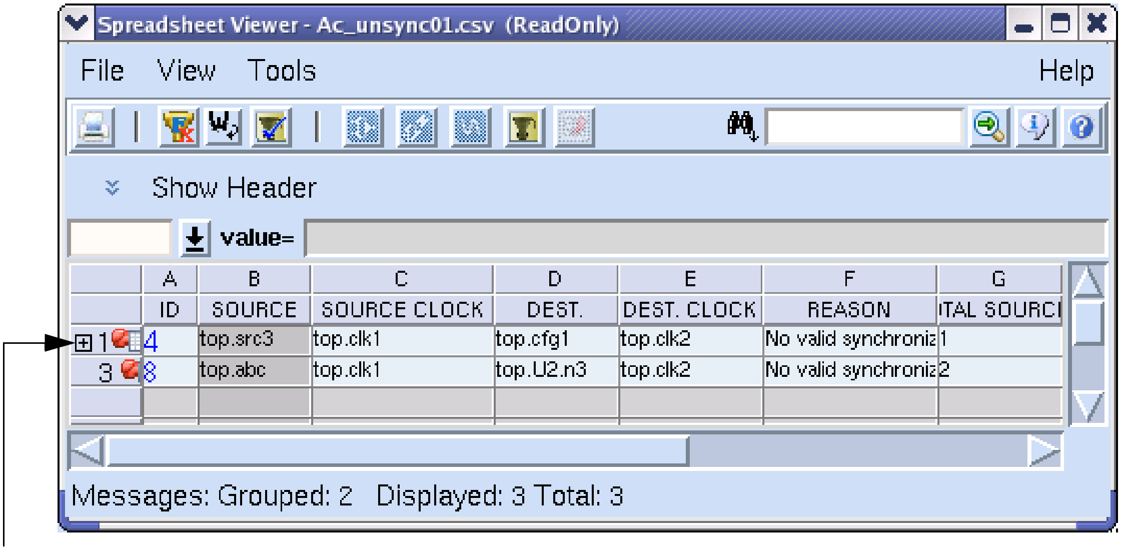
To enable cross-probing from the *Spreadsheet Viewer* window, SpyGlass provides hexadecimal violation *ID*, which corresponds to the hexadecimal violation *ID* generated for individual violation messages.

You can select multiple rows in the above spreadsheet by using Ctrl + click, and view a cumulative schematic showing all violations reported in the selected rows. These violations are shown in the Complete Display mode only.

NOTE: *If you de-select a primary violation in the spreadsheet, all its corresponding secondary violations are also de-selected.*

Viewing Grouped Messages in a Spreadsheet

Grouped messages in a spreadsheet appear as shown in the following figure:



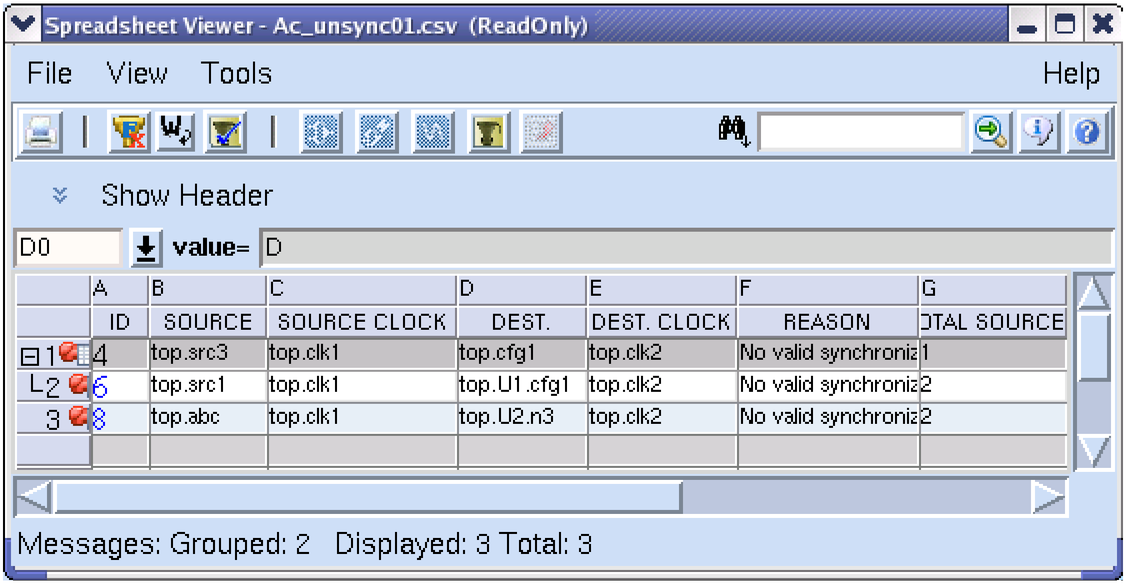
	A	B	C	D	E	F	G
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	ITAL SOURCI
⊞ 1	4	top.src3	top.clk1	top.cfg1	top.clk2	No valid synchroniz	1
3	8	top.abc	top.clk1	top.U2.n3	top.clk2	No valid synchroniz	2

Messages: Grouped: 2   Displayed: 3 Total: 3

Messages grouped

FIGURE 78. Viewing Grouped Messages - Collapsed View

In the above figure, click the + sign to view all grouped messages, as shown in the following figure:



D0 value= D

	A	B	C	D	E	F	G
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	OTAL SOURCE
⊟ 1	4	top.src3	top.clk1	top.cfg1	top.clk2	No valid synchroniz	1
└ 2	6	top.src1	top.clk1	top.U1.cfg1	top.clk2	No valid synchroniz	2
3	8	top.abc	top.clk1	top.U2.n3	top.clk2	No valid synchroniz	2

Messages: Grouped: 2 Displayed: 3 Total: 3

FIGURE 79. Viewing Grouped Messages - Expanded View

Waiving Rows in a Spreadsheet

The *Spreadsheet Viewer* in SpyGlass contains all the violation messages and related information for a particular rule in a tabular format. Sometimes, voluminous information is reported for a rule. The Waiver feature allows you to view the relevant messages and remove the irrelevant rows from the spreadsheet.

This section explains the following topics related to the waiver feature of the Spreadsheet Viewer:

- *Specifying Product-Specific and User-Specific Waivers*
- *Applying the Waiver Command*
- *Example of Applying the Waiver Command*
- *Editing the Waiver Commands Using Waiver Editor*

Specifying Product-Specific and User-Specific Waivers

You can waive rows in a spreadsheet to remove the irrelevant information by specifying following types of waivers:

- **Product-specific waivers**: Product-specific waivers are used to specify initial set of waiver commands. These waivers are specified in the product-defined waiver file. Commands in the product-defined waiver file gets overwritten on re-run.

  Syntax

  The syntax for defining a product-specific waiver file is as follows:

  ```
  spreadsheet_waiver_info/<CSV_NAME>_waiverinfo_product
  ```

  User-specific waivers: You can specify user-specific waiver commands for any CSV file to remove the specific information from being displayed. Right-click a row in the spreadsheet and select **Waive Row(s)** option to waive a row from the spreadsheet. This waiver command gets added in the user-specified waiver file. The user-specified waiver file does not get overwritten on re-run. Instead, new waiver commands get appended to existing commands in the file.

  Syntax

  The syntax for defining a user-specific waiver file is as follows:

  ```
  spreadsheet_waiver_info/<CSV_NAME>_waiverinfo_user
  ```

Applying the Waiver Command

The *sg_waive* command is used to waive a row from the spreadsheet. The syntax of the sg_waive command is as follows:

```
sg_waive -col {<column name/column_number>} -criteria
{<waiving_criteria>} -value {<value_for_criteria>}
```

Arguments

The *sg_waive* command takes the following arguments:

- **<column name/column number>**: name of the column or column number in the spreadsheet
- **<waiving_criteria>**: Waiving criteria for pattern matching based on which the row would be removed. See valid waiving criterion for a list of valid expressions that you can use as an input to this argument.

Following table lists the valid waiver criteria that you can specify for the -criteria argument:

TABLE 1 Valid Criterion

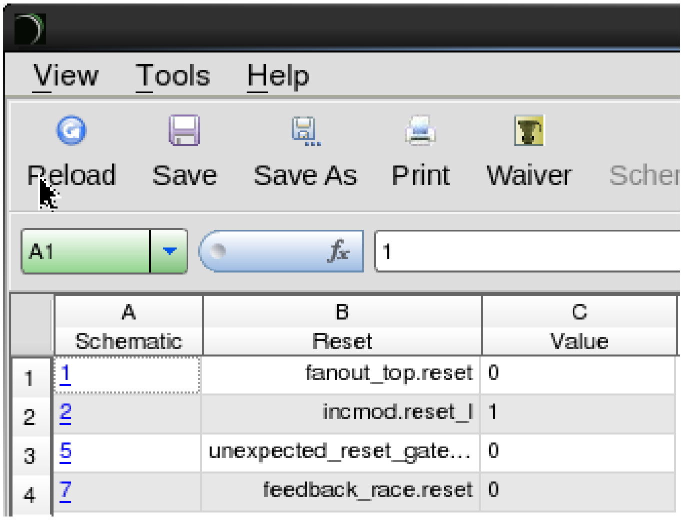
Syntax	Description
contains	Contains
! contains	Does not contain
==	Equals
!=	Not equal to
=~	Regular expression
begins	Begins with
!begins	Does not begin with
ends	Ends with
!ends	Does not end with

- **<value_for_criteria>**: Value to match with the value in the spreadsheet. Apart from strings and integers, you can also specify regular expression for pattern matching.

NOTE: *Do not include the columns, which contain the dynamic information, for generating the waiver commands. For example, Schematic ID,* in the *Example of Applying the Waiver Command* section described below, can change in each run. Therefore, it is not recommended to specify Schematic ID in a waiver command.

Example of Applying the Waiver Command

Consider following CSV file, which lists the asynchronous resets in the design:



	A Schematic	B Reset	C Value
1	1	fanout_top.reset	0
2	2	incmod.reset_l	1
3	5	unexpected_reset_gate...	0
4	7	feedback_race.reset	0

FIGURE 80. Example: Waiving Rows

Consider the following *sg_waive* command:

```
sg_waive -col {RESET} -criteria {Equals} -value
{incmod.reset_l} -col {VALUE} -criteria {Equals} -value {1}
```

The above command waives 2nd row from the CSV file.

To waive multiple messages, you can also use regular expression as shown in the following example:

```
sg_waive -col {RESET} -criteria {Contains} -value {*reset} -
col {VALUE} -criteria {==} -value {0}
```

The above command waives 1st, 2nd, and 3rd rows from the CSV file.

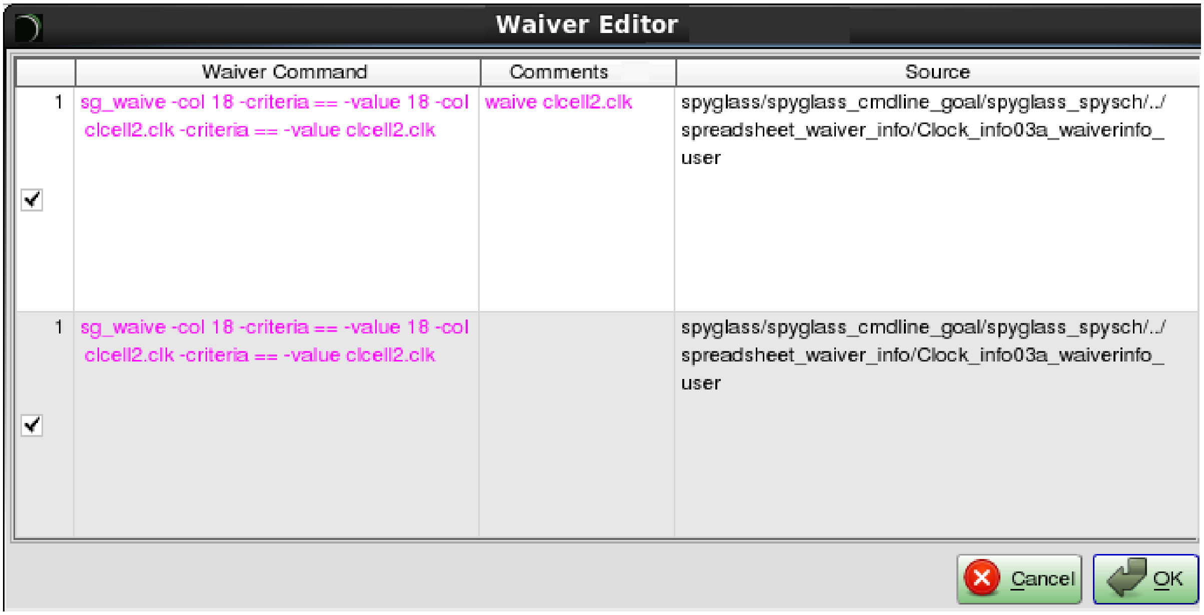
Editing the Waiver Commands Using Waiver Editor

You can edit the waiver command (sg_waive) for the waived row using the Waiver Editor window.

To edit a waiver command using the Waiver Editor window, perform the following steps from the spreadsheet viewer:

1. Click the Waiver option from the toolbar in the spreadsheet viewer.

   The Waiver Editor window is displayed as shown in Figure :



Waiver Editor

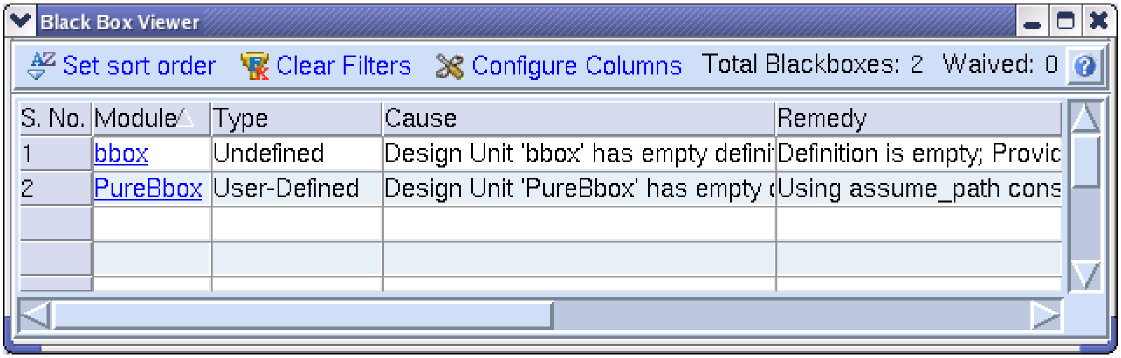
	Waiver Command	Comments	Source
1 ☑	sg_waive -col 18 -criteria == -value 18 -col clcell2.clk -criteria == -value clcell2.clk	waive clcell2.clk	spyglass/spyglass_cmdline_goal/spyglass_spysch/../spreadsheet_waiver_info/Clock_info03a_waiverinfo_user
1 ☑	sg_waive -col 18 -criteria == -value 18 -col clcell2.clk -criteria == -value clcell2.clk		spyglass/spyglass_cmdline_goal/spyglass_spysch/../spreadsheet_waiver_info/Clock_info03a_waiverinfo_user

FIGURE 81. Waiver Editor

Black Box Viewer Window

The *Black Box Viewer* window displays comprehensive information about all black box modules in the currently loaded design. This information appears in the form of a spreadsheet.

The following figure shows the *Black Box Viewer* window:



Total Blackboxes: 2 Waived: 0

S. No.	Module	Type	Cause	Remedy
1	bbox	Undefined	Design Unit 'bbox' has empty defini	Definition is empty; Provic
2	PureBbox	User-Defined	Design Unit 'PureBbox' has empty	Using assume_path cons

FIGURE 82. Black Box Viewer Window

In the above window, each row represents information about a particular black box module. These details are mentioned in the following table:

Column	Description
Module	Displays the name of the black box module
Type	Displays the type of the black box module
Cause	Displays the cause of the module being considered as a black box
Remedy	Displays the suggested remedy for converting the black box module to a regular module
File	Displays the design file name in which the black box module is present
Line	Displays the line number in the design file where the black box module is present

In this window, you can:

- Click a column header to sort the information in that column in ascending or descending order.
- Copy information of a cell to the clipboard.

  You can copy information by right-clicking the cell and selecting *Copy* option from the shortcut menu. The copied information can then be pasted in any other application or location.

Toolbar Options in the Black Box Viewer Window

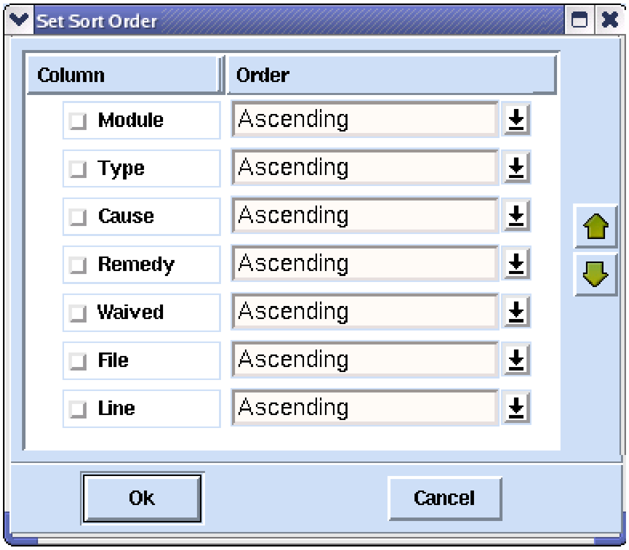
The *Black Box Viewer* window provides the following options in the toolbar:

- *Set Sort Order*
- *Clear Filters*
- *Configure Columns*

Set Sort Order

Select this option to specify a sorting order for various columns.

When you select this option, the *Set Sort Order* dialog appears. The following figure shows the *Set Sort Order* dialog:



Column	Order
□ Module	Ascending
□ Type	Ascending
□ Cause	Ascending
□ Remedy	Ascending
□ Waived	Ascending
□ File	Ascending
□ Line	Ascending

FIGURE 83. Black Box Viewer - Set Sort Order

The above dialog lists all the column names along with the sorting order for each column in a tabular format. The description of each field is given in the following table:

Field	Description
Column	Specifies the column name. You can select the check box adjacent to a column for which the sorting order needs to be changed.
Order	Specifies the sorting order for a particular column. To change the sorting order of a column, click ⤓ corresponding to that column and select the required sorting order from the drop-down list. NOTE: By default, each column is sorted in an ascending order

Specifying Priority for Columns

You can change (increase or decrease) the priority of columns to sort information within the entire table.

To change the priority of a column, select the entire row of that column and click ⬆ or ⬇ buttons to increase or decrease the priority for the selected column.

For example, if you want the information in the *Black Box Viewer* window to be first sorted by *Line*, then by *Cause*, and then by *Remedy*, perform the following steps:

1. Select the check boxes for the *Line*, *Cause*, and *Remedy* columns.
2. Select the entire row of the *Line* column.
3. Click ⬆ as many times as required to place the row for the *Line* column at the top-level.
4. Similarly, change the priority of the *Cause* and *Remedy* columns by clicking ⬆ or ⬇ buttons to place these columns in the required priority (*Line* > *Cause* > *Remedy*).
5. Click the *Ok* button.

After performing the above steps, the information in the *Black Box Viewer* window is rearranged according to the new sorting order.

Clear Filters

Select this option to clear all the applied filters.

For details on applying filters, see *Filters*.

Configure Columns

Select this option to show/hide columns or rearrange the placement of columns.

When you select this option, the *Give Column Order* dialog appears. The following figure shows the *Give Column Order* dialog:

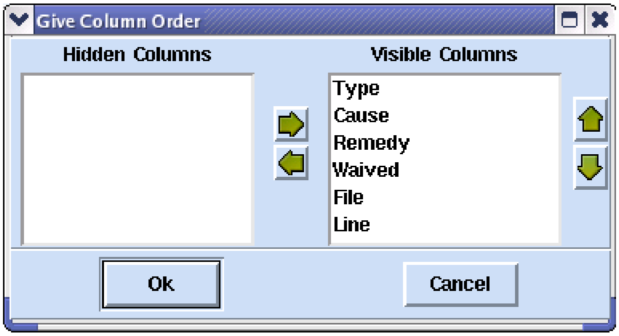


FIGURE 84. Black Box Viewer - Give Column Order

In the above dialog, select the required column name and click ➡ or ⬅ buttons to show or hide that column.

To rearrange the placement of displayed columns, select the column name in the *Visible Columns* section and click ⬆ or ⬇ buttons.

Right-Click Menu Options in the Black Box Viewer Window

When you right-click on a cell in the *Black Box Viewer* window, the following menu options appear:

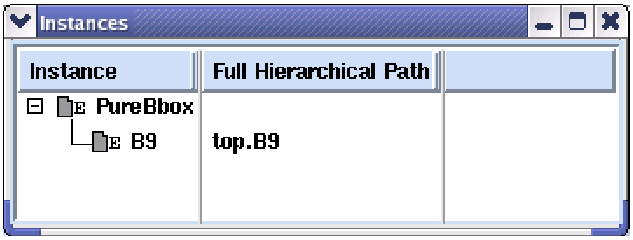
Copy

Select this option to copy the selected text or row.

Show Instance List

Select this option to view the instances of the selected black box module.

When you select this option, the *Instances* window appears. The following figure shows the *Instances* window:



Instance	Full Hierarchical Path	
PureBbox		
B9	top.B9	

FIGURE 85. Black Box Viewer - Instances Window

The above window displays all instances of the selected black box module along with their complete hierarchical paths.

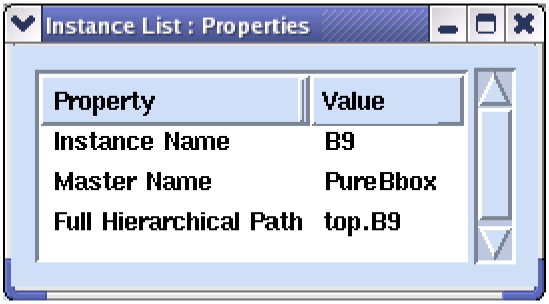
Right-Click Options in the Instances Window

When you right-click in the *Instances* window, the following options appear in the shortcut menu:

Properties

Select this option to view the properties of the selected black box instance.

When you select this option, the *Instance List: Properties* window appears. The following figure shows the *Instance List: Properties* window:



Property	Value
Instance Name	B9
Master Name	PureBbox
Full Hierarchical Path	top.B9

FIGURE 86. Black Box Viewer - Instances List

The above window displays the instance name, master module name, and the complete hierarchical path of that instance.

You can copy the contents of this window (using the right-click shortcut menu option or <*Ctrl*>+<*C*> key combination) and paste the contents at a different location.

Copy Instance Path

Select this option to copy the instance path.

Save Instance List

Select this option to save the instance list at a desired location.

When you select this option, the *Save Report As* dialog appears in which you can specify the location and file name containing the instance list.

Configure Page Size

Select this option to resize the *Instances* window if the list of instances is long and does not fit in the available space.

When you select this option, the *Configure page Size* dialog appears in which you can specify page size.

Waive this message

Select this option to waive the selected message.

When you select this option, the *Waiver Editor Window* appears in which you can specify the required waiver settings.

Edit Waivers

Select this option to edit the waiver settings specified for a particular message.

When you select this option, the *Waiver Editor Window* appears in which you can edit the previously created waiver settings.

Add Module Definition

Select this option to add a module definition for the selected black box to resolve that black box.

When you select this option, a sub-menu appears from which you can select the required option.

To resolve the selected black box, you can specify an HDL file, technology library file, Verilog library file, or Verilog library directory. Depending upon the case, select the required option from the sub-menu.

Set Sorting Order

Selecting this option is equivalent of selecting the *Set Sort Order* option in the toolbar.

Configure Columns

Selecting this option is equivalent of selecting the *Configure Columns* option in the toolbar.

Filters

Specifying a filter enables you to display only those rows that match the specified filter criteria.

Depending upon your requirement, select any of the following options under the *Filters* section:

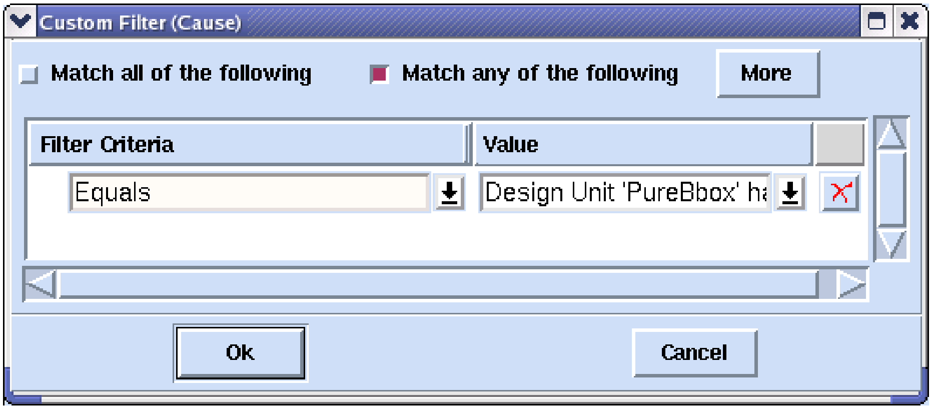
Filter Type	Description
All	Select this option to display all the rows.
Custom	Select this option to display specific rows based on the specified filter criteria. When you select this option, the *Custom Filter* dialog appears in which you can specify the required filter criteria. For details, see *Specifying Custom Filters*.
Un-synthesized	Select this option to display rows for only un-synthesized black box modules.
User-Defined	Select this option to display rows for only user-defined black box modules.
Undefined	Select this option to display rows for only undefined black box modules.

Specifying Custom Filters

You can specify custom filters on each column to view selective information in the *Black Box Viewer* window.

To specify your own custom filter, right-click on a column and select the *Custom* option from the shortcut menu. When you select this option, the *Custom Filter* dialog appears for that column.

The following figure shows the *Custom Filter* dialog:



Custom Filter (Cause)

□ Match all of the following

■ Match any of the following

Filter Criteria	Value
Equals	Design Unit 'PureBbox' ha

FIGURE 87. Black Box Viewer - Custom Filters

In the above dialog, you can:

- Specify the required filter criteria for the selected column from the *Filter Criteria* drop-down list.
- Select the required value for particular filter criteria from the *Value* drop-down list, which contains all the possible values for the selected column.
- Specify multiple filters by clicking the *More* button.

  When you click the *More* button, a new row is added in which you can specify an additional filter.
- Delete a particular filter by clicking ✗ button adjacent to the row containing that filter.

When more than one filter is used, you can specify whether to match any of the created filters or match all of the created filters. Depending upon your requirement, select the *Match any of the following* or *Match all of the following options*, respectively.

After specifying the required filters, click the *Ok* button to apply the filters in the *Black Box Viewer* window.

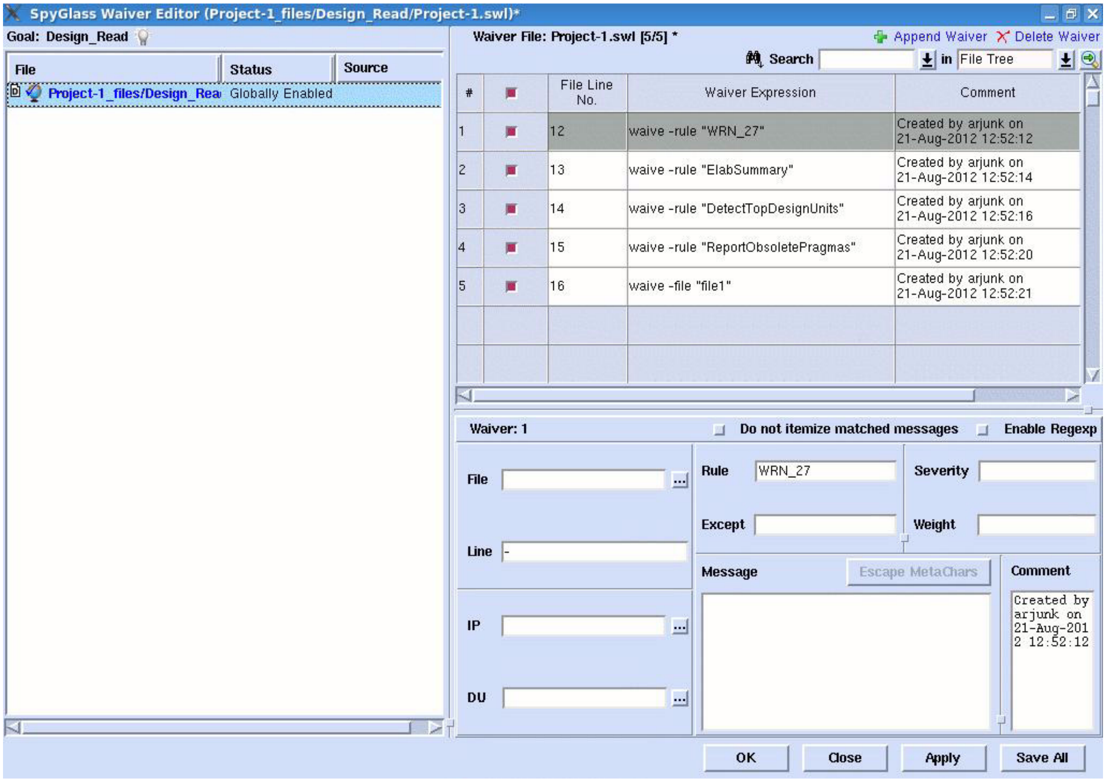
Waiver Editor Window

If a violation message does not indicate a serious problem, you can waive that message so that it does not appear in the reported messages list in the *Results Pane*. This way, you can focus on fixing other violations that indicate serious design issues.

Details of all waived messages are saved in waiver files (.swl files). You can load and modify these waiver files in the *Waiver Editor window*.

Details of the Waiver Editor Window

The following figure displays the *Waiver Editor window*:



SpyGlass Waiver Editor (Project-1_files/Design_Read/Project-1.swl)*

Goal: Design_Read

File	Status	Source
Project-1_files/Design_Rea	Globally Enabled	

Waiver File: Project-1.swl [5/5] *

#	▪	File Line No.	Waiver Expression	Comment
1	▪	12	waive -rule "WRN_27"	Created by arjunk on 21-Aug-2012 12:52:12
2	▪	13	waive -rule "ElabSummary"	Created by arjunk on 21-Aug-2012 12:52:14
3	▪	14	waive -rule "DetectTopDesignUnits"	Created by arjunk on 21-Aug-2012 12:52:16
4	▪	15	waive -rule "ReportObsoletePragmas"	Created by arjunk on 21-Aug-2012 12:52:20
5	▪	16	waive -file "file1"	Created by arjunk on 21-Aug-2012 12:52:21

Waiver: 1

□ Do not itemize matched messages □ Enable Regexp

Rule: WRN_27

Line: -

Comment: Created by arjunk on 21-Aug-2012 12:52:12

FIGURE 88. SpyGlass Waiver Editor Window

The *Waiver Editor window* contains the following four buttons at the bottom of the window:

Button	**Description**
OK	Click this button to apply the current waiver settings and exit the *Waiver Editor window* NOTE: SpyGlass prompts you for a re-run of SpyGlass when edited block waiver files, which have changed from the last run of SpyGlass are applied, or when some new hierarchical waive import commands are added in the waiver files.
Apply	Click this button to apply the current waiver settings (that is, any new waiver expressions or any changes made to the existing waiver expressions) in the *Waiver Editor window* and continue. NOTE: If the *Apply Waiver When Created* option is selected in the *Waiver* page in the *Preferences* dialog, any message selected for waiving is added to the *Waiver Editor window* and waived immediately. When you select the *Apply Waiver From Spread Sheet* option, the message is just added to the *Waiver Editor window* and is waived only when you click *Apply* or *OK* in the *Waiver Editor window*
Cancel	Click this button to close the *Waiver Editor window* to abandon the unapplied changes and exit. If there are any unapplied changes, SpyGlass prompts you to either cancel the action (takes you back to the Waiver Editor window so that you can apply changes) or continue with the action (closes the Waiver Editor window abandoning the unapplied modifications).
Save All	Click this button to save the modifications/changes made in the waiver file. If you have any read-only files with unsaved changes, then pressing the Saveall button prompts you to save the file by displaying the read-only Files dialog. Clicking the *Continue* button in this dialog saves the waiver files (other than read-only ones) to the disk. Clicking the *Cancel* button does not save any file to the disk.

This window is divided into the following sections:

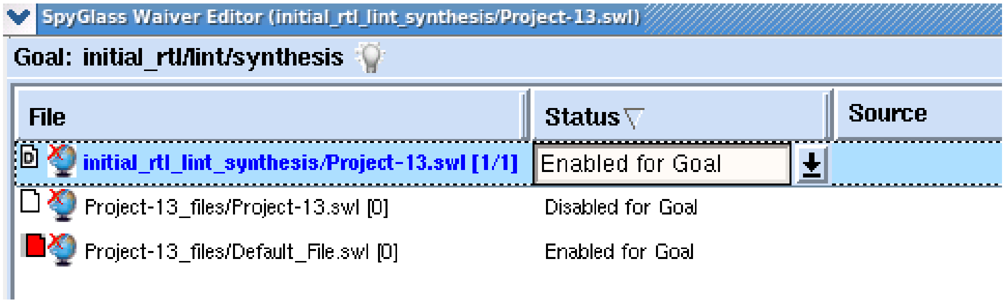
- *Tree-View Section*
- *Grid View Section*
- *Details Section*

You can resize all the above sections by dragging the edges of the required section.

Tree-View Section

This section lists the waiver files for the currently loaded goal. The name of the currently loaded goal appears on the header of this section.

The following figure illustrates the sample tree-view section:



SpyGlass Waiver Editor (initial_rtl_lint_synthesis/Project-13.swl)

Goal: initial_rtl/lint/synthesis

File	Status	Source
initial_rtl_lint_synthesis/Project-13.swl [1/1]	Enabled for Goal	
Project-13_files/Project-13.swl [0]	Disabled for Goal	
Project-13_files/Default_File.swl [0]	Enabled for Goal	

FIGURE 89. Tree -View Section

The above figure shows three waiver files for the currently loaded goal, **initial_rtl/lint/synthesis**. To enable or disable a waiver file for this goal, select the waiver file and then select the **Enabled for Goal** or **Disabled for Goal** option in the **Status** column.

Symbols Appearing for Different Waiver Files

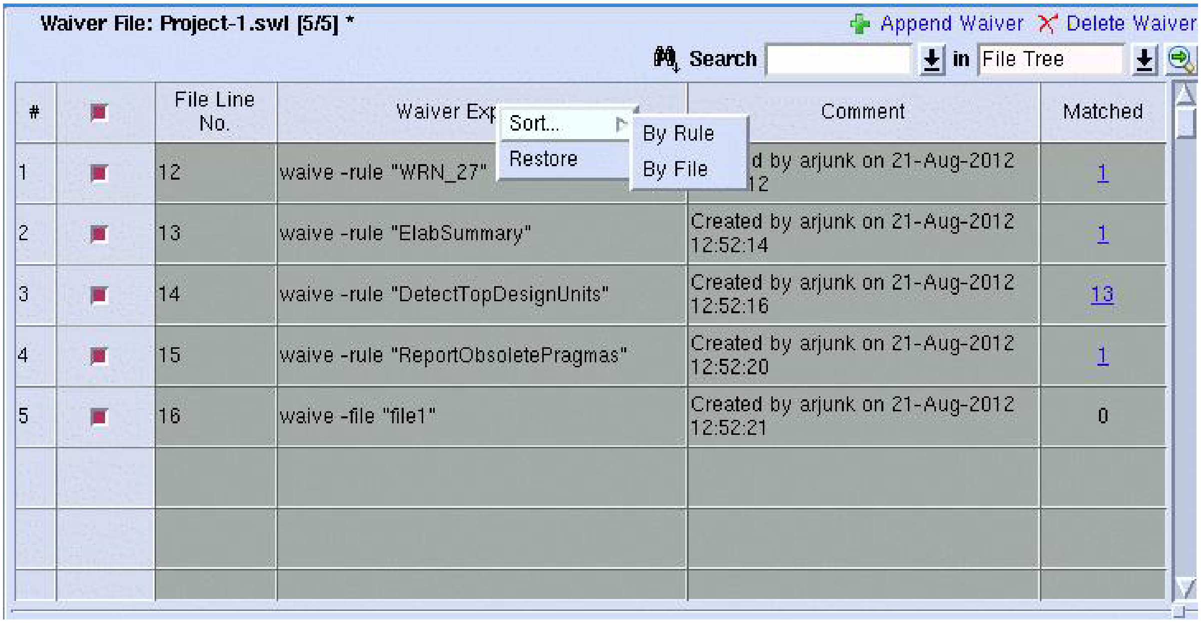
The following table provides the description of each symbol:

Symbol	Description
🟥	Specifies that the waiver file is read-only. This icon also appears for hierarchical migrated waiver files and pragma-based waiver files.
🟩🟥	Specifies that the waiver file contains partially-migrated waiver commands. When you move the cursor over such waiver files, a tool-tip appears specifying the name of that waiver file and the number of migrated commands over the total number commands.
D	Specifies the default waiver file. For details on a default file, see *Set File As Default*.

Grid View Section

This section contains waiver information in the form of a spreadsheet.

The following figure illustrates a sample grid-view:



Waiver File: Project-1.swl [5/5] *

#	■	File Line No.	Waiver Exp	Comment	Matched
1	■	12	waive -rule "WRN_27"	d by arjunk on 21-Aug-2012 12	1
2	■	13	waive -rule "ElabSummary"	Created by arjunk on 21-Aug-2012 12:52:14	1
3	■	14	waive -rule "DetectTopDesignUnits"	Created by arjunk on 21-Aug-2012 12:52:16	13
4	■	15	waive -rule "ReportObsoletePragmas"	Created by arjunk on 21-Aug-2012 12:52:20	1
5	■	16	waive -file "file1"	Created by arjunk on 21-Aug-2012 12:52:21	0

FIGURE 90. Grid View Section

When you double-click on a waiver file appearing in the tree-view section, the individual waiver details of that file appear in separate rows and columns in the grid-view. In addition, the name of that file appears on the title bar of the *Waiver Editor window*.

In the grid-view, you can perform the following actions:

- You can sort the available waiver expressions by a rule or a file. To sort, right-click on the waiver file header.

  Sorting the waivers disables the **Details** section of the Waiver Editor window. To re-enable the **Details** section, select the **Restore** option.

- Enable or disable individual waivers by selecting or deselecting the corresponding buttons (□) in each row.

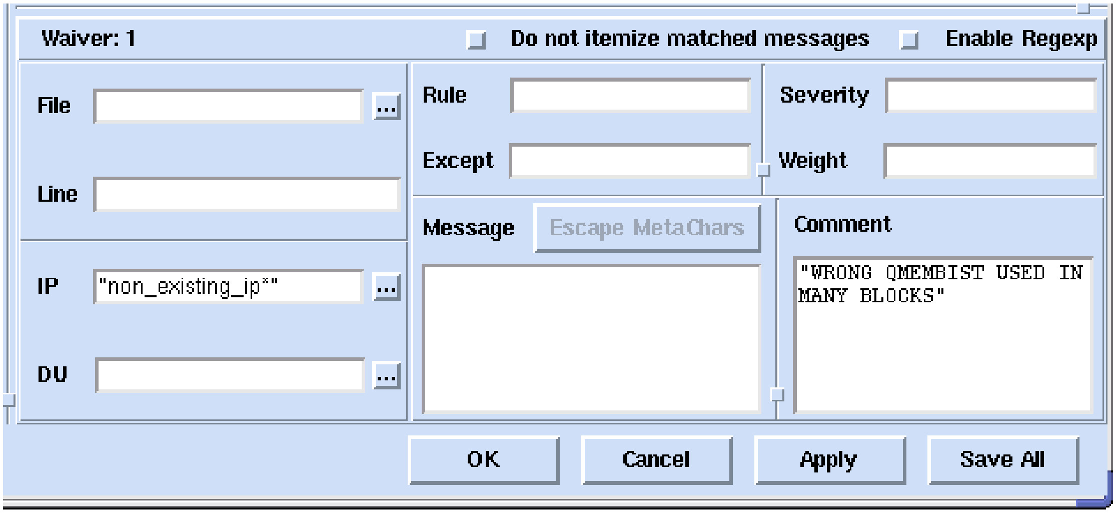
- To modify the settings of a waiver item, select □ button corresponding to that waiver. Details of the waiver is displayed in the *Details Section*. Update the required fields in the details section as required.
- Resize rows and columns of the grid by dragging the edges of rows and columns.
- Specify new waivers by clicking the *Append Waiver* option.
- Delete an existing waiver setting by clicking the *Delete Waiver* option.

You can view the list of rules for which waivers have been applied. The count of such rules is displayed as a hyperlink in the *Matched* column of the spreadsheet. When you click the count in the *Matched* column a list rules for which the waiver has been applied is displayed in the shortcut menu. You can also click a rule to locate the message in the Waived tree.

Details Section

This section enables you to specify details for the currently selected waiver file in the tree-view section.

The following figure illustrates the details-view section:



Waiver: 1

□ Do not itemize matched messages □ Enable Regexp

IP: "non_existing_ip*"

Comment: "WRONG QMEMBIST USED IN MANY BLOCKS"

FIGURE 91. Waiver Details

In the above section, you can specify the required waiver information in different fields.

The following points discuss various fields of this section:

- *File:* Use this field to specify files for which rule messages should be waived. You can also click ... and select a file from the *File Selection* dialog.
- *Line*: Use this field to specify the line for which rule messages should be waived.
- *IP*: Use this field to waive rule messages for the specified design unit (IP blocks) or all design units specified in the IP library. You can also click ... and select a file from the *IP Selection* dialog.
- *DU:* Use this field to waive rule messages for the specified design unit or all design units in the specified library. You can also click ... and select a file from the *DU Selection* dialog.
- *Rule*: Use this field to specify the rules, rule groups, or products for which the messages should be waived.
- *Severity*: Use this field to specify the severity class or severity label for which the rule messages should be waived.
- *Except*: Use this field to specify the rules, rule groups, or products for which the messages need to be excluded.
- *Weight*: Use this field to specify the weight of rules that need to be waived.
- *Comment*: Use this field to add a comment for the specified waiver. By default, when a row is added in the spreadsheet, the comment text box displays information about the user who created the waiver and the time when the waiver was created.
- *Message:* Use this field to specify the message that should be waived.

  In this section, at least one of the fields must be specified from one of the following field groups:

  Group 1: *File*/*Line*, *IP*, *DU*

  Group 2: *Rule*, *Severity*

  When more than one field is specified from Group 1, a message is waived if any one of the Group 1 field criteria is met. When more than one field is specified from Group 2, a message is waived only if all the

Group 2 field criteria are met. If fields from both Group 1 and Group 2 are specified, a message is waived only if any one of the Group 1 field criteria is met and all Group 2 field criteria are met.

When you type a value in the *File*, *Line*, *IP*, *DU*, *Rule*, *Severity*, *Except*, or *Weight* fields, the *Waiver Expression* column is dynamically updated with the value entered by you. For example, if you enter the value “NoXInCase-ML” in the *Rule* field, the *Waiver Expression* column is dynamically updated as follows:

```
waive -rule “NoXInCase-ML”
```

- *Enable Regexp*: Use this option to enable regular expression support in the selected fields. When you select this option, the *Escape MetaChars* option is also enabled.
- *Escape MetaChars*: As you enter a message expression in the *Message* field, any meta-characters entered are shown in red color. You can escape these meta-characters yourself or select the *Escape MetaChars* option to escape meta-characters.

  Escaped meta-characters are displayed in black color. The message expression entered in the *Message* field appears in the waiver description in the *Waiver Expression* field of the table.
- *User Comment*: Use this field to add/edit comments for waivers.
- *Do not itemize matched messages*: Select this option to report only the number of messages waived (instead of individual messages) in the waiver report.

Right-Click Menu Options in the Waiver Window

Different right-click menu options appear in the tree-view and grid-view section of the *Waiver* window. They are discussed in the following sections:

- *Right-Click Options of Tree-View Section*
- *Right-Click Options of Grid-View Section*

Right-Click Options of Tree-View Section

When you right-click on the *Waiver Files* node or a file under this node, the following options appear in the shortcut menu:

Add File

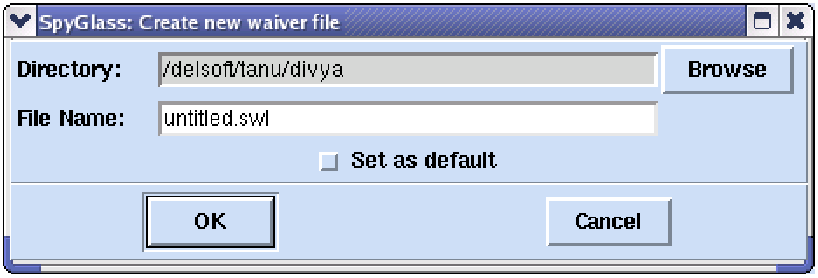
Select this option to add existing waiver files in the tree-view.

When you select this option, the *Open* dialog appears in which you can browse through the file system and select one or more waiver files to display.

New File

Select this option to add a new waiver file in the tree-view.

When you select this option, the *Create new waiver file* dialog appears, as shown in the following figure:



Directory: /delsoft/tanu/divya

File Name: untitled.swl

□ Set as default

FIGURE 92. Create New Waiver File

Selecting this option does not create the actual disk file until you explicitly save the file by using the *Save File As* option from the shortcut menu. Waiver files that are not yet saved on the disk are indicated with a red-colored icon in the tree-view.

Details of this dialog are given in the following points:

- *Directory* textbox enables you to specify the directory in which the waiver file is present.
- *Browse* button is used to browse to the directory of the waiver file.
- *File Name* textbox enables you to specify the name of the waiver file (preferably with .swl extension).
- *Set as default* option is used to set the waiver file as the default file.
- *OK* button is used to create the specified waiver file and add it in the tree-view.

NOTE: *Waiver files can have any extension. When you are creating a new waivers file, Atrenta Console suggests the .swl extension to facilitate better recognition and handling. If you do not specify the file extension for a waiver file, then by default, the .swl file extension is appended to the waiver file name.*

Set File As Default

Select this option to set the selected waiver file as the default waiver file for all SpyGlass sessions until you select another waiver file to be the default file.

The default waiver file name is highlighted in blue color in the tree-view, and the 🗎 icon appears before such file name.

Edit File

Select this option to edit the selected waiver file.

When you select this option, a text editor appears in which you can make the required changes.

Save File

Select this option to save the selected file with its current waiver settings.

Save File As

Select this option to save the selected waiver file with a different name and/or at a different location.

When you select this option, the *Save As* dialog appears in which you can specify the file name and the directory in which the file should be saved.

NOTE: *The waiver files are not saved automatically. You need to explicitly save them or abandon them as required. You are also prompted to save the unsaved waiver files, if any, when you close the Waiver Editor window.*

Export Waiver Files

Select this option to save waiver information in a tab-separated format. This option appears only when you right-click on the *Waiver Files* node.

This format contains various tab-separated columns, such as FileName, Rule, Comment, No of matches, DU, IP, and Waiver Expression. Appropriate information is present under each column.

Saving waivers in this format is useful when you load this format in a csv file from a different editor window. In any editor window, waiver information in the csv file appears in appropriate columns with proper headings.

NOTE: *You cannot open this csv file through Atrenta Console GUI.*

Reload File

Select this option to reload a waiver file.

It is not recommended to edit the same file inside as well as outside the *Waiver Editor* window simultaneously. One set of changes will be lost whenever the file is saved and/or reloaded.

It is recommended to edit waiver files in the *Waiver Editor* window only. If you edit waiver files outside this window (for example by using a text editor), such edits are not effective until you reload that waiver file in the *Waiver Editor* window.

Remove File

Select this option to remove the selected waiver file from the tree-view.

Such files are not even visible in the *Unused Files* category in the tree-view.

NOTE: *Default waiver files that are not yet saved, indicated by a red colored file icon with the alphabet "D" (D), cannot be removed from the tree-view until they are first disabled.*

Right-Click Options of Grid-View Section

When you right-click one or more rows in the grid, following options appear in the right-click shortcut menu:

Insert Waiver > Row Above

Select this option to insert a new waiver row above the currently selected row.

Insert Waiver > Row Below

Select this option to insert a new waiver row below the currently selected row.

Insert Import Waiver > Row Above

Select this option to import a new hierarchical waiver row above the currently selected row.

Insert Import Waiver > Row Below

Select this option to import a new hierarchical waiver row below the currently selected row.

Delete Waiver(s)

Select this option to delete the currently selected row(s).

Copy

Select this option to copy waiver commands in the selected row.

Paste

Select this option to paste the copied waiver commands in the currently selected row.

Paste Below

Select this option to paste the copied waiver commands in the next row of the currently selected row.

Undo

Select this option to undo the last change.

Redo

Select this option to redo the last undo change.

Display Comment

Select this option to view the complete text of the User Comment field in a pop-up box.

Set Row Height

Select this option to set the row height between 1 (default) to 5 units for the entire table. This option is useful if the table contains long message expressions that do not fit in the default row height.

Insert Common Comment

Select this option to add a common comment to all the selected waiver commands. When you select this option, SpyGlass displays the *Add Common Comment* dialog. In this dialog, add the required comment in the textbox, and click the *OK* button. This adds the specified comment to all the selected waiver commands.

Cross Probe to File

Select this option to cross probe between the selected waiver file and the Waiver Editor window.

Search Capability

You can search for a waiver expression or a waiver file using the search options illustrated in the following figure:

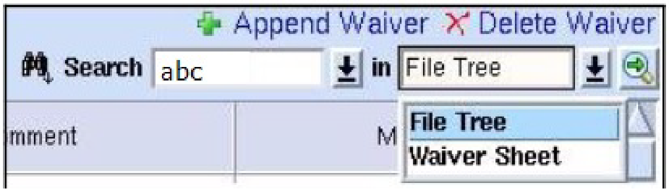


FIGURE 93.

To search for a string, perform the following steps from the Waiver Editor window:

1. Specify a string in the **Search** box.
2. Click 🔍 icon, if you want to make your search case insensitive or if you want to search backwards.
3. Select one of the following options from the **in** drop-down list:
   a. **File Tree**: Select this option, if you want to search for the specified string in the Tree-view section.

b. **Waiver Sheet**: Select this option, if you want to search for the specified in the Grid-View section.

4. Click the Search 🔍 button.

   The matching results are highlighted in the Tree-View section or the Grid-view section, based on your selection.

Waiver Settings Affecting the Msg Tree Results

If SpyGlass analysis has already been run on the current design, any modifications in the waiver settings are reflected in the *Msg Tree*. However, the *Msg Tree* itself is persistent and retains its expanded state.

NOTE: *It is recommended that you save (or discard) the settings made in the Waiver Editor window before starting a new SpyGlass analysis. Otherwise, there can be a mismatch between the number of messages actually displayed in Atrenta Console (this is based on the current settings of Waiver Editor window) and the number of messages that should be displayed based on the saved (physical) waiver files. Atrenta Console also prompts you if such a mismatch exists at the time of starting SpyGlass analysis run.*

Hierarchy Traversal across Atrenta Console Windows

For the purpose of hierarchy traversal, the following types of design units are assumed:

- RTL Module: Refers to a module that contains line number dumped by the message
- Schematic Top Module: Refers to a top hierarchy for which schematic back-annotation data is dumped by a message
- Schematic Module: Refers to a module for which schematic back-annotation data is dumped by a message

When a message is selected, the RTL module is always shown in *HDL Viewer Pane*. However, the Modular Schematic window and Incremental Schematic window can show a Schematic Module depending on the schematic highlight data dumped by the message.

There can be the following three cases where the ‘Instance hierarchies’ can differ:

1. RTL Module and Schematic Top Module are same.

   Schematic Top Module is shown in the schematic windows and Schematic Top Module is picked as the ‘Instance hierarchy’ for schematic windows.

   Highlighting is visible in both Modular Schematic window and Incremental Schematic window.

2. RTL Module is different from Schematic Top Module but there is a Schematic Module same as RTL Module.

   Schematic Module is shown in Modular Schematic window and hierarchy of Schematic Module is picked for Modular Schematic window. For example, if Schematic Top Module is A.b.c and the RTL module is D, an instance of D is searched in the schematic data and shown. A.b.c.d will be considered as ‘Instance hierarchy’ for Modular Schematic window. Both Modular Schematic window and Incremental Schematic window show the schematic highlighting. The case in which no instance of D is found is covered in Case 3 below.

3. No Schematic module for RTL Module

   RTL Module is shown in Modular Schematic window and any ‘Instance hierarchy’ is picked for that RTL Module from the Design view page. This behavior is also valid for the non-schematic messages also. There is no highlighting in the Modular Schematic window. However, if the top module of the picked ‘Instance hierarchy’ is same as that of the Schematic Top module, the Incremental Schematic window shows the back-annotation data.

The Configuration File in Atrenta Console

Overview

Atrenta Console has a Configuration File feature using which you can specify configuration settings like default startup mode (Atrenta Console GUI or batch), default product to be run, default language setting, and default report format.

The setup information in a SpyGlass Configuration file is read by SpyGlass transparently and automatically. Therefore, it differs from a command file that must be always specified by using the `read_file sourcelist` *`<file-name>`* project file command.

SpyGlass Configuration file defines working defaults that are lower priority and can be overridden by settings in a command file or directly through command-line.

The SpyGlass Configuration File is an ASCII text file named .spyglass.setup that can be located in four different locations so that you set four levels of configuration settings:

1. A configuration file specified using the *configfile* command.
2. A SpyGlass Configuration File located in the Current Working Directory.
3. A SpyGlass Configuration File located in your Home directory ($HOME)

4. A SpyGlass Configuration File specified by setting environment variable SPYGLASS_CUSTOMER_CONFIG_FILE
5. A SpyGlass Configuration File located in SpyGlass Installation Directory (<installation-dir>/SPYGLASS_HOME)

You can have a separate SpyGlass Configuration File in each of the five locations or a combination of these locations.

The following lists the order of priority of the configuration settings:

1. The configuration settings in a configuration file specified using the *configfile* command
2. The configuration settings in a SpyGlass Configuration File located in the Current Working Directory
3. The configuration settings in a SpyGlass Configuration File located in the User’s Home directory
4. The configuration settings in the SPYGLASS_CUSTOMER_CONFIG_FILE environment variable
5. The configuration settings in a SpyGlass Configuration File located in the SpyGlass installation Directory.

Structure of SpyGlass Configuration File

The SpyGlass Configuration File (the .spyglass.setup file) contains the different configuration settings in following format:

```
[ DEFAULT_STARTUP_MODE = gui | batch ]
[ USE_32_BIT_EXECUTABLE_ONLY = no | yes ]

[ DEFAULT_LANGUAGE_MODE = VHDL | Verilog | mixed | def | none ]
[ DEFAULT_TEMPLATE = <goal-name> | none ]
[ DEFAULT_TEMPLATE_DIRECTORY = GUIDEWARE_NEW_RTL |
                               GUIDEWARE_IP_RTL |
                               GUIDEWARE_IP_NETLIST |
                               GUIDEWARE_SOC
[ DEFAULT_POLICY_FOR_SPYEXPLAIN = <product-list> | none | all ]
[ DEFAULT_REPORT_FORMAT = moresimple | none | <report-name> ]
[ DEFAULT_REPORT_FORMAT_FOR_SLC = <report-name> ]
[ DEFAULT_PRAGMA = default | none | <pragma-name-list> ]
[ DEFAULT_BBOX_MODEL = BBOX_ILM | BBOX_CELLDEFINE |
                       BBOX_LIBCELL | BBOX_ENCRYPTED_LIB
                       <any combination> | NONE ]
[ AUTOENABLE_RULEGROUP_PARAMETER_CONTROL = no | yes ]
[ AUTOENABLE_MEMORY_HANDLING = no | yes ]
[ AUTOENABLE_HUGE_SCHEMATIC_DISPLAY = no | yes ]

[ VHDL_LIB_MAP = SYNOPSYS
           $SPYGLASS_HOME/vhdl_libs/$SPYGLASS_PLATFORM/SYNOPSYS |
  { VHDL_LIB_MAP = <logical-lib-name> <physical-path> }]
[ AUTOENABLE_VHDL_SORT = no | yes ]
[ DEFAULT_VHDL_SORT_METHOD = <method> ]

[ AUTOENABLE_INFERBLACKBOX = no | yes | yes_netlist | yes_rtl ]
[ AUTOENABLE_PRECOMPILED_VLOG = no | yes ]

[{ COMMAND_OPTION_FILENAME = <file-name> }]
[{ COMMAND_FILE_ARGS = <arg-list> }]
[ SGDC_INCLUDE_FILE_PATH = <dir-name> ]
[ OVERLOAD = <named-overload-list> ]
[ DEFAULT_SLF_CONFIG_FILE = <file-name> ]
[ AUTOENABLE_BUILTIN_CHECKS_FOR_POLICY = <product-list> ]
[ DEFAULT_METHODOLOGY = $SPYGLASS_HOME/GuideWare/New_RTL ]
[ METHODOLOGY_SEARCH_PATH = <space-separated-list-of-directories> ]
[ UI_WAIVER_DEFAULT_REGEXP_EXCLUDE_FIELDS = <comma-separated-list-of-
options> ]
DC_DWARE_FILES_PATH = <file_path>
DC_DW_FILES_PATH = <file_path>

[ ABSTRACT_FILE_NAME_STYLE = <compat | short> ]
```

Please note the following points:

- The default value of each configuration setting is shown underlined. The default value is applicable if the corresponding configuration setting is not found in the SpyGlass Configuration File(s).
- You should specify commands in this file only in classic batch format. This file does not support Tcl format.
- The SpyGlass Configuration file can have comments of format (VHDL-like comments), format (Verilog-like comments), or # format (SpyGlass common format).
- The SPYGLASS_HOME environment variable is supported in a Configuration File and is appropriately set to the SpyGlass Home directory when SpyGlass is run. All user-defined environment variables are supported in a Configuration File just like in a command file.
- The Configuration File support replaces the currently available .spyrc feature from SpyGlass version 3.3.0 onwards. Thus, it is recommended that you move to the SpyGlass Configuration File method as the .spyrc feature may be removed in a future release.
- Following configuration keys are not supported when the configuration file is specified in the project file:
  - ❐ DEFAULT_STARTUP_MODE
  - ❐ USE_32_BIT_EXECUTABLE_ONLY
  - ❐ DEFAULT_LANGUAGE_MODE
  - ❐ DEFAULT_POLICY
  - ❐ DEFAULT_POLICY_FOR_SPYEXPLAIN
  - ❐ DEFAULT_REPORT_FORMAT_FOR_SLC

- DEFAULT_TEMPLATE
- DEFAULT_TEMPLATE_DIRECTORY
- LICENSE_EXPIRY_NOTIFICATION_DAYS
- LICENSE_QUEUING_INTERVALS_IN_SECS
- DEFAULT_METHODOLOGY
- DEFAULT_EXE_TYPE_ON_64BIT
- METHODOLOGY_SEARCH_PATH
- AUTO_UI_LICENSE_TIMEOUT
- UI_LICENSE_QUEUING_WHEN_FEATURE_EXISTS

General Configuration Settings

The following keys are used for general configuration settings:

DEFAULT_STARTUP_MODE	*DEFAULT_EXE_TYPE_ON_64BIT*
USE_32_BIT_EXECUTABLE_ONLY	

DEFAULT_STARTUP_MODE

The DEFAULT_STARTUP_MODE key sets the SpyGlass startup mode as the Atrenta Console mode or the batch mode.

You can change the startup mode to batch mode or Atrenta Console mode by changing the value of the DEFAULT_STARTUP_MODE key as follows:

- gui: Sets Atrenta Console as the startup mode
- batch: Sets batch as the startup mode

NOTE: *By default, the startup mode is the Atrenta Console mode (key value gui).*

The DEFAULT_STARTUP_MODE key can be overridden by the -gui or -batch command-line options for values batch and GUI respectively.

DEFAULT_EXE_TYPE_ON_64BIT

The DEFAULT_EXE_TYPE_ON_64BIT key sets the default executable binaries (64-bit or 32-bit) to be executed for SpyGlass on 64-bit architectures.

By default, 64-bit SpyGlass binaries are executed on 64-bit architectures. To specify 32-bit binaries to be executed on 64-bit architectures, set the value of the DEFAULT_EXE_TYPE_ON_64BIT key to 32.

The DEFAULT_EXE_TYPE_ON_64BIT key can be overridden by the *-32bit* or *-64bit* command-line options for values 64 and 32 respectively.

USE_32_BIT_EXECUTABLE_ONLY

The USE_32_BIT_EXECUTABLE_ONLY key has been deprecated as this key was used only for 64-bit HP platform, which is no longer supported.

Product and Rules Configuration Settings

The following keys are used to configure settings for products and rules:

DEFAULT_LANGUAGE_MODE	*DEFAULT_TEMPLATE*
DEFAULT_TEMPLATE_DIRECTORY	*DEFAULT_POLICY*
DEFAULT_POLICY_FOR_SPYEXPLAIN	*DEFAULT_REPORT_FORMAT*
DEFAULT_REPORT_FORMAT_FOR_SLC	*DEFAULT_PRAGMA*
DEFAULT_BBOX_MODEL	*AUTOENABLE_RULEGROUP_PARAMETER_CONTROL*
AUTOENABLE_MEMORY_HANDLING	*AUTOENABLE_HUGE_SCHEMATIC_DISPLAY*
AUTOENABLE_BUILTIN_CHECKS_FOR_POLICY	*ABSTRACT_FILE_NAME_STYLE*

DEFAULT_LANGUAGE_MODE

Use this key to set a language for the HDL sources for SpyGlass analysis if no language is specified.

If this setting is not available, you must specify the language at command-line or a command file.

The DEFAULT_LANGUAGE_MODE key can be overridden by any of the specified language (vhdl, verilog, mixed, or def).

DEFAULT_TEMPLATE

Use this key to set a goal to run.

By default, SpyGlass does not pick any goal.

You can set the value to a desired goal in the *<methodology-name>/<goal-name>* format. For example, the following setting sets the default goal to be the Coverage goal of the SpyGlass DFT methodology:

```
DEFAULT_TEMPLATE = DFT/Coverage
```

You can also set a custom goal as the default goal. Just ensure that the full path to the methodology directory is specified by using the *I* command. For example, you have goal named `mygoal1` located in /usr/john/myGoals directory. Then, set the `DEFAULT_TEMPLATE` key as follows:

```
DEFAULT_TEMPLATE = myGoals/mygoal1
```

Also, specify the following command in the project file:

```
set_option I /usr/john ...
```

NOTE: *The I command is not recommended to be used in the project file. For details, see Options Not Recommended. However, you should specify the directory paths using the -I option on the command-line itself while invoking console or sg_shell.*

The DEFAULT_TEMPLATE key can be overridden by the *-template <goal-names>* command-line option.

DEFAULT_TEMPLATE_DIRECTORY

Use this key to set a goal directory.

You can set the `DEFAULT_TEMPLATE_DIRECTORY` key to the following values:

Value	Indicates
`GUIDEWARE_NEW_RTL`	(Default) The GuideWare goals for the New_RTL methodology (installed at <your-inst-dir>/SPYGLASS_HOME/GuideWare/New_RTL directory)
`GUIDEWARE_IP_RTL`	The GuideWare goals for the IP_RTL methodology (installed at <your-inst-dir>/SPYGLASS_HOME/GuideWare/IP_RTL directory)
`GUIDEWARE_IP_NETLIST`	The GuideWare goals for the IP_RTL methodology (installed at <your-inst-dir>/SPYGLASS_HOME/GuideWare/IP_netlist directory)
`GUIDEWARE_SOC`	The GuideWare goals for the SoC methodology (installed at <your-inst-dir>/SPYGLASS_HOME/GuideWare/SoC directory)

The `DEFAULT_TEMPLATE_DIRECTORY` key can be overridden by the *-templatedir* command-line option.

DEFAULT_POLICY

NOTE: *This key is applicable only for classic batch. The classic batch mode of SpyGlass has entered in its End-of-Life period and will not be supported starting the SpyGlass 5.4.0 release.*

Use this key to lock the license of product/products to be run while invoking Atrenta Console if none of the *-policies | -policy*/and *-template <goal-names>* command-line option is specified directly at the command-line.

While invoking SpyGlass, if you do not use the DEFAULT_POLICY key and try to run a product, it might be possible that the license for the same product is checked out by some other user.

To avoid such a situation, you can lock the license for such product/products to be run by specifying them using the DEFAULT_POLICY key.

You should specify the product names separated by a space. The DEFAULT_POLICY key can be overridden by the *-policies | -policy* or *-template <goal-names>* command-line options.

DEFAULT_POLICY_FOR_SPYEXPLAIN

Use this key to set a default product to search with the spyexplain utility if the *-policies | -policy* command-line option is not specified directly on the command-line.

SpyGlass searches all installed products by default if you have not specified any product on the command-line.

To search a specified product by default, set the product mnemonic as the value. You can also specify multiple product mnemonics as a space or comma-separated single-line list.

You can set the value to none to disable searching any product including the SpyGlass Built-in products.

The `DEFAULT_POLICY_FOR_SPYEXPLAIN` key can be overridden by the *-policies | -policy* command-line option.

DEFAULT_REPORT_FORMAT

Use this key to set a default report format in which messages will be reported at end of SpyGlass run if the *report* command is not specified in project file or through *-f* file(s) on the command-line.

By default, the SpyGlass run generates the *moresimple* report.

You can set the value to none to disable this setting and not print any report at the end of SpyGlass run (that is, same behavior as *noreport* command).

The `DEFAULT_REPORT_FORMAT` key can be overridden by the *report* command.

DEFAULT_REPORT_FORMAT_FOR_SLC

Use this key to set a default report format in which messages will be reported at end of a SpyGlass Library Compiler run if the *report* command is not specified in project file or through *-f* file(s) on command-line.

DEFAULT_PRAGMA

Use this key to set pragma keywords.

By default, SpyGlass assumes synopsys as pragma keyword for Verilog designs and synopsys and pragma as pragma keywords for VHDL designs.

Set the name of the default pragma keyword as the value. You can also specify multiple pragma keywords as a space or comma-separated single-line list.

You can set the value to none to disable this setting (that is, same behavior as `set_option` *pragma* `nopragma` command in a project file).

DEFAULT_BBOX_MODEL

Use this key to define BBOX_MODEL types to be recognized for SpyGlass rule-checking.

By default, the DEFAULT_BBOX_MODEL configuration key is set to BBOX_LIBCELL so that only the SpyGlass Library Compiler-generated cells are considered as the BBOX_MODEL type.

You can set the DEFAULT_BBOX_MODEL configuration key to BBOX_ILM for user-defined BBOX_MODEL types, BBOX_CELLDEFINE for Verilog 'celldefine modules, BBOX_LIBCELL for the SpyGlass Library Compiler generated cells, BBOX_ENCRYPTED_LIB for the Precompiled and encrypted library cells, or any combination of these values as a space or comma-separated list.

You can also set the DEFAULT_BBOX_MODEL configuration key to NONE to disable the feature.

DEFAULT_VERBOSITY

Use this key to control the verbosity option for all SpyGlass runs.

Set the value of the key to one of the following values: 0, 1, 2, or 3.

NOTE: *Use of debug flags will automatically dump the verbosity level to 3. Even if config key DEFAULT_VERBOSITY is set to some lower level, and DEBUG options is specified, the verbosity will be changed to level 3.*

AUTOENABLE_RULEGROUP_PARAMETER_CONTROL

Use this key to specify whether rules of a rule group specified with the *rules* command are always run or are run according to their rule-running condition.

By default, the AUTOENABLE_RULEGROUP_PARAMETER_CONTROL configuration key is set to no so that the rules of a rule group specified with the *rules* command are run irrespective of their rule-running conditions. In addition, rules that are enabled/disabled by a Boolean-type rule parameter (for example, the fast rule parameter in the SpyGlass lint solution disables some rules) are run irrespective of the rule parameter status.

You can set the AUTOENABLE_RULEGROUP_PARAMETER_CONTROL configuration key to yes so that the rules of a rule group specified with the *rules* command are run according to their rule-running condition. A rule that is switched off by default will not be run. In addition, rules that are enabled/disabled by a Boolean-type rule parameter are run based on the rule parameter status only.

AUTOENABLE_MEMORY_HANDLING

Use this key to enable the memory reduction feature.

By default, the AUTOENABLE_MEMORY_HANDLING key is set to no and the Memory Reduction feature is not enabled.

AUTOENABLE_HUGE_SCHEMATIC_DISPLAY

Use this key to load huge schematics in *The Modular Schematic Window*.

By default, such schematics are not loaded.

AUTOENABLE_BUILTIN_CHECKS_FOR_POLICY

Use this key to run built-in rules of products specified in its value list, irrespective of rule set specified in a project file or goals. Products specified in this list are loaded and built-in rules are run in following situations:

- While precompiling your design using the command `set_option` *noelab* `yes` specified in a project file, the `set_goal_option` *norules* `yes` command is specified in a project file.
- Rules are run without the noelab option. For products that are common in the list specified through the goals and that specified in the configuration key, built-in rules are enabled irrespective of the rule set through commands (such as *rules*) or goals.

ABSTRACT_FILE_NAME_STYLE

The ABSTRACT_FILE_NAME_STYLE key controls the naming style of the abstract files generated by SpyGlass. You can specify one of the following values specified below:

- **short**: Generates short file name in the <module_name>_<product>_abstract.sgdc format. For example, deep_cdc_abstract.sgdc.
- **comapt**: Generates unique file names with parameter details in the <module_name>_<parameter_details>_<product>_abstract.sgdc. For example, deep_BUS_1_WIDTH_1_cdc_abstract.sgdc.

By default, the value of the *ABSTRACT_FILE_NAME_STYLE configuration key* is set to `compat`.

Configuration Settings for VHDL Designs

The following keys are used to configure settings for VHDL designs:

VHDL_LIB_MAP	*AUTOENABLE_VHDL_SORT*
DEFAULT_VHDL_SORT_METHOD	

VHDL_LIB_MAP

Use this key to set default VHDL library mappings.

This configuration setting is equivalent to the *lib* command in the project file.

By default, the VHDL_LIB_MAP key is set to the following value:

```
SYNOPSYS $SPYGLASS_HOME/vhdl_libs/$SPYGLASS_PLATFORM/SYNOPSYS
```

The VHDL_LIB_MAP configuration setting can be repeated for specification of multiple logical library maps. However, for any one logical library name, only one setting is taken as per order of precedence defined earlier. The value specified for a logical library map in a project file or the *lib* command has higher precedence than this specification.

When SpyGlass is run, library map of all logical libraries as defined in either configuration file or command file is taken. For example, if library L1 and L2 are mapped in the configuration File and library L3 and L4 are mapped in a specified project file, SpyGlass is run with all four library map arguments.

The VHDL_LIB_MAP Configuration Setting also has an additive effect. Therefore, a VHDL library mapping specified using the VHDL_LIB_MAP configuration setting that is not specified either directly through the *lib* command or through a project file is added to SpyGlass command-line options in addition to other VHDL library mappings specified either directly on command-line or through a project file.

The VHDL_LIB_MAP Configuration Setting has an additive effect for multiple-level Configuration Files. Thus, different VHDL library mappings specified using the VHDL_LIB_MAP Configuration Setting at different Configuration File levels, are all added to SpyGlass command-line options.

AUTOENABLE_VHDL_SORT

Use this key to specify automatic sorting of VHDL source files.

By default, the AUTOENABLE_VHDL_SORT key is set to no and the automatic sorting feature is disabled.

You can set the value of the AUTOENABLE_VHDL_SORT key to yes to enable the automatic sorting feature and specify the sorting algorithm using the DEFAULT_VHDL_SORT_METHOD key.

The AUTOENABLE_VHDL_SORT key can be overridden by the *sort* option in the project file.

DEFAULT_VHDL_SORT_METHOD

Use this key to set an algorithm type for SpyGlass automatic VHDL file sorting feature.

By default, the special automatic sorting algorithm is enabled. You can set the values of the DEFAULT_VHDL_SORT_METHOD key to lexical.

In case you are okay with the default sorting technique specified in the configuration file(s) (the last one takes the priority), simply specify the sort option to enable sorting of VHDL files based on this default sorting technique.

Configuration Settings for Verilog Designs

The following keys are used to configure settings for Verilog designs:

AUTOENABLE_INFERBLACKBOX	*AUTOENABLE_PRECOMPILED_VLOG*

AUTOENABLE_INFERBLACKBOX

Use this key to specify whether the SpyGlass Inferblackbox feature is enabled for all design phases, is enabled for a particular design phase, or is disabled.

By default, the AUTOENABLE_INFERBLACKBOX key is set to yes and the feature is enabled for both RTL and Netlist phases.

You can set the following values to the AUTOENABLE_INFERBLACKBOX key:

Value	**Effect**
yes or yes_netlist	Equivalent to specifying the set_option *inferblackbox* yes command
no	Do nothing

The AUTOENABLE_INFERBLACKBOX key can be overridden by the inferblackbox or disable_inferblackbox options.

AUTOENABLE_PRECOMPILED_VLOG

Use this key to support precompiled Verilog libraries.

To enable precompiled Verilog library support, you need to supply the `set_option` *enable_precompile_vlog* `yes` command in project file.

Setting the `AUTOENABLE_PRECOMPILED_VLOG` key to `yes` is equivalent to supplying the enable_precompile_vlog option.

Other Configuration Settings

The following configuration settings require you to set values as they do not have a default value:

COMMAND_OPTION_FILENAME	*COMMAND_FILE_ARGS*
SGDC_INCLUDE_FILE_PATH	*OVERLOAD*
DEFAULT_SLF_CONFIG_FILE	*AUTOENABLE_GATESLIB_AUTOCOMPILE*
DEFAULT_METHODOLOGY	*METHODOLOGY_SEARCH_PATH*

COMMAND_OPTION_FILENAME

Use this key to set a default command file.

This configuration setting is equivalent to the `read_file -type` *sourcelist* `<file-names>` command. The file specified with the `COMMAND_OPTION_FILENAME` key is always read before any actual command-line options.

You can supply this configuration setting multiple times in one Configuration File.

The `COMMAND_OPTION_FILENAME` configuration setting also has an additive effect. Thus, command files specified at any level (Configuration File(s), command-line, or in a command file) are all supplied to SpyGlass.

COMMAND_FILE_ARGS

Use this key to specify contents of a `read_file -type` *sourcelist* `<file-names>` command inside the configuration file.

A COMMAND_FILE_ARGS configuration setting must have its value in the same line where the key is specified and it cannot span multiple lines. However, you can have multiple specifications of the COMMAND_FILE_ARGS key in the same configuration file and the result of adding all these specifications will be used during the SpyGlass run. Also, the contents of this key in various configuration files are added (and not replaced as true for most of the other keys) and used for the SpyGlass run. This key is more convenient to use when you have concise -f listing for the configuration file and this option saves the overhead of creating a separate -f file and then using it inside the configuration file.

SGDC_INCLUDE_FILE_PATH

Use this key to specify a path from where included SpyGlass Design Constraints files can be picked.

By default, the SGDC_INCLUDE_FILE_PATH key is not set and the included SGDC files (specified using the INCLUDE directive in an SGDC file) are searched and included as follows:

1. If included SGDC file name is an absolute file name, the specified file at the specified location is included.
2. If included SGDC file name is a relative file name, the included SGDC file is searched in the relative directory location with respect to the location of the parent SGDC file.
3. If the included SGDC file name is any other type of file name (that is the file name does not start with a forward slash (first case above) or a period (second case above), the included SGDC file is searched with respect to the location of the parent SGDC file.

Use the SGDC_INCLUDE_FILE_PATH key to specify a different location for the last case above. Then, you can include company-, project-, or user-specific SGDC files without needing to specify exact path in each INCLUDE specification.

You can specify a space-separated list of directory names with the SGDC_INCLUDE_FILE_PATH key as in the following example:

```
SGDC_INCLUDE_FILE_PATH = /usr/corporate/sgdc
     /usr/projectfiles/sgdc /usr/john/mySGDC
```

You can also specify comma-separated or colon-separated lists of directory names.

The included SGDC files are searched in the specified directories in the same order in which they are specified with the SGDC_INCLUDE_FILE_PATH key.

NOTE: *In the last case, the included SGDC files are first searched with respect to the location of the parent SGDC file. Only if they are not found in these locations, the directories specified with the SGDC_INCLUDE_FILE_PATH key are searched.*

OVERLOAD

Use this key to set default named overloads.

By default, SpyGlass assumes no named overload. You can specify the named overloads as a space-separated list as in the following example:

```
OVERLOAD = CAD BOB METEOR
```

To disable named overloads, specify the OVERLOAD key with none value.

To set an additive behavior of the OVERLOAD key, use the OVERLOAD value. Consider the following example configuration files:

Configuration File	OVERLOAD Key Setting
$SPYGLASS_HOME/.spyglass.setup	OVERLOAD = CAD2
$HOME/.spyglass.setup	OVERLOAD = OVERLOAD METEOR
$CWD/.spyglass.setup	OVERLOAD = BOB OVERLOAD

The OVERLOAD value (shown in green color) in the $HOME/.spyglass.setup file indicates that SpyGlass should include the OVERLOAD key setting from the lower-precedence configuration file (that is, $SPYGLASS_HOME/ .spyglass.setup file). Similarly, the OVERLOAD value (shown in red color) in the $CWD/.spyglass.setup file indicates that SpyGlass should include the OVERLOAD key setting from the lower-precedence configuration file (that is, $HOME/.spyglass.setup file). Therefore, the effective value of the OVERLOAD key is as follows:

```
OVERLOAD = BOB CAD2 METEOR
```

DEFAULT_SLF_CONFIG_FILE

Use this key to specify the name of a text file that contains names of library attributes not to be reported as un-supported by the SpyGlass Library Compiler.

The format of the text file contents is as follows:

```
IGNORE_LIB_CONSTRUCT =
{
  [<attr-group>::]<attr-name>
  [<attr-group>::]<attr-name>
  [<attr-group>::]<attr-name>
...
}
```

Here, <attr-group> is the name of a library attribute group and <attr-name> is the name of a library attribute that belongs to the library attribute group <attr-group>. Thus, the following example suppresses the warnings for all instances of time_unit library attribute under the pin library group:

```
IGNORE_LIB_CONSTRUCT =
{
...
pin::time_unit
...
}
```

Specifying <attr-group> is optional. Thus, you can just specify the library attribute name to ignore it under all applicable library attribute groups. Thus, the following example causes library attribute direction to be ignored under all its applicable library attribute groups:

```
IGNORE_LIB_CONSTRUCT =
{
...
direction
...
}
```

You can add comments in the text file using the #-type comment format.

AUTOENABLE_GATESLIB_AUTOCOMPILE

Use this key to enable compilation of gate libraries (.lib) automatically to SpyGlass-compatible format library files (.sglib).

By default, the value of this key is set to no and the auto compilation of gate libraries does not occur.

You can set the value to yes or yes_forced to enable auto compilation of the gate libraries.

If you set the value to yes, the specified .lib files are compiled to .sglib file unless there is an up-to-date copy in the cache directory. However, if you set the value to `yes_forced`, any criteria for re-compilation of gate libraries are not evaluated. In such case, the specified .lib files are always compiled and they overwrite the existing .sglib file present in the cache directory.

The `AUTOENABLE_GATESLIB_AUTOCOMPILE` key value set to `yes` or `yes_forced` can be overridden by the `set_option` *enable_gateslib_autocompile* `no` command in a project file. In this case, the auto-compilation is triggered by specifying the `set_option` *enable_gateslib_autocompile* `yes` and/or `set_option` *force_gateslib_autocompile* `yes` commands in project file. Similarly, if value of this key is `no` then you can enable automatic compilation by specifying the *enable_gateslib_autocompile* command.

DEFAULT_METHODOLOGY

Use this key to set a directory path of default methodology files.

By default, the DEFAULT_METHODOLOGY key points to the directory containing the GuideWare New_RTL methodology files.

The DEFAULT_METHODOLOGY key takes the following values:

- Absolute path of the directory where the methodology resides.
- Relative path to the SPYGLASS_HOME directory. For example, $SPYGLASS_HOME/GuideWare/New_RTL
- Relative path to the current working directory. For example, ../GuideWare/New_RTL

METHODOLOGY_SEARCH_PATH

Use this key to specify paths of directories containing custom methodologies.

All such custom methodologies appear in the drop-down list adjacent to the *User Specified Methodology* option in the *Select Methodology* dialog.

For details on this option, refer to *Atrenta Console User Guide*.

Following is the example of setting the `METHODOLOGY_SEARCH_PATH` key:

```
METHODOLOGY_SEARCH_PATH=/path1/GuideWare
```

If a directory name appearing in the path contains spaces, enclose the path within double-quotes.

UI_WAIVER_DEFAULT_REGEXP_EXCLUDE_FIELDS

Use this key to specify the sub-fields to be excluded while generating waivers with regexp enabled.

You can specify one or more of the following values:

- du
- msg
- file
- ip
- severity

Specify multiple values as a comma-separated list.

Following is the example of setting the `UI_WAIVER_DEFAULT_REGEXP_EXCLUDE_FIELDS` key:

```
UI_WAIVER_DEFAULT_REGEXP_EXCLUDE_FIELDS = du, msg, file
```

DC_DWARE_FILES_PATH

Use this key to pick designware `dware` packages from a customized path other than the path where dc has been installed.

DC_DW_FILES_PATH

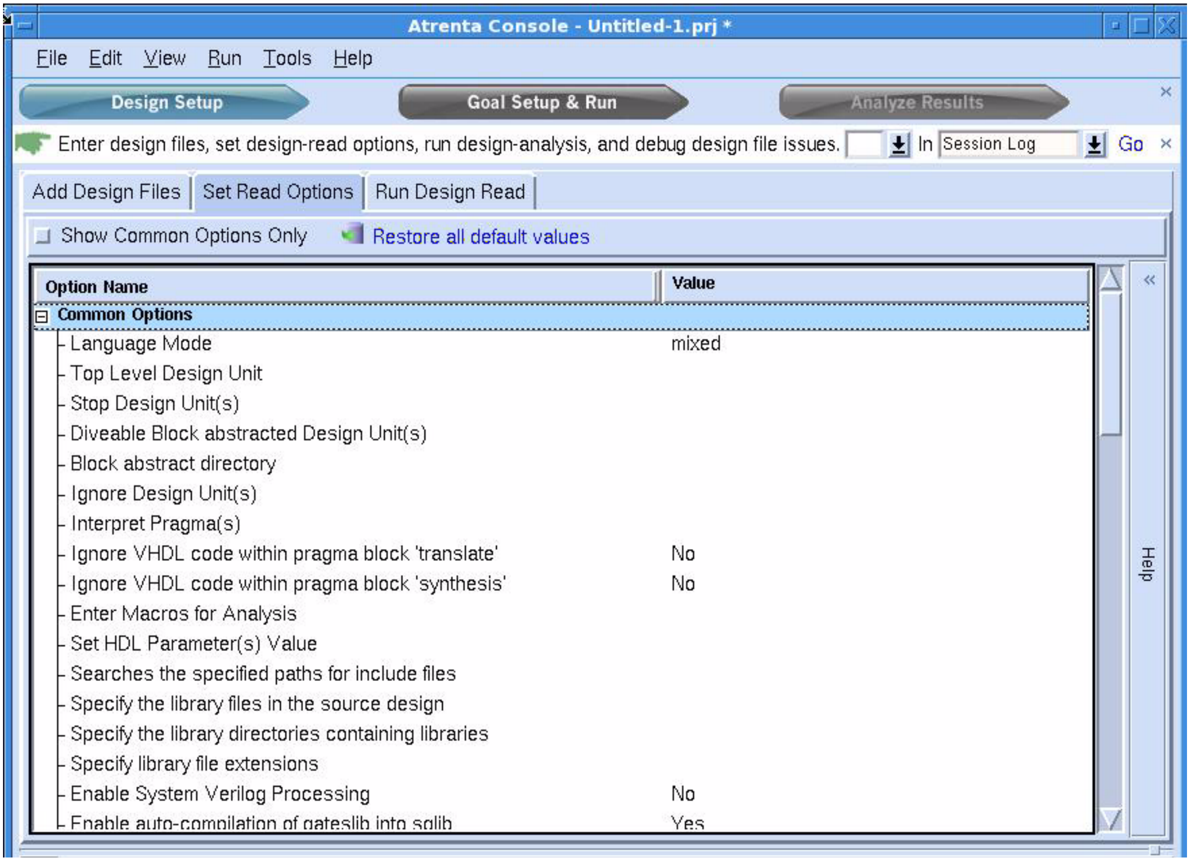
Use this key to pick designware `dw` packages from a customized path other than the path where dc has been installed.

Design-Read Options in Atrenta Console

Overview

The design-read options enable you to specify various settings during the design setup stage. For example, you can specify top-level modules of your design and set the logical working directory.

All these options are present under the *Set Read Options* tab, as shown in the following figure:



Option Name	Value
Common Options	
Language Mode	mixed
Top Level Design Unit	
Stop Design Unit(s)	
Diveable Block abstracted Design Unit(s)	
Block abstract directory	
Ignore Design Unit(s)	
Interpret Pragma(s)	
Ignore VHDL code within pragma block 'translate'	No
Ignore VHDL code within pragma block 'synthesis'	No
Enter Macros for Analysis	
Set HDL Parameter(s) Value	
Searches the specified paths for include files	
Specify the library files in the source design	
Specify the library directories containing libraries	
Specify library file extensions	
Enable System Verilog Processing	No
Enable auto-compilation of gateslib into sglib	Yes

FIGURE 1. Design Read Options

To set a design-read option, click that option and set it to a value in the *Value* field corresponding to that option. Alternatively, use the following command in the project file to set a design-read option:

```
set_option <option-name> <value>
```

There are certain Atrenta Console setting options that are determined when a project is created or saved. These options are displayed in the following table:

Option	Fixed Value	Default Value
project_directory	-	<CWD>
current_methodology	-	value of DEFAULT_METHODOLOGY in .spyglass.setup
language_mode	verilog/vhdl/mixed	mixed

NOTE: *Arguments using $ or other meta-characters that are meant to be passed directly to SpyGlass should be enclosed in curly brackets to prevent evaluation by the Tcl interpreter.*

Atrenta Console design-read options are divided into the following categories:

- *Common Design-Read Options*
- *Advanced Design-Read Options*

Common Design-Read Options

Common design-read options are used to set additional options important for design analysis that are not associated with any goal.

By default, the *Show Common Options Only* check box is selected and Atrenta Console displays only the commonly-used design-read options under the *Set Read Options* tab.

Whenever you save your project, Atrenta Console saves only the modified design-read options in the project file by default. If you also want to save the unmodified design-read options in the project file, perform the following steps:

1. Select the *File -> Project Properties* menu option.

   The *Project Properties* dialog appears.
2. Select the *Write unmodified options in Project File* check box in the *Project Properties* dialog.
3. Click the *OK* button.

The structure of the common options section in the project file: is as follows:

```
##Common Options Section
set_option <option-name> [<option-arguments>
```

Language Mode

Use this design-read option to set a language for the current project. The default language mode is `mixed`.

Other possible language modes are `verilog` and `vhdl`.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option language_mode <verilog | vhdl | mixed>
```

Top Level Design Unit

Use this design-read option to specify a top-level module so that all design units instantiated directly or indirectly under this module are included in the scope of SpyGlass analysis.

Modules outside the hierarchy of the top-level module are considered as black boxes during SpyGlass analysis. Only syntax checking and lexical rule checking is performed on such modules.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option top <name>
```

Example

The following command defines `alu` as the top-level module:

```
set_option top alu
```

Stop Design Unit(s)

Use this design-read option to specify design units which you want to skip from SpyGlass analysis.

Such design units are considered as black boxes during SpyGlass analysis.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option stop <module-name>
```

When this command is run, Atrenta Console generates the *stop_summary report*.

Format of Specifying Design Units with the stop Command

Specify design unit names in any of the following formats:

VHDL	`<entity-name>`	Skip rule-checking on the specified entity and all its architectures (in all logical libraries)
	`<entity-name>.<arch-name>`	Skip rule-checking on the specified architecture of the specified entity (in all logical libraries)
	`<lib-name>.<ent-name>`	Skip rule-checking on the specified entity and all its architectures (for the specified logical library)
	`<lib-name>.<ent-name>.<arch.name>`	Skip rule-checking on the specified architecture of the specified entity in the specified logical library
	`<lib-name>`	Skip rule-checking on all design units in the specified logical library
	`ALL`	Skip rule-checking on all design units in all logical libraries
	`ALL.<arch-name>`	Skip rule-checking on the specified architecture in all logical libraries
Verilog	`<module-udp-name>`	Skip rule-checking on the specified module or UDP
	`<lib-name>.<module-udp-name>`	Skip rule-checking on the specified module or UDP from the specified logical library

Examples of using the stop Command

Following are some examples:

- The following command specifies `alu` as the design unit to be skipped from SpyGlass analysis:

```
set_option stop alu
```

- The following command specifies block1 and block2 as the design units to be skipped from SpyGlass analysis:

  ```
  set_option stop {block1 block2}
  ```

- The following commands use wildcard expressions to specify design units:

Command	Action
set_option stop {lib1.*}	Skips all design units from the lib1 logical
set_option stop {e.*}	Skips all architectures of the e entity
set_option stop {*.e}	Skips all entities named e from any library

If you specify escaped names, you must escape wildcard characters. For example, \a123* should be specified as the following to avoid any unexpected matches:

```
set_option stop {\a123\*}
```

In addition, the following examples have different meanings:

```
set_option stop {a1*}
set_option stop {\a1\*}
```

Here, the first example matches a11, a12, and a13 while the second example matches \a1*.

Making Project File Read Only

Use this design read option to make the project file read only.

Sometimes a project file contains many environment variables and include paths. In such a case, you may not want to overwrite the project file. When working with GUI, the GUI prompts you to save the project file when exiting. If you click yes, the project file is overwritten.

To prevent overwriting the project file, add the following option to the project file:

```
set_option project_read_only yes
```

Specifying this option makes the project file as read only and the GUI does not prompt you to save any changes in this mode. You can still change values of parameters or design options and run the design, without saving them in the project file. The results of the run are saved and available when the project is loaded next time.

Also, if the value of this option is set to `yes`, Atrenta Console generates the following three files in the project directory on exit:

- **<project_wdir>Sel/<project_name>/project_sources.f**: Contains all the design sources used in the project.
- **<project_wdir>Sel/<project_name>/project_options.f**: Contains all the design options that you may have set.
- **<project_wdir>Sel/<project_name>/project_goals.f**: Contains all the goal related setup, such as, parameters and constraints.

You can source these files in the original project file, to preserve any changes in the GUI for the next session.

Diveable Block Abstracted Design Unit(s)

Use this design-read option to specify a list of abstract modules such that when you double-click on these modules in the schematic, SpyGlass shows the hierarchy within these modules. By default, SpyGlass does not allow you to view the hierarchy within such modules.

Setting this design-read option is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_abstract_blocks_schematic <space-or-comma-
separated-list-of-modules>
```

Consider that you specify the following command:

```
set_option enable_abstract_blocks_schematic E2
```

Now, consider the following schematic showing the E2 module:

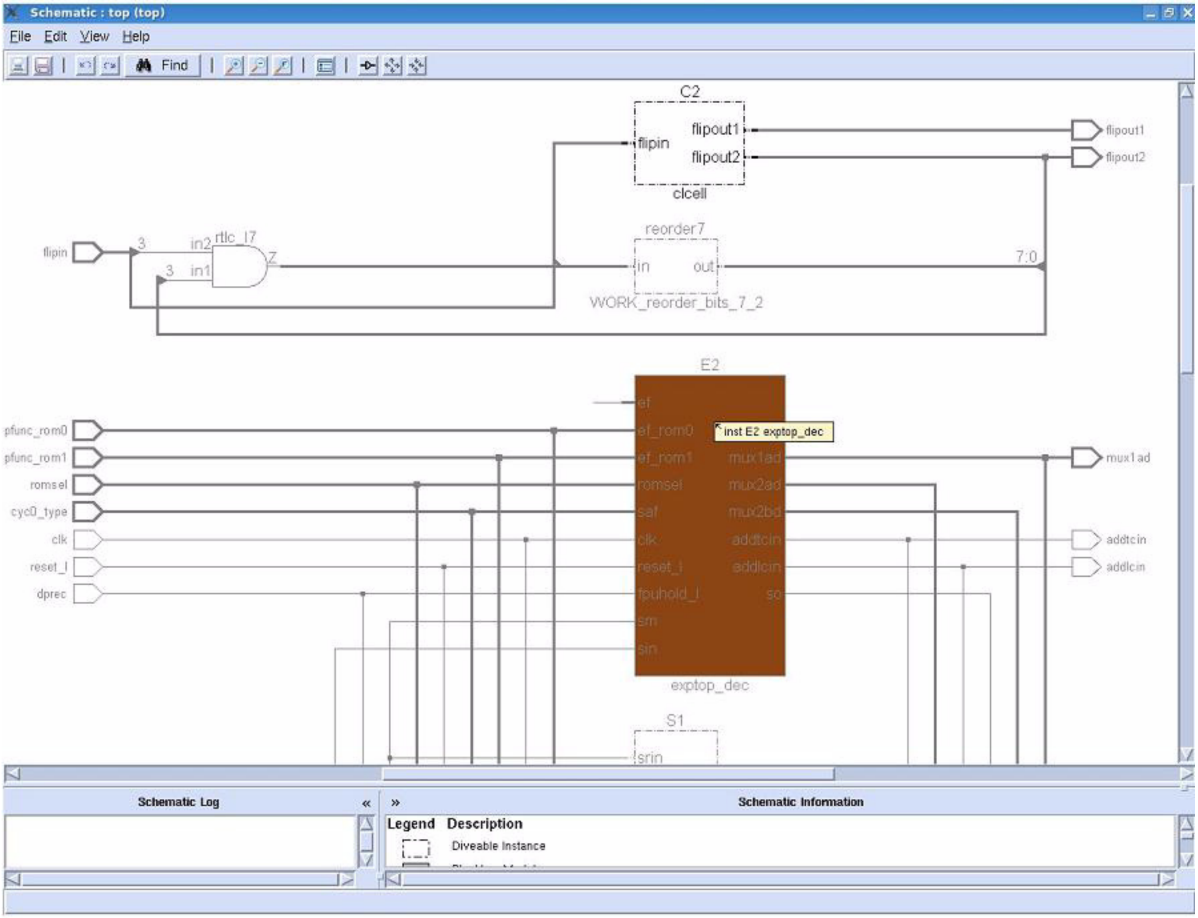


FIGURE 2. Abstract Blocks Schematic

To view the hierarchy within E2, run **Design Read** or **Run Goal** again and double-click on the E2 module in the schematic.

The following figure shows the hierarchy within E2:

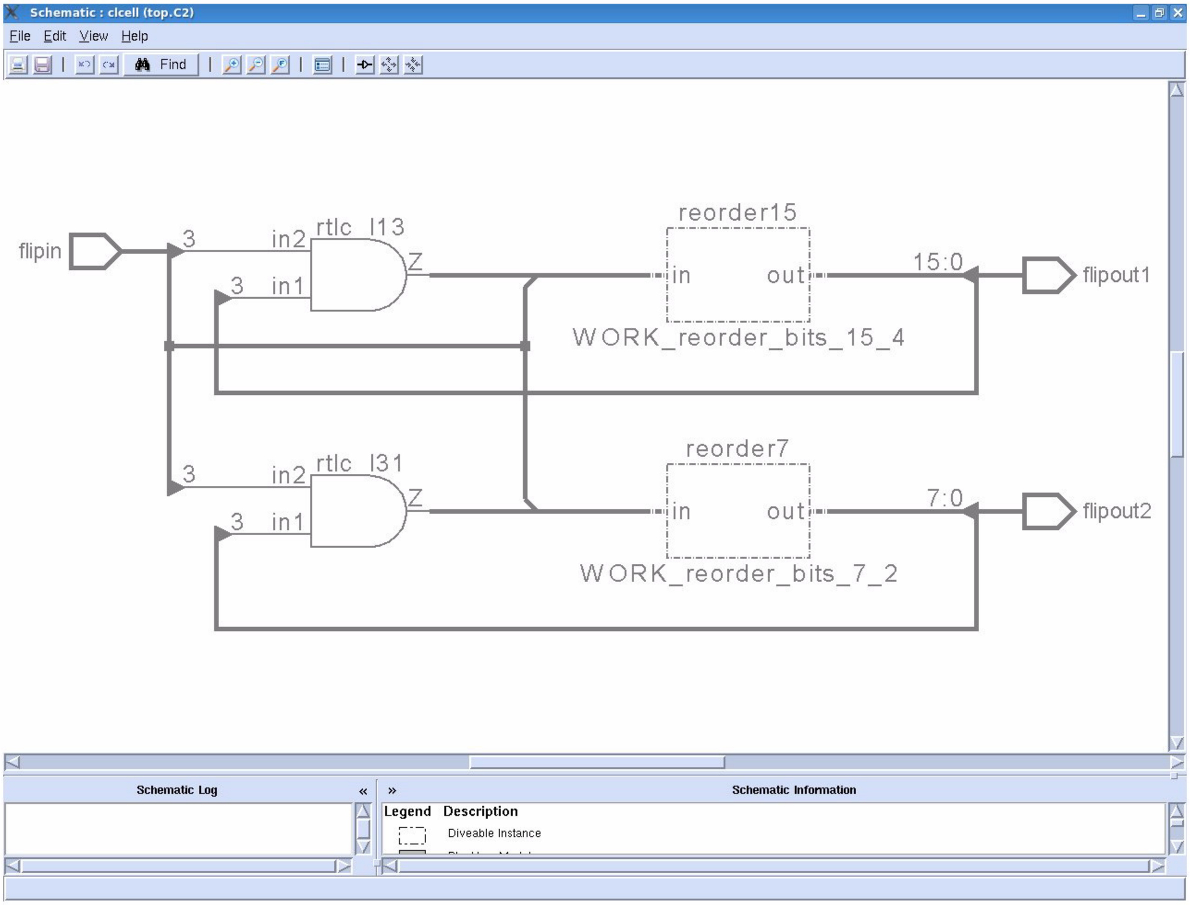


FIGURE 3. Hierarchical Schematic for E2 Module

Block Abstract Directory

Use this design read option to specify a directory in which the abstract view of a block should be saved.

By default, the directory value is set to <project_dir>/<top>/<goal>/spyglass_reports.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option block_abstract_directory <directory>
```

Examples

```
sg_shell>set_option block_abstract_directory my_abstract_out
sg_shell>current_goal soc_abstraction/lint_abstract -top top
sg_shell>run_goal
```

In the above example, an abstract view of the block for SpyGlass lint solution is generated in the following path:

```
<current_working_directory>/my_abstract_out/
spyglass_reports/abstract_view/<top>_lint_abstract.sgdc
```

Ignore Design Unit(s)

Use this design-read option to ignore the specified VHDL design units or Verilog modules during the design-read (parsing) stage.

NOTE: *For details, refer to the Ignoring Individual Design Units topic in Atrenta Console User Guide.*

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option ignoredu { <design-unit-names> }
```

When this command is run, Atrenta Console generates the *ignore_summary report*.

SpyGlass does not consider the ignorefile and ignoredu specifications, if they are specified in a source file that is passed to the set_option libhdlf <logical-library-name> <source-files> project file command.

If you specify an ignore specification by using the `ignorefile` or ignoredu commands on modules having generic parameters, the *ErrorAnalyzebbox* rule reports such modules as black boxes.

Interpret Pragma(s)

Use this design-read option to specify pragmas that should be set in your Verilog/VHDL analysis run.

For Verilog, the default pragma is set to synopsys, $s, and $S. For VHDL, the default pragma is set to synopsys, $s, and $S, and pragma.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option pragma { <list-of-pragmas> }
```

Interpreting Pragmas in Source Code

A design may contain pragmas, which are runtime commands for specific EDA tools. These commands are embedded as comments in the source code.

Following are some pragmas that SpyGlass recognizes during the design synthesis step:

translate_on	translate_off	full_case	parallel_case

By default, SpyGlass assumes that you are using pragmas from Synopsys. For example, following is the Synopsys pragma that SpyGlass automatically recognizes and interprets:

```
//synopsys full_case
```

Using Pragma Families

You can specify pragmas of other vendors. However, if you use the pragma command to define one or more pragma families, the default pragma support no longer applies. You must therefore include pragma synopsys, pragma $s, and pragma $S in the list of pragma families if you plan to use these default pragmas along with the newly defined pragma sources.

Using Single Pragma Family

To enable SpyGlass recognize a pragma family of a specific vendor, use the pragma command followed by the first word of the pragma family.

For example, to enable SpyGlass recognize the Quickturn pragma family in the test.v file under the PIC directory PIC, use the following project file commands:

```
read_file -type verilog PIC/test.v
set_option pragma quickturn
```

In this case, SpyGlass recognizes the following Quickturn pragmas:

```
//quickturn translate_off
//quickturn translate_on
```

Using Multiple Pragma Families

The following example enables SpyGlass to recognize the Quickturn and Synopsys pragma families:

```
read_file -type verilog PIC/test.v
set_option pragma quickturn
set_option pragma synopsys
```

Here, you need to enable the Synopsys pragma family because it is no longer recognized by default once you define a new pragma family (the Quickturn family in this case).

Ignoring All Pragma Families

To enable SpyGlass not recognize any pragma family, including the default Synopsys pragma family, specify the nopragma argument with the pragma command.

For example, to disable pragma recognition for the test.v file under the PIC directory, specify the following project file commands:

```
read_file -type verilog PIC/test.v
```

```
set_option pragma nopragma
```

Ignore VHDL code within pragma block 'translate'

Use this design-read option to ignore VHDL code within the pragma block, Synopsys `translate_off/translate_on`.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option hdlin_translate_off_skip_text yes
```

By default, this option is enabled. To disable this option, set the value of the *disable_hdlin_translate_off_skip_text* command to `yes`.

Ignore VHDL code within pragma block 'synthesis'

Use this design-read option to ignore VHDL code within the pragma blocks, Synopsys `synthesis_off/synthesis_on`.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option hdlin_synthesis_off_skip_text yes
```

By default, this option is enabled. To disable this option, set the value of the *disable_hdlin_synthesis_off_skip_text* command to `yes`.

Enter Macros for Analysis

Use this design-read option to specify macro definitions in your Verilog analysis run.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option define <list-of-options>
```

Purpose of Specifying Macro Definitions

A Verilog design may contain source code that should be compiled if certain conditions are met. Following is the example of such code:

```
'ifdef MacroName ...
'else ...
'endif
```

Alternatively, you can create text macros that are substituted with actual values during compilation. You can specify such values by using the *define* command.

Examples of using the define Option

Following are some examples:

- The following command sets the `State0` macro to `3`:

  ```
  set_option define {State0=3}
  ```

- The following command sets the `State0` and `State1` macros to `3` and `5`, respectively:

  ```
  set_option define { {State0=3} {State1=5} }
  ```

Set HDL Parameter(s) Value

Use this design-read option to specify parameters used during Verilog/ VHDL analysis run.

To specify parameters, perform the following steps:

1. Click ⬇ adjacent to this option. The following dialog appears:

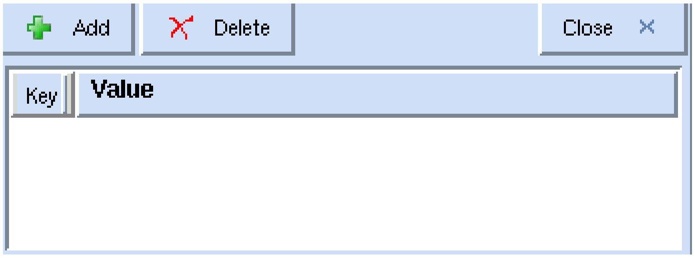


FIGURE 4. Specify HDL Parameter(s)

2. In the above dialog, click the *Add* button. A new row appears in the dialog, as shown in the following figure:

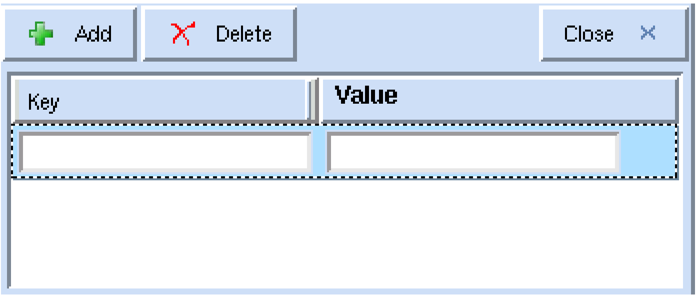


FIGURE 5. Add HDL Parameters

3. In the newly added row, specify the generic/parameter name in the *Key* column and the corresponding value in the *Value* column.
4. Repeat step 2 to set more generics/parameters.
5. Click the *Close* button to close the dialog.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option param <name>
```

In a normal design hierarchy, it is possible to define design units that can be parameterized, where the value of the parameter (in Verilog) or generic (in VHDL) is defined when the design unit is instantiated in the hierarchy.

If you then choose to run SpyGlass from the top of such design unit, this value is obviously not visible to SpyGlass. To be able to define a value for each parameter or generic, you can use this option.

Defining a Value for a Generic/Parameter

To define a value for a parameter, specify an instance of the parameter you want to define along with the value you want to set. In VHDL, entity_name.generic_name is considered while in Verilog module_name.parameter_name is considered.

The param command supports only integer constants, binary, octal, decimal, and hexadecimal. However, SpyGlass reports an error, if the parameters are overridden with a string value.

For Verilog, you can specify a parameter in the following format:

```
<width>'<base> <value>
```

Where:

- <width> and <base> are optional.
- <base> can take the following values:
  - b for binary,
  - o for octal
  - d for decimal
  - h for hexadecimal
- <value> is a valid number for the specified base.

For VHDL, you can specify a parameter in the following format:

```
<base>#<value>#
```

Where:

- <base> is optional.
- <base> can take following values:
  - 2 for binary
  - 8 for octal
  - 10 for decimal
  - 16 for hexadecimal
- <value> is a valid number for the specified base.

NOTE: *Do not override parameters with a string value.*

For example, to set the VHDL generic width in the entity control to 8, specify the following command in the project file:

```
set_option param { control.width=8 }
```

To set the Verilog parameter limit in the module block1 to 4, specify the following command in the project file:

```
set_option param { block1.limit=4 }
```

Overriding Generics in VHDL

In VHDL, you can override the following type of values:

- Integer, positive, or a natural value, as shown in the following examples:

Integer value	`set_option param { entity_name.generic_name=20 }`
Binary value	`set_option param { entity_name.generic_name=2#10# }`
Octal value	`set_option param { param entity_name.generic_name=8#76# }`
Decimal value	`set_option param { entity_name.generic_name=10#93# }`
Hexadecimal value	`set_option param { entity_name.generic_name=16#AF# }`

- Boolean/enum value, as shown in the following examples:

Boolean value	`set_option param { entity_name.generic_name true }`
Enum value	`set_option param { entity_name.generic_name red }`

- Bits, as shown in the following examples:

Bit value	`set_option param { entity_name.generic_name="0" }` Note that if double quotes around 0 are not present, the value is considered as the decimal value 0.
bit_vector value	`set_option param { entity_name.generic_name="0000" }`

- std_logic, as shown in the following examples:

std_logic	`set_option param { entity_name.generic_name="1" }` Note that if double quotes around 1 are not present, the value is considered as the decimal value 1.
std_logic_vector	`std_logic_vector set_option param { entity_name.generic_name="0000" }`

- String value, as shown in the following example:

```
set_option param { entity_name.generic_name=ABC }
```

Overriding Parameters in Verilog

In Verilog, you can override parameter values with the following type of values:

- Integer

  When you override a parameter value with an integer value or a value in a valid based number format, it is considered as overriding the parameter value with an integer value.

  Following are some examples:

Integer	`set_option param { module_name.parameter_name=123 }`
Binary	`set_option param { module_name.parameter_name=16'b10101010 }`
Octal	`set_option param { module_name.parameter_name=32'o657 }`
Decimal	`set_option param { module_name.parameter_name=16'd94 }`
Hexadecimal	`set_option param { module_name.parameter_name=32'hAB4 }`

NOTE: *If you pass a value in an invalid based number format, SpyGlass reports a violation.*

- String

  When you override a parameter with a value containing alphabets and/ or special characters (with or without numerical values), it is considered as overriding with a string value.

  Following are some examples:

```
set_option param { module_name.parameter_name=12ab3 }
set_option param { module_name.parameter_name=abc }
```

Re-definition of a Generic/Parameter at Module Instantiation

If a generic/parameter is re-defined at module instantiation, the value specified during module instantiation is given preference over the value specified by the `param` command.

For example, consider the following code (line numbers are also highlighted):

```
module sr(srin, clk, srout);
```

```
1.   parameter SIZE1 = 8;
2.   input srin, clk;
3.   output srout;
4.   reg [SIZE1:0] sr, srout;
5.     integer i;
6.   always @(posedge clk)
7.     for ( i = 0 ; i < SIZE1 ; i=i+1 )
8.     if ( i==0 ) sr[i] <= srin;
9.     else if ( i < SIZE1-1 ) sr[i] <= sr[i-1];
10.     else
11.       srout <= sr[i-1];
12. endmodule
13. module srtop(in, clk, out);
14.   input in, clk;
15.   output out;
16. sr #(32)__sr1(.srin(in),.clk(clk),.srout(out));
17. sr __sr2(.srin(in),.clk(clk),.srout(out));
18. endmodule
```

In line 16 of the above example, the value, 8, of the SIZE1 parameter is overridden with the value 32. However, for the functionality specified in line17, following cases may arise:

- If the param command is specified, SpyGlass overrides the value, 8, of the SIZE1 parameter with the specified value.
- If the param command is not specified, SpyGlass retains the value, 8, of the SIZE1 parameter.

Searches the specified paths for include files

Use this design-read option to include a relative path name of a directory that contains include files.

NOTE: *This option is available only if your design files also contain Verilog files.*

Include files contain code that defines some frequently performed actions. You can include such files at required locations in your code by using the 'include compiler directive.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option incdir <path-name>
```

NOTE: *It is recommended to use relative path names instead of absolute path names to ensure portability of include files.*

If your `incdir` directory name includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as a literal. For example, if your incdir directory name is abc*d, you need to refer to it as {{abc*d}}. However, if you want to refer to two directories, for example abc1d and abc2d, you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}}, in this case. For details on pattern matching support, see *Pattern Matching Across Features.*

SpyGlass first searches the current directory for the `'include` files. It then searches the directories specified by the `incdir` option in the sequence in which you list them. SpyGlass returns an error message and terminates if it cannot find all the required `'include` files in your directory structure.

While specifying relative path names using the `incdir` command, ensure that no duplicate file names are present in the directory list. The file corresponding to the second instance of the duplicate file name is never read because SpyGlass searches the directory structure until it finds the first instance of the file and then stops.

Specify the library files in the source design

Use this design-read option to specify Verilog library files used in a source design.

To specify library files, click ⤓ adjacent to this option. The following dialog appears:

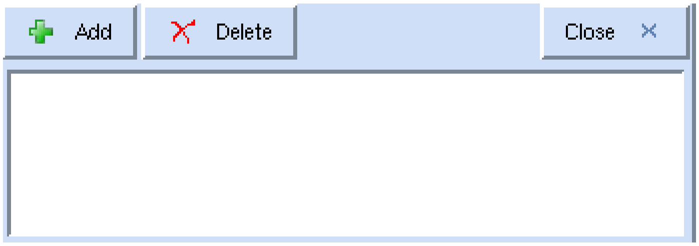


FIGURE 6. Specify Library Files

In the above dialog, click *Add* and browse to the directory containing library files. When you are done, click *Close*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option v {space-separated list of lib names}
```

Modules or User-Defined Primitives (UDPs) missing from your Verilog source code are usually present in a single library file or in files stored in a library directory. In such cases, use the *v* command (to locate the library) or the *y* command (to locate the library directory) so that SpyGlass can compile your Verilog design correctly.

SpyGlass checks the current directory for such libraries first. If it cannot find them, SpyGlass searches the path you specify with the *v*/*y* command.

Key Points

- This option is available only if your design files also contain Verilog files.
- If your filename includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your filename is "abc*d", you need to refer to it as {{abc*d}}. However, if you want to refer to two files, for example "abc1d" and "abc2d", you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see *Pattern Matching Across Features*.
- By default, SpyGlass performs rule-checking on all the cells specified by the *v* option. To disable rule-checking on such cells, specify the *ignorelibs* option.

Specify the library directories containing libraries

Use this design-read option to specify directories containing libraries to compile your Verilog design correctly.

To specify directories, click ⬇ adjacent to this option. The following dialog appears:

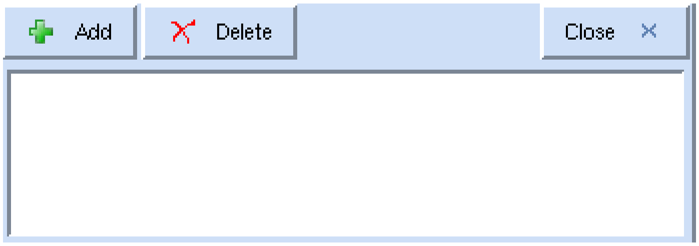


FIGURE 7. Specify Library Directories

In the above dialog, click *Add* and browse to the directory containing the required library files. When you are done, click *Close*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option y { space-separated path-names of directories }
```

NOTE: *You must also specify library file extensions using the libext command to enable SpyGlass to read library files.*

Modules or User-Defined Primitives (UDPs) missing from your Verilog source code are usually present in a single library file or in files stored in a library directory. In such cases, use the v command (to locate the library) or the y command (to locate the library directory) so that SpyGlass can compile your Verilog design correctly.

SpyGlass checks the current directory for such libraries first. If it cannot find them, SpyGlass searches the path you specify with the *v*/*y* command.

NOTE: *If your directory name includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your directory name is "abc*d", you need to refer to it as {{abc*d}}.*
However, if you want to refer to two directories, for example "abc1d" and "abc2d",

*you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see Pattern Matching Across Features.*

NOTE: *By default, SpyGlass performs rule-checking on all the cells specified by the y command. To disable rule-checking on such cells, use the ignorelibs command.*

Specify library file extensions

Use this design-read option to specify the library file extension. You can add multiple library extensions by specifying a space-separated list of extensions in the text field provided.

NOTE: *You must specify the library file extension when specifying the library file directory.*

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option libext { space-separated list of extensions }
```

Enable SystemVerilog Processing

Use this design-read option to enable or disable SystemVerilog compatibility mode. By default, this option is disabled.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enableSV <yes |no>
```

Enable auto-compilation of gateslib into sglib

Use this design-read option to compile the Synopsys liberty files (.lib files) automatically to a SpyGlass-compatible format library file (.sglib files).

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_gateslib_autocompile <yes | no>
```

If there is a pre-existing .sglib file, such .sglib file is not recompiled, if you change the value of the `set_option include_opt_data <true|false|yes|no>` command in the subsequent runs.

However, if the .sglib file is recompiled because of reasons, such as .lib files changed, the `include_opt_data` command is taken into consideration.

Allow Duplicate Module Names in Verilog Designs

Use this design-read option to allow duplicate module/UDP definitions.

Only the last found module/UDP definition is processed, and earlier definitions with the same name are ignored.

By default, duplicate module definitions result in STX_589 syntax error.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option allow_module_override <yes |no>
```

Disable Verilog 2k Processing

Use this design-read option to enable or disable Verilog 2001 compatibility mode. By default, this preference is disabled.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option disablev2k yes
```

Run in VHDL87 Compatibility Mode

Use this design-read option to enable or disable VHDL (IEEE Std 1987) compatibility mode.

By default, this preference is disabled, and Atrenta Console considers VHDL (IEEE Std 1993) as the VHDL language standard.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option 87 yes
```

Automatically Sort VHDL File(s)

Use this design-read option to sort VHDL files before analyzing them and print the sorted file order in the log file.

NOTE: *The Automatically Sort VHDL File(s) option is available only when your design files also include VHDL files for analysis.*

For a VHDL design to compile correctly, it must be analyzed in the correct order (that is, lower-level dependent design units and libraries must be analyzed before the top-level or primary units).

You can select the following sorting options from GUI:

- *DU based sort* (default): Select this option to sort the VHDL files by design units.

  Selecting this option is equivalent to the following project file command:

  ```
  set_option sortmethod du
  ```

- *Lexical*: Select this option to quickly tokenize a given set of VHDL files and identify dependency of a file on remaining files. The dependency information can then be used to build a sorted list where dependent files are compiled prior to the files using them.

  Selecting this option is equivalent to the following project file command:

  ```
  set_option sortmethod lexical
  ```

  NOTE: *Use the* `lexical` *option only when you encounter a problem with DU-based sorting. Lexical-based sorting is not able to handle dependency specified through the configuration file and so on. Therefore, VHDL sorting may not be correct is such cases.*

- *None*: Select this option if you do not want VHDL files to be sorted. When you select this option, you should specify source files in the correct order for VHDL compilation.

  Selecting this option is equivalent to the following project file command:

  ```
  set_option sort no
  ```

If you want SpyGlass to automatically sort design units, specify the following project file command:

```
set_option sort yes
```

Printing Sorted VHDL Files

To view the order of sorted VHDL files, specify the following project file command after specifying the sort command:

```
set_option print_sortorder_only yes
```

NOTE: *SpyGlass prints the order automatically in the log file when you use the* sort *command. The* print *command is retained for backward compatibility.*

Cases when the sort Command is not Used

If you do not use the sort command, you must specify source files in correct order of VHDL compilation.

The order must have design unit definitions preceding any VHDL files that use the design units, which is standard VHDL ordering. Following is the example of an explicitly defined order:

```
read_file -type hdl bottom_ent.vhd bottom_arch.vhd
upper_ent.vhd upper_arch.vhd
```

Where, the upper design unit contains instances of the bottom design unit.

Points to be Noted

Please note the following points:

- Use the sort command only if you do not know the correct design order. However, there are design configurations in which the sort command cannot reliably infer the correct order, no matter how good the sort algorithm is.
- If you use the *.vhd wild card without the sort option, SpyGlass compiles the files in the order they are returned from the shell (that is, alphabetically). The dependency order determined by SpyGlass for the design is printed in the log file.
- During sort operation, if SpyGlass is unable to determine the files to be updated, SpyGlass reports an error indicating that one or more files

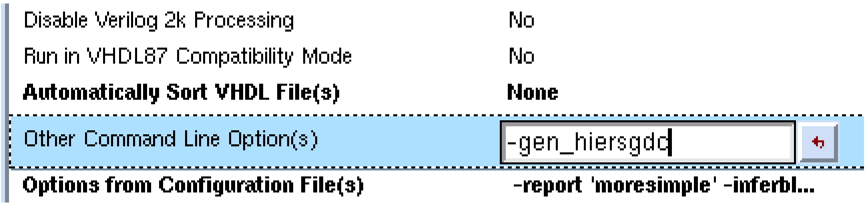
need to be recompiled. To resolve this issue, delete your work directory and any library directories and rebuild them.

- The specified VHDL libraries are checked for existence before SpyGlass attempts to determine any dependency. If any such libraries do not exist, SpyGlass reports an error and exits.

Other Command Line Option(s)

Use this design-read option to specify command-line options that you want to specify during the SpyGlass analysis run.

For example, the following figure shows the *-gen_hiersgdc* command-line option specified in the *Other Command Line Option(s)* field:



Disable Verilog 2k Processing	No
Run in VHDL87 Compatibility Mode	No
Automatically Sort VHDL File(s)	**None**
Other Command Line Option(s)	-gen_hiersgdc
Options from Configuration File(s)	**-report 'moresimple' -inferbl...**

FIGURE 8. Other Command Line Option(s)

Options from Configuration File(s)

Use this design-read option to view the options specified in the .spyglass.setup configuration file.

This field is the read-only. To modify any option, update the configuration file directly.

For details on the configuration file, see *The Configuration File in Atrenta Console*.

Advanced Design-Read Options

To view the advanced design-read options, de-select the *Show Common Options Only* check box.

Define Verbosity Level for Log File

Enables you to control the verbosity of the log file thereby ensuring that the spyglass.log file is easily readable and content is controllable.

This option can have take one of the following four values: 0, 1, 2, or 3.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option verbosity <0 | 1 | 2 | 3>
```

Following table describes the various options for the verbosity option:

Level	Output
3	Represents the output with DEBUG option set. Generates the maximum information in the spyglass.log file.
2	Represents the default output in the current scenario.
1	Represents the reasonable output
0	Represents the minimal output

Enable Abstract Blocks Schematic

Enables schematic debugging of abstracted modules. This option accepts abstracted module names, for which you want to view schematic to debug issues in the abstracted IPs of SoC.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_abstract_blocks_schematic
<abstract_module_names>
```

Enable Save Restore Flow

Use this design-read option to enable the design save-restore feature.

By default, the save-restore feature is on. To disable this feature, set the value of this option to `no`.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_save_restore <yes | no>
```

Enable Save Restore for BuiltIn Rules

Use this design-read option to view built-in messages during the restore run that were reported during the save run.

By default, built-in message that were reported in the save run are also reported in the restore run. However, if you do not want to view such messages during the restore run, set the value of this option to `no`.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_save_restore_builtin <yes | no>
```

Atrenta Console saves built-in messages (parsing, elaboration, and synthesis) reported during the save run as a part of the saved design database. These messages are then displayed during the restore run if the `enable_save_restore_builtin` option is set to `yes`.

Dump BuiltIn Rules in Precompile Flow

Use this design-read option to view built-in messages while using precompiled RTL dump. These messages are same that were reported when that precompiled RTL dump was generated.

By default, Atrenta Console reports such built-in messages when you use the precompiled dump created earlier. If you do not want to view such messages, set the value of this option to No.

Setting this field is equivalent to specifying the following command in the

Atrenta Console project file:

```
set_option dump_precompile_builtin <yes | no>
```

Built-in message reported during the generation of RTL precompiled dump are stored as a part of that dump in a file. You can later view these messages during usage of that precompiled dump if the dump_precompile_builtin option is set to yes.

NOTE: *Please note the following points:*

- 🖹 *Messages may change across SpyGlass releases. Therefore, you are recommended to precompile your libraries for each release in which you want to use the precompiled dump.*
- 🖹 *If you want some product-specific built-in checks to be reported on the usage of precompiled design units, then during the RTL precompilation step, you must set the value of the AUTOENABLE_BUILTIN_CHECKS_FOR_POLICY configuration key to an appropriate product name in the .spyglass.setup file.*

Ignore SpyGlass BuiltIn Rules

Use this design-read option to ignore certain built-in rules reported during SpyGlass analysis.

When you set this option to *Yes*, the *Directory Path Containing Ignore BuiltIn Files* option is enabled using which you can specify a file mentioning built-in rules to be ignored.

By default, built-in rules to be ignored are picked from the ignore_builtin file present under the $SPYGLASS_HOME/auxi/policy_data/spyglass/ directory.

By default, this option is set to No, and built-in rules are not ignored during SpyGlass analysis.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option ignore_builtin_rules <Yes | No>
```

Directory Path Containing Ignore BuiltIn Files

Use this design-read option to specify the path of a directory that contains built-in files.

Each built-in file contains the `-ignorerules` command to specify the built-in rules to be ignored, as shown in the following example:

```
-ignorerules builtin_rule1
-ignorerules builtin_rule 2
-ignorerules builtin_rule3
```

By default, the directory path is set to $SPYGLASS_HOME/auxi/policy_data/spyglass/ that contains the ignore_builtin file.

You can specify a different directory path containing your own built-in files for each language, such as Verilog, VHDL, and mixed. However, you must ensure that the name of all such built-in files should have the following naming convention:

ignore_builtin-<*language*>.spq

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option ignore_builtin_spqdir <path>
```

design-read Synthesis Flavor

Use this design-read option to specify the type of synthesis you wish to perform during the design-read process.

Atrenta Console considers this option only if you enable synthesis during the design-read process. To enable synthesis, select the *Synthesize Netlist* option under the *Run design-read* tab or specify the following command in the project file:

```
set_option designread_enable_synthesis yes
```

You can specify the type of synthesis by specifying the following command in the project file:

```
set_option designread_synthesis_mode <mode>
```

Where, <mode> can accept any of the three values, as specified in the following table:

Mode	Description
base	Executes the CLASSIC mode rules only (which run on non-optimized netlist) and disables rules of EST and ESYNTH mode.
opt	Executes the ESYNTH mode rules only (which run on optimized netlist) and disables rules of CLASSIC and EST mode.
techmap	Executes the EST mode rules only (which run on optimized and technology-mapped netlist) and disables rules of CLASSIC and ESYNTH mode.

Goal Run Synthesis Flavor

As part of the goal run, the synthesis mode is determined by the rules in `<goal>.spq` file.

If the synthesis modes are conflicting, spyglass FATALs out with the following message:

```
ERROR [294] Rules corresponding to multiple synthesis mode
selected in current run.
```

If above error is flagged, then the given rule set is not compatible, and you need to run the compatible product/rule set together.

Though majority of the product's rules use classic synthesis, the products that use other modes are mentioned below:

- *SpyGlass Power Estimation and Reduction*: This product uses the EST mode. The rules of this product are compatible with the following modes:
  - ❐ base
  - ❐ opt
  - ❐ optimized and techmap
  - ❐ All of the above
- *SpyGlass SEC*: This product uses the ESYNTH mode by default.
- *SpyGlass DFT and SpyGlass DFT DSM:* These products use classic mode by default. Most of their rules are compatible with base and opt mode, and only a few are compatible with just opt mode.

The behavior of multi-mode rule is driven by the mode of the other rules it is run with. For example, consider a rule, R1, is compatible with Classic and Esynth synthesis modes and rule, R2, is compatible with Esynth and techmap synthesis mode. Also, consider that R1 and R2 are run together, then the run proceeds with the Esynth synthesis mode since it is common to both the rules.

Enable Incremental Mode for All Goals

Use this design-read option to enable reporting of incremental messages so that you can compare results of a previously run goal with the current goal.

By default, this option is set to *No*, and Atrenta Console does not perform incremental reporting of messages.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option report_incr_messages <Yes | No>
```

Logical Working Directory

Use this design-read option to specify a logical working directory for compilation of Verilog/VHDL libraries.

By default, the working directory is set to the WORK directory in the current directory.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option work <value>
```

After setting this option, ensure that you define a logical library of the same name in the *HDL Libraries* section under the *Add Design Files* tab.

For example, to use the working library `work2` present in the my_work_lib directory, specify the following commands:

```
set_option work work2
set_option lib work2 /u/<user_name>/my_work_lib
```

NOTE: *Libraries are platform-specific. For example, you cannot compile on Sun platform, use a library on Linux platform. If you switch platforms, you need to delete the working library and other user-defined VHDL libraries and rebuild them. Atrenta supplies versions of all four default libraries for all supported platforms. These should not need to be rebuilt.*

Stop Directory(s)

Use this design-read option to specify directories that you want to skip for rule-checking during SpyGlass analysis.

To specify a directory that you want to skip for rule-checking, perform the following steps:

1. Select the *Stop Directory(s)* option. The following text box appears in the *Value* column:

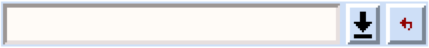


FIGURE 9. Stop Directory(s)

2. Click ⬇. This displays the following dialog:

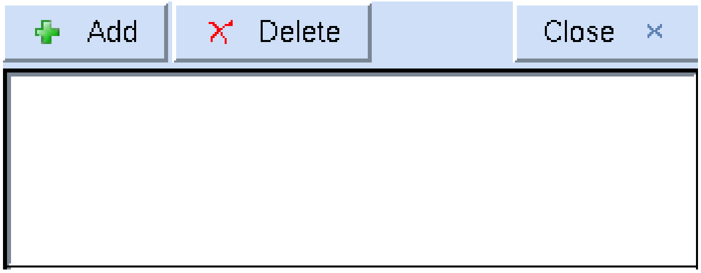


FIGURE 10. Add Directories

3. In the above dialog, click the *Add* button. This displays the *Select Directory* dialog.
4. In the *Select Directory* dialog, select the directory that you want to skip from rule-checking.
5. Click the *OK* button to close the *Select Directory* dialog.

After performing the above steps, the specified directory is not considered for rule-checking.

If a file added under the *Add Design Files* tab exists in the directory selected by performing the above steps or if you later add a file from this directory, the 🖐 icon appears before the name of that file to indicate that this file is not considered for rule-checking.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option stop_dir <name>
```

Upper Threshold for Compiling Memories

Use this design-read option to set an upper threshold for compiling memories. You must specify a positive integer value in this field. By default, the value of this option is set to 4096.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option mthresh <value>
```

If the individual net/variable size (bit-count) is more than the specified threshold, the module is not compiled and is treated as a black box and the SYNTH_ERROR[5273] message is reported. To solve the problem, specify the threshold value greater than or equal to the value displayed in the error message. However, this higher value may not be feasible due to system memory size limitations.

NOTE: *If handlememory option is specified, the individual memory size is reduced memory size.*

Synthesizing memories (2-dimensional arrays) in RTL designs has always been a resource- and time-consuming task; some designs even run out of system memory. When the memory is synthesized, each bit location of the synthesized memory is represented by a flip-flop or latch in the synthesized netlist. This can easily consume an appreciable amount of system memory, resulting in design capacity problems.

To control compilation of memories in a module, SpyGlass does not compile modules where the total bit-count (post-elaboration) of the memories in a module exceeds the value specified in this field.

Enable Handle Memories

Use this design-read option to enable processing of memories in an optimized manner. By default, the value is set to *No*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option handlememory <yes | no>
```

Enable HDL Encryption

Use this design-read option to enable the encryption of VHDL/Verilog libraries during compilation. By default, the encryption feature is disabled.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_hdl_encryption <yes | no>
```

By default, RTL rule-checking on encrypted precompiled design units is enabled. Any highlighting information within such modules is shown on the library module boundary only. You can disable RTL rule-checking on such design units by setting the *Disable Encrypted HDL Checks* option to `yes`.

NOTE: *When you specify the* `enable_hdl_encryption` *command, a precompile dump is created on both 32-bit and 64-bit platforms irrespective of the platform on which SpyGlass is run. By default, the dump_all_modes command is enabled with the* `enable_hdl_encryption` *command.*

Disable Encrypted HDL Checks

Use this design-read option to disable RTL rule-checking on encrypted design units.

When you set this option to *Yes*, SpyGlass internally removes messages coming from post-flattening rules that do not provide any highlighting information in the schematic.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option disable_encrypted_hdl_checks <yes | no>
```

NOTE: *Please note the following points:*

- *The Enable RTL Checking of Precompiled HDL Libraries option does not have any effect on the above rule-checking behavior for encrypted modules.*
- *All design units instantiated in an encrypted design unit are treated as encrypted even if they are not encrypted.*
- *Encryption using any robust encryption algorithm is outside the scope of SpyGlass.*

Enable Analysis of Instantiated DesignWare Components

In case your design has instances of Synopsys® DesignWare® components, set this option to map these components in terms of technological gates.

By default, this option is set to *No*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option dw <yes | no>
```

You can also use the *dw_options* command to specify multiple options for the DesignWare components.

The following table explains the results when different values are specified for the *dw* and *dw_options* commands:

Value of the dw option	**Value of the dw_options option**	**SpyGlass Behavior**
yes	{hide_all_dw_violations}	It waives all violations across all rules using the waive –ip -delete_internal_use_only command and doesn’t report anything in the waiver report
yes	{report_violations, enable_waiver} OR {enable_waiver} OR {enable_waiver, report_violations }	Enables DW waivers

Hierarchical SGDC Modes

Use this design-read option to specify a hierarchical SGDC flow. You can specify the required flow by selecting any of the following options from the adjacent drop-down list.

- *Generation Mode*

  Selecting this option enables hierarchical migration of block-level SGDC commands to the chip-level. Setting this field to the value, *Generation Mode*, is equivalent to specifying the following command in the Atrenta Console project file:

  ```
  set_option gen_hiersgdc yes
  ```

- *Validation Mode*

  Selecting this option enables validation of hierarchically migrated block-level SGDC commands to the chip-level. Setting this field to the value, *Validation Mode*, is equivalent to specifying the following command in the Atrenta Console project file:

  ```
  set_option validate_hiersgdc yes
  ```

- *None*

(Default) Selecting this option means that none of the flow is specified.

In this case, the *Synthesize Netlist* option (under *Run design-read* tab) is enabled. You can select this option to enable synthesis during the design-read process.

By default, the *Synthesize Netlist* option is disabled during *Generation Mode* and *Validation Mode* as synthesis always occurs in these modes.

Maximum Messages Per Rule

Use this design-read option to set an upper limit for messages to be reported per rule.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option lvpr <value>
```

Cache Directory

Use this design-read option to specify a cache directory where you want library compilation to be performed.

By default, the value of this option is set to `spyglass_cache`.

SpyGlass considers <cwd>/spyglass_cache as the default cache directory if you leave this field blank.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option cachedir <directory-name>
```

Exit on Detecting Blackboxes in the Design

Use this design-read option to force SpyGlass to stop analysis and exit if a black box is found in a design.

By default, the option is set to *No*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option nobb yes
```

Enable RTL Checking of Precompiled HDL Libraries

Use this design-read option to perform design-read checks (including lexical checks) on precompiled HDL libraries.

By default, this option is set to *no*.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option hdllibdu yes
```

Check IP

Use this design-read option to specify design units on which rule-checking should be done. For the rest of the design units, SpyGlass skips rule-checking.

When you specify this command, SpyGlass not only considers the design unit specified by this command, but also considers all those design units starting from the top in the hierarchy till the design unit specified by this command.

For example, consider the design hierarchy shown in the following figure:

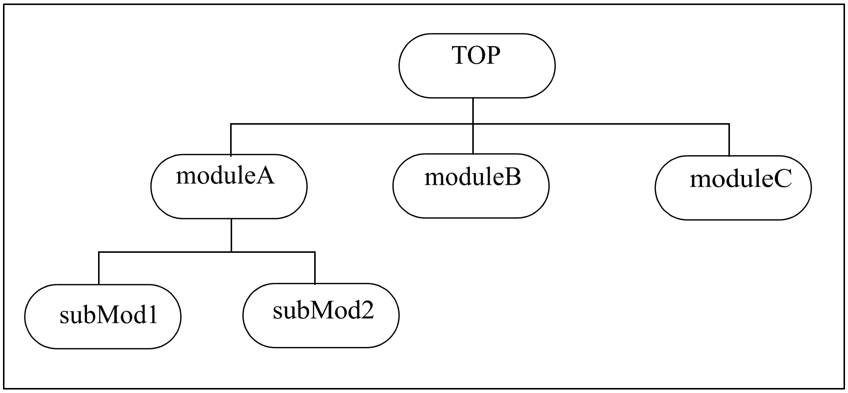


FIGURE 11. Sample Design Hierarchy

Now, if you set the `checkip` command to `ModuleA`, SpyGlass synthesizes the modules, `TOP`, `ModuleA`, `subMod1`, and `subMod2`, and consider only these modules for rule-checking.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option checkip <ip-name>
```

Check DU

Use this design-read option to specify a design hierarchy for which rule-checking should be done for interconnects.

All design units instantiated under the design unit specified by this option are treated as grey boxes. For the rest of the design units, SpyGlass skips rule-checking.

When you specify this option, SpyGlass not only considers the design unit specified by this option for rule-checking, but also considers all those design units starting from the top in the hierarchy till the design unit specified by this option.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option checkdu <du-name>
```

Consider the design hierarchy shown in the following figure:

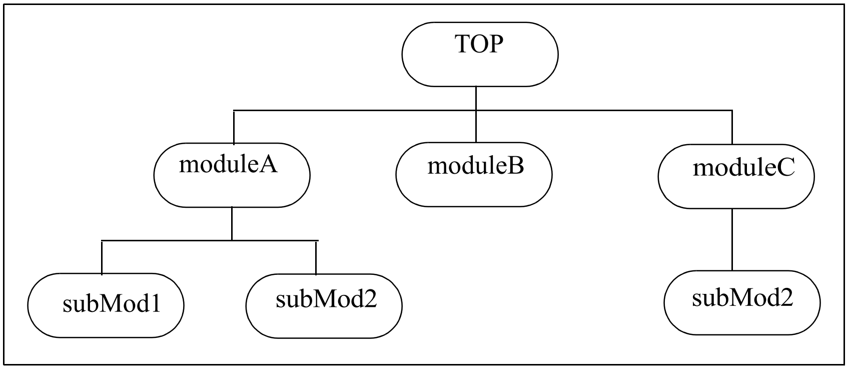


FIGURE 12. Sample Design Hierarchy

Now, if you set the `checkdu` option to `moduleA`, SpyGlass synthesizes the modules, `TOP` and `ModuleA`, and considers only these modules for rule-checking.

Consider another design hierarchy, as shown in the following figure:

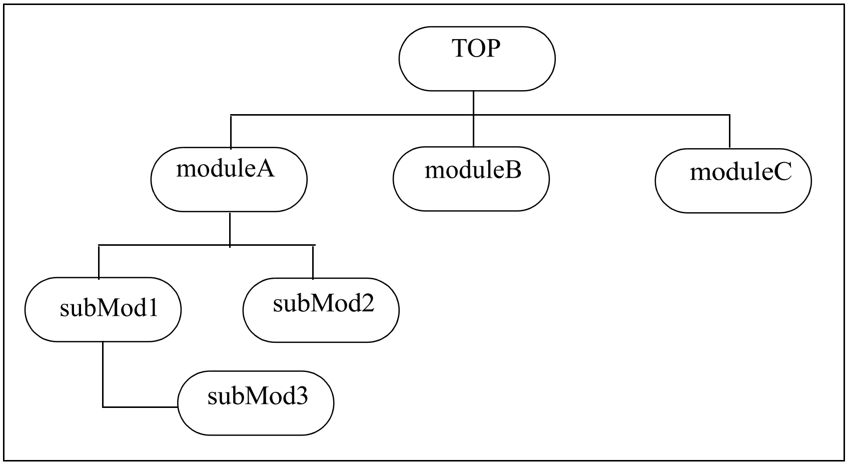


FIGURE 13. Sample Design Hierarchy

Based on the above design hierarchy, consider that you specify the following commands:

```
set_option checkdu moduleA
set_option checkip subModule1
```

In this case, SpyGlass synthesizes the modules, `TOP`, `moduleA`, `subModule1`, and `subModule3`, and consider only these modules for rule-checking.

NOTE: *The checkdu option is given preference over the checkip option, if these options are specified for the same design unit.*

Please note the following points for the `checkdu` and `checkip` options:

- The *stop*, *stopfile*, and *stopdir* options are given preference over the *checkdu* and *checkip* options. For example, SpyGlass ignores the *checkdu* and/or *checkip* options on design units for which the `stop` option is specified.
- If a design unit specified with the *checkdu* and *checkip* options contains an overlapping spanning tree (hierarchical tree rooted at that design unit), the option with the design unit at a higher hierarchical level (in the tree) is given higher priority.

The *checkip* and *checkdu* options cannot be used to skip synthesis and rule-checking on library cells. All library cells used inside the design units specified by the *checkip* and *checkdu* options are also considered a part of the reduced design and are considered for rule-checking.

Enable SDC-to-SGDC translation

Use this design-read option to translate design attributes from SDC format to SGDC format. These attributes are then used during SpyGlass analysis.

By default, the output of SDC-to-SGDC translation is saved in the file sdc2sgdc_*<mode>*.sgdc.*<pid>* in the $CWD/*<vdb-name>*_reports/sdc2sgdc directory.

NOTE: *By default, the sdc2sgdc generated constraint files are retained in the subsequent SpyGlass run. Set the retain_old_sgdc parameter to no to remove the SGDC file generated in the previous SpyGlass runs.*

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option sdc2sgdc <yes | no>
```

Specify the mode of the SDC file to be translated to SGDC

Use this design-read option to specify a mode of the SDC file to be translated to an SGDC file.

Consider an example in which you specify two different modes in an SGDC file, as shown below:

```
sdc_data -file one.sdc -mode one
sdc_data -file two.sdc -mode two
```

Now, under the *Set Read Options* tab, if you specify the mode as *one*, only SDC data given under the mode, *one*, is translated. As a result, only one.sdc file is converted into SGDC.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option sdc2sgdc_mode <mode-name>
```

Specify the file to save output of SDC-to-SGDC translation

Use this design-read option to specify a file to save the output of SDC-to-SGDC translation.

If the file specified in this field already exists, Atrenta Console overwrites the existing file.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option sdc2sgdcfile <file-name>
```

Dump SDC Generated Clocks in SGDC

Use the `sdc_generated_clocks` parameter to dump the generated clocks having same domain in an uncommented form in the sgdc file.

By default, the generated clocks having the same domain, if not given on black box, are dumped in a commented form in the sgdc file.

Set the value of the `sdc_generated_clocks` parameter to `yes` to dump all the generated clocks in an uncommented form in the sgdc file.

Used by	SDC2SGDCPARSE
Options	yes, no
Default value	no
Example	
Console/Tcl-based usage	`set_parameter sdc_generated_clocks yes`
Usage in goal/source files	`-sdc_generated_clocks=yes`

Define a Mode to Infer Domains from SDC

You can use the `sdc_domain_mode` parameter to specify the mode of domain inference. By default, the domain inference mode is `sta_compliant`. In this mode, the clock domains are extracted as per the following guidelines:

- A path verified by STA implies that the source and destination have the same domain and therefore the clocks in such a path will be assigned the same domain. SpyGlass CDC therefore does not verify such paths.
- A path not verified by STA implies that the source and destination have different domains and therefore the clocks in such a path will be assigned different domains. SpyGlass CDC therefore verifies such paths.

NOTE: *If none of the* set_clock_group*,* set_clock_uncertainty*,* set_false_path *constraints is specified for a clock pair, they are considered synchronous.*

The following table lists the additional details about the sdc_domain_mode parameter:

Used by	sdc2sgdc flow
Options	sta_compliant, pessimistic, strict, async, sta_scg, strict_sta
Default value	sta_compliant
Example	
Console/Tcl-based usage	`set_parameter sdc_domain_mode strict`
Usage in goal/source files	`-sdc_domain_mode=strict`

When the mode is set to strict, you must provide all clock relationships in a single set_clock_groups command. SpyGlass reports a FATAL error if domain is not inferred.

When the mode is set to pessimistic, the behavior is similar to that of sta_compliant with the exception that if none of the set_clock_group, set_clock_uncertainty, or set_false_path constraint is specified for a clock pair, they are considered asynchronous.

NOTE: *The values, strict and pessimistic, will be deprecated in a future release.*

When the mode is set to async, then no domain inference from sdc constraints is done and clocks are considered asynchronous to each other.

When the mode is set to strict_sta or sta_scg, only the user-specified set_clock_group command would be considered and all the other commands related to domain computation are ignored. Also, a spreadsheet (.csv) file showing clock relationship is generated.

NOTE: *The value, strict_sta, for the sdc_domain_mode parameter is deprecated and will be removed in a future release.*

Inferring cdc_false_path for Clocks in Different Domains

Use the *sdc_generate_cfp* option to infer *cdc_false_path* in case we have different domains that were assigned to the clocks but no asynchronous relationship was specified between these clocks.

Used by	SDC2SGDCPARSE
Options	yes, no
Default value	no
Example	
Console/Tcl-based usage	`set_option sdc_generate_cfp yes`
Usage in goal/source files	`-sdc_generate_cfp=yes`

For example, consider the following SDC declaration:

```
create_clock -name Clk1 -period 10.00 in1
create_clock -name Clk2 -period 10.00 in2
create_clock -name Clk3 -period 10.00 in3
create_clock -name Clk4 -period 10.00 in4

set_clock_group -asynchronous -group { Clk1 } -group { Clk3 }
```

Following corresponding SGDC commands are generated when you set the value of the *sdc_generate_cfp* option to yes:

```
cdc_false_path -from Clk1 -to Clk2
cdc_false_path -from Clk1 -to Clk4
cdc_false_path -from Clk2 -to Clk1
cdc_false_path -from Clk2 -to Clk3
cdc_false_path -from Clk3 -to Clk2
cdc_false_path -from Clk3 -to Clk4
cdc_false_path -from Clk4 -to Clk1
cdc_false_path -from Clk4 -to Clk3
```

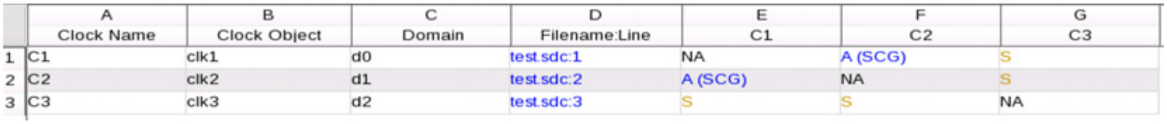
Capturing Domain Inferring Results

The following rules capture the domain inferring results:

- `Domain_Missing01`: Reports clocks for which no domain relationship could be inferred from SDC commands. The domain assigned to these clocks depends on the mode specified in the `sdc_domain_mode` parameter. The severity of this rule is Error by default. In case of `strict` mode, it reports a FATAL violation.
- `Domain_Conflict01`: Reports conflicts found during domain inference. The severity of this rule is Error by default. In case of `strict` mode, it reports a FATAL violation.
- `Domain_Matrix01`: Generates spreadsheet to show clock relationships and the inferred domain depending upon the mode specified in the `sdc_domain_mode` parameter. It is an informational rule. For example, consider the following input in the sdc file:

```
create_clock -name C1 -period 10 { clk1 }
create_clock -name C2 -period 10 { clk2 }
create_clock -name C3 -period 10 { clk3 }
set_clock_group -asynchronous -group { C1 } -group { C2 }
```

  For the above input, the spreadsheet generated by the `Domain_Matrix01` rule when the `sdc_domain_mode` parameter is set to `sta_compliant` and the `sdc_generate_cfp` parameter is set to `yes`, is shown in the following figure.



	A Clock Name	B Clock Object	C Domain	D Filename:Line	E C1	F C2	G C3
1	C1	clk1	d0	test.sdc:1	NA	A (SCG)	S
2	C2	clk2	d1	test.sdc:2	A (SCG)	NA	S
3	C3	clk3	d2	test.sdc:3	S	S	NA

FIGURE 14. Domain_Matrix01 Spreadsheet

- *If you generate the **cdc_false_path** constraint through the **sdc_generate_cfp** command, SpyGlass CDC ignores the asynchronous crossings for the paths specified in the **cdc_false_path** constraints.*

Changing the Default Hierarchy Separator of the SDC2SGDC Constraints

Use the `use_hier_sep_slash` parameter to change the default hierarchy separator of the SGDC constraints.

By default, the hierarchy separator used in SGDC constraint's name field is '.'

Example

Consider the following `create_clock` command in SDC.

```
create_clock -name "CLK" -add -period 10.0
-waveform {0.0 5.0} [get_pins buf_cts/clkout]
```

The above command is converted into following SGDC clock using the default hierarchy separator, '.', as shown below:

```
clock  -name "test_top.buf_cts.clkout"  -domain CLK
-edge { 0.000000 5.000000}  -period 12 -tag CLK
```

When the value of the `use_hier_sep_slash` parameter is set to `yes`, the same clock is converted to the following SGDC clock:

```
clock  -name "test_top/buf_cts/clkout"  -domain CLK
-edge { 0.000000 5.000000}  -period 12 -tag CLK
```

Handling False Paths

The *false_path* constraint is generated in the SGDC file when the following commands are specified:

- **set_false_path**

  Consider the following SDC commands:

  ```
  set_false_path -from [get_pins f1/q] -through [get_pins
  A2/out] -to [ get_pins f2/in]
  set_false_path -from [get_pins f1/q] -through [get_pins
  A2/out] -to [ get_pins f2/in]
  ```

  The following false_path constraints are generated corresponding to the above SDC commands:

  ```
  false_path -from f1/q -to f2/in -through A2/out  -type sfp
  false_path -from f1/q -to f2/in -through A2/out  -type sfp
  ```

- **set_clock_group**

  Consider the following SDC commands:

  ```
  set_clock_group -logically_exclusive
  set_clock_group - physically_exclusive
  set_clock_group - asynchronous
  ```

  The following false_path constraints are generated corresponding to the above SDC commands:

  ```
  false_path -scg_logically_exclusive
  false_path -scg_physically_exclusive
  false_path -scg_asynchronous
  ```

Handling Multi-cycle Paths

The sg_multicycle_path constraint is generated in the sgdc file when the set_multicycle_path command is given.

Consider the following SDC commands:

```
set_multicycle_path  -from [get_pins f1/q] -through [get_pins
```

```
A2/out] -to [ get_pins f2/in] 2
```

The following sg_multicycle_path constraint is generated corresponding to the above SDC commands:

```
sg_multicycle_path -from f1/q -to f2/in -through A2/out
-path_multiplier 2
```

Handling Mutually Exclusive Clocks

The cdc_false_path constraint is generated in the sgdc file when the -logically_exclusive option is given with the set_clock_groups command.

Consider the following SDC commands:

```
create_clock -name Clk1 -period 10.00 in1
create_clock -name Clk2 -period 15.00 in2
set_clock_group -logically_exclusive
-group{ Clk1 } -group { Clk2 }
```

The following cdc_false_path constraints are generated corresponding to the above SDC commands:

```
cdc_false_path -from Clk1 -to Clk2
cdc_false_path -from Clk2 -to Clk1
```

Handling of Directional Clocks

The cdc_false_path constraint is generated in the sgdc file when the set_clock_uncertainity command is used but the clock is inferred as asynchronous.

Consider the following SDC commands:

```
create_clock -name Clk1 -period 10.00 in1
create_clock -name Clk2 -period 15.00 in2
set_clock_groups -asynchronous
-group { Clk1 } -group { Clk2 }
set_clock_uncertainty -from Clk1 -to Clk2
```

The following cdc_false_path constraint is generated corresponding to the above SDC commands:

```
cdc_false_path -from Clk1 -to Clk2
```

Translating set_clock_sense command

The sdc *set_clock_sense* SDC command is converted to the *clock_sense* SGDC command during the sdc2sgdc flow.

Currently, the *set_clock_sense* command is translated only when you specify the -*stop_propagation* option.

For example, consider the following SDC command:

```
set_clock_sense -stop_propagation -clocks Clk2  [get_pins
orinst1/Z ]
```

Following is the converted SGDC command:

```
clock_sense -pins "top.orinst1.Z" -tag  Clk2
```

Translating set_disable_timing command

The set_disable_timing SDC command is converted to the *disable_timing* SGDC command when:

- Values for the -from and -to fields of the command is specified
- Object list in the command is a lib cell module or lib cell instance

For example, consider the following SDC command:

```
set_disable_timing -from A -to Z [ get_lib_cells lsi_10k/OR2
]
```

Following is the converted SGDC constraint:

```
- disable_timing -name OR2 -from A -to Z
```

Translating set_mode command

The *set_mode* SDC command in translated into the *set_lib_timing_mode* SGDC command, when the type of *set_mode* is cell.

For example, consider the following SDC command:

```
set_mode -type cell mode1 U1
```

Following is the converted SGDC constraint:

```
set_lib_timing_mode -modes mode1 -instances U1
```

Specify the manner in which virtual-to-real clock mapping to be done

Use this design-read option to specify a manner in which virtual to real clock-mapping should be done.

If you set this field to *No* or do not specify any value in this field, a traversal method is used in which involves finding clocks in the fan-in/fan-out of output/input delays by matching clock characteristics, that is, period, waveform, followed by name-mapping.

Set this field to *Yes* to implement name mapping only.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option mapVirtualClkByName <yes | no>
```

Specify parameter to give the list of suffix strings

Use this design-read option to specify a list of suffix strings used for mapping virtual clocks to real clocks.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option mapSuffixList <comma-separated-list>
```

Consider the following example:

```
set_option mapSuffixList virtual
```

In this case, `CLKAvirtual` maps to `CLKA`.

If you do not specify this command or if none of the real clocks satisfies the matching criteria for the concerned virtual clock name, no mapping is done. However, virtual clocks are still dumped, but with their actual names only.

Specify parameter to give the list of prefix strings

Use this design-read option to specify a list of prefix strings used for mapping virtual clocks to real clocks.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option mapPrefixList <comma-separated-list>
```

Consider the following example:

```
set_option mapPrefixList virtual
```

In this case, `virtualCLKA` maps to `CLKA`.

If you do not specify this command or if none of the real clocks satisfies the matching criteria for the concerned virtual clock name, no mapping is done. However, virtual clocks are still dumped, but with their actual names only.

Enable the physical aware power estimation flow

Use this design read option to enables Physical Aware Power Estimation flow and check out required licenses.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option enable_physical_aware_pe <yes|no>
```

If you do not specify this option, Spyglass runs normally.

Path (physical_dbdir) where design database is saved

Use this design read option to specify directory path to the design database for save & restore runs in the Physical Aware Power Estimation flow.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option physical_dbdir <directory_path>
```

If you do not specify this option, SpyGlass uses default location for the design database Save and Restore runs.

Is the current design a netlist design

Use this design-read option to specify an input design as a netlist design that contains no behavioral construct.

When you set this option to *Yes*, certain runtime optimizations are enabled during elaboration.

If this option is set to *Yes* and the design is not a netlist design, SpyGlass skips the behavioral constructs, and synthesizes only the structural constructs.

A netlist design can only contain the following:

- Instantiations of other modules or library cells
- Simple assign statements

This option is not allowed when any of the following options are specified:

- `enable_mmdelete/higher_capacity`

  This set enables module-by-module deletion that is not allowed for a netlist design.

- `enable_precompile_vlog/libhdlfiles/libhdlf`

  This set allows precompile dump for Verilog designs, and it is not supported with this option.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option netlist <yes | no>
```

The default value of this options is no. Set the value of this option to yes to specify an input design as a netlist design.

Reports Max Count Size

Use this design-read option to specify the maximum number of messages for sorted reports (simple, moresimple, and waiver reports).

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option report_max_size <value>
```

Reports Name

Use this design-read option to specify the name of the report to be generated.

The specified report is written to the <report-name>.rpt file (default name) in the current working directory. To specify a different file name and location, use the `reportfile` project file command.

NOTE: *If you generate the same report again with the same settings, the existing report file is overwritten by the new information.*

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option report <name>
```

For example, the following command generates the summary report for the current design:

```
set_option report { "summary" }
```

The following example generates the summary and inline report for the current design:

```
set_option report { {summary} {inline} }
```

Report File

Use this design-read option to specify the name and location of the report file.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option reportfile <file-name>
```

For example, to send results from the SpyGlass review of your design file to the file myoutput, which you intend to edit later, use the following command:

```
set_option reportfile myoutput
set_option report simple
```

NOTE: *If you generate the same report again with same settings, the existing report file is overwritten by the new information.*

Report Style

Use this design-read option to specify the style in which you want to customize a report.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option report_style <style>
```

Where, *<style>* can accept any of the following values:

```
[flat | grouped] | [display_msgid | hide_msgid] |
[display_rulegroup | hide_rulegroup] | display_sdcgroup |
display_taggroup
```

NOTE: *You can specify only one value from each set of values (mentioned above) in a given run.*

The following table describes the purpose of each of the above values:

Value	**Description**
flat	Displays the report in an ungrouped format
grouped	Groups the content of the report (for example, by goals)
display_msgid	Enables the display of the message index column in the reports
hide_msgid	Hides the message index column in the reports
display_rulegroup	Allows grouping of rule messages in the reports by rule group
hide_rulegroup	Disallows grouping of rule messages by rule group in the report

Value	Description
display_sdcgroup	Groups messages of the schema-based rules in the SpyGlass Constraints solution based on the SDC schema specified in the SGDC file. This option is enabled by default.
display_taggroup	Groups messages of the *Ac_sync_group* rules of the SpyGlass CDC solution based on instance names or user-specified names

The default values of the `report_style` command are `grouped`, `display_msgid`, and `hide_rulegroup`.

If you specify conflicting values to this command, only the last specified value is considered. For example, consider that you specify conflicting values, as shown in the following command:

```
set_option report_style { hide_msgid display_msgid }
```

In the above example, the `display_msgid` value is considered.

Aggregated Report(s) Name

Use this design-read option to specify the type of aggregate reports (project_summary, datasheet, or dashboard) to be generated.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option aggregate_report {<report-names>}
```

Specify Configuration File to be Used to Create the Report

Use this design-read option to specify a configuration file that contains a list of projects and run directories generated by batch console or GUI. This data is used to generate the specified aggregate report.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option aggregate_report_config_file <config-file-path>
```

Report Output Directory Path

Use this design-read option to specify a directory in which you want to generate data for the specified aggregate report.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option aggregate_reportdir <report-directory-path>
```

Disable HTML Reports generation

Use this design-read option to disable the auto-generation of the specified reports. You can specify 'datasheet', 'dashboard' and/or 'html' to stop auto-generation of either of the reports.

Setting this field is equivalent to specifying the following command in the Atrenta Console project file:

```
set_option disable_html_report {datasheet dashboard}
```

Special Features in Atrenta Console

Memory Reduction Feature

When a memory is synthesized, each bit location of the synthesized memory is represented by a flip-flop or latch in the synthesized netlist. This can easily consume an appreciable amount of system memory, resulting in design capacity problems.

Synthesizing memories (two-dimensional arrays) in RTL designs has always been resource and time-consuming task. Some designs may even run out of system memory.

To solve this problem, SpyGlass provides an optional memory reduction feature for both Verilog and VHDL designs. You can use this feature when you are not able to read in the design because of large memories.

The memory reduction feature enables you to perform rule-checking on large memory modules within limited system resources. By using this feature, SpyGlass can analyze designs containing large two-dimensional arrays and still fully analyze around and through such arrays.

Specify the following command in the project file to enable the memory reduction feature:

```
set_option handlememory yes
```

Impact of Using this Feature

When the memory reduction feature is enabled, there will be some differences in the messages reported, as discussed below:

- The following rules will not report violations in some cases:

SpyGlass lint solution	W111	W456	W488
SpyGlass OpenMore solution	NotInSens	NotReqSens	

- The following rule will report violations in some cases:

SpyGlass lint solution	W175	W415
SpyGlass DFT solution	Topology_05	

- Violation message of some rules, such as BitPartInLHS rule may change in some cases.
- Some rules, such as DeadCode will report violation when the designs were earlier passing the rule-check and vice versa.
- Some rules, such as SYNTH_5243 do not report violation to suppress unwanted noise.

Apart from the above rules being affected, there are cases where rule messages will be generated only when the memory reduction feature is enabled as the memory modules are now being synthesized. In some other cases, number of rule messages will decrease.

Limitations

The memory reduction feature has the following limitations for the VHDL designs:

- Only two-dimensional arrays declared explicitly through TYPE declarations are processed. Memory handling for a design aborts in the following cases:
  - If any identifier (signal/variable/constant) of three-dimensional or higher array type is encountered.
  - If two-dimensional array TYPE declaration of any identifier in a design unit is in a precompiled library because precompiled library is left untouched for memory handling.
- Two-dimensional records are not supported.
- Two-dimensional ports are not supported.
- Functions of precompiled libraries having 2-D argument/return type are not supported.

Pattern Matching Across Features

Atrenta Console uses pattern matching in waiver commands and in the search dialogs in the UI.

There are following modes for pattern matching:

- Using wildcard
- Using simple regular expressions

Wildcard Mode

The wildcard mode is the default mode. A string is considered as a wildcard if it includes an asterisk (*) or a question mark (?). However, if these characters are escaped by prefixing them with a backslash (\), they are considered as literals. For example, \?, *, and \\ are considered as ?, *, and \, respectively.

All other characters are treated as normal strings.

NOTE: _Atrenta Console treats square brackets as literals._

NOTE: _Interpretation of * and ? is similar for escaped design names. For example, if you want to specify a design object, '\abc* ', you must specify it as '\abc* '. On the other hand, if you want to specify all design objects whose names start with '\abc' then you need to write '\abc* ' (Verilog only) or '\abc*\' (VHDL only)._

A few parameters (examples taken from SpyGlass CDC solution) which support wildcard:

- synchronize_cells
- synchronize_data_cells
- clock_gate_cell
- glitch_protect_cell
- reset_synchronize_cell

The following constraints support wild cards:

Constraint	**Product**
memoryType	SpyGlass DFT solution
requirePath	SpyGlass DFT solution

Constraint	Product
testmode	SpyGlass DFT solution
clockgatingcell	SpyGlass DFT solution
pdsequence	SpyGlass Power Verify solution
powerdomainoutputs	SpyGlass Power Verify solution

Regular Expression Mode

A string that is prefixed with m/ and terminated with / (forward slash) is treated as a regular expression. The following delimiters can also be used in place of the delimiter /:

%	**!**	**@**	**^**	**&**	*****	**;**	**/**	**~**	**?**	**<**	**>**	**+**	**=**	**|**

Normal Mode

A string that is prefixed with q/ and terminated with / is treated literally, that is, if a string contains *, ?, or \, it will be treated literally. The delimiters explained in the *Regular Expression Mode* can also be used in place of /.

If a string does not contain an escaped *, ?, or \, it is treated as a wildcard string. To treat a string containing * or ? or a \ as a normal string, escape *, ?, or \ characters or prefix the string with q/ and terminate it with /.

Hierarchy Separator

All character except the following are treated as hierarchy separators in a hierarchy string:

- Any alpha numeric character
- $ (dollar)
- _ (underscore)
- [(opening square bracket)
-] (closing square bracket)

- : (colon)
- \ (backslash)

The hierarchy string contains only one hierarchy separator, that is, the first character that is not from the above set is considered the hierarchy separator, and the rest of the string is considered by using this character as a hierarchy separator.

NOTE: *For files and directory type objects, / is the only hierarchy separator.*

NOTE: *When you use pattern matching in a hierarchy, SpyGlass matches only a single hierarchy. This holds true for all objects, such as design objects (net, port, instance, terminals, and so on), files, and directories that use pattern matching.*

Design Save and Restore Feature

SpyGlass provides the Design Save-Restore feature that enables you to reuse the data from a prior synthesis run. This saves significant runtime for analysis that requires synthesis.

This feature saves runtime on repeated SpyGlass analysis runs when the HDL source is unchanged. It allows you to either restore or synthesize the entire list depending upon the change across save-restore runs.

Some types of analysis require you to run synthesis. In such cases, the Save-Restore capability is disabled.

By default, the Save-Restore is enabled in Atrenta Console. You can disable this feature in the *Preferences* dialog (*Tools-> Preferences*), under the *Miscellaneous* category. Alternatively, you can specify the following command in the project file to disable this feature:

```
set_option enable_save_restore no
```

This section also explains the *Incremental Save and Restore* and *Restoring From Multiple Databases* along with the *Examples*.

Incremental Save and Restore

SpyGlass provides support for incremental restore of saved design using the incremental save and restore feature. If any of the design files have changed across save-restore runs then instead of synthesizing the entire netlist again, SpyGlass only synthesizes the modules that are impacted by the change. This feature helps in saving the synthesis time for subsequent restore runs.

NOTE: *You can use this feature for a single database. To restore using multiple DBs, see Restoring From Multiple Databases.*

This feature works only when the *Design Save and Restore Feature* is enabled. This means that the *enable_save_restore* `option` is enabled to use the Incremental Save and Restore Feature.

To enable this feature, specify the following command in the Other Command Line Option(s) field in the Set Read Options tab:

allow_incr_save_restore

Alternatively, you can specify the following command in the project file to enable this feature,

```
set_option allow_incr_save_restore yes
```

Example 1

Consider the following sample message for a verilog module:

```
Synthesizing module: vlog_module (elaborated name: vlog_module)
... (Module 1 of total 100) done
```

The above message signifies that fresh synthesis is done for the module, `vlog_module`, due to change in the design file.

Example 2

Consider the following sample message for a VHDL module:

```
Synthesizing design unit: Entity.Arch (library: WORK,
elaborated name: Entity) ... (Module 1 of total 100)  done
(restored from dbdir)
```

The above message signifies that the module, `Entity.Arch`, is restored from the database.

Restoring From Multiple Databases

You can perform incremental restore/synthesis of the saved design from multiple databases using the `ipdbdir` switch. Specify the following command in the project file to specify a list of db's from which you want to restore.

```
set_option ipdbdir {list of db's}
```

Here, each individual db corresponds to an individual IP block. Depending on the impact of change across save-restore run, the netlist modules are either restored from their respective databases or synthesized afresh incrementally in the restore run.

NOTE: *Picking up synthesized view of modules coming from precompiled dump is not supported. That is, any module which is coming from precompiled dump will always get resynthesized.*

You can use a mix of design files and precompiled dumps. In this case, modules coming from design files are picked up from synthesized DB, while modules coming from precompiled dump are resynthesized.

Examples

Examples for the following cases are described below:

- *No Change Across Save-restore Runs*
- *Design File Change*
- *Module-Specific Option Change*
- *Global Option Change*

No Change Across Save-restore Runs

Run 1 (save1)

DB for hierarchy-1 is saved using following project file, project.prj:

project.prj

```
##Data Import Section
read_file -type verilog bottom1.v mid1.v top1.v


##Common Options Section
set_option enable_save_restore yes
set_option allow_incr_save_restore yes
set_option projectwdir .
set_option top top1

compile_design
```

The database is saved at the following location:

```
<projectwdir>/<project-name>/<top-name-if-
any>.SG_SaveRestoreDB
```

That is, for this example, the database is located at the following location:

```
save1/project/top1/.SG_SaveRestoreDB
```

Run 2 (save2)

Similarly, second database is created at the following location:

```
save2/project/top2/.SG_SaveRestoreDB
```

Run 3 (restore)

Both `top1` and `top2` are instantiated in a high-level SoC, top, defined in `top.v` as shown below:

```
read_file -type verilog bottom1.v mid1.v top1.v bottom2.v
mid2.v top2.v top.v
set_option ipdbdir {save1/project/top1/.SG_SaveRestoreDB
save2/project/top2/.SG_SaveRestoreD}
```

The `top` is synthesized as shown below:

```
Synthesizing module: bottom2 (elaborated name: bottom2) ...
(Module 1 of total 7) done (restored from dbdir save2/
project/top2/.SG_SaveRestoreDB)

Synthesizing module: mid2 (elaborated name: mid2) ... (Module
2 of total 7) done (restored from dbdir save2/project/top2/
.SG_SaveRestoreDB)

Synthesizing module: top2 (elaborated name: top2) ... (Module
3 of total 7) done
Synthesizing module: bottom1 (elaborated name: bottom1) ...
(Module 4 of total 7) done (restored from dbdir save1/
project/top1/.SG_SaveRestoreDB)

Synthesizing module: mid1 (elaborated name: mid1) ... (Module
5 of total 7) done (restored from dbdir save1/project/top1/
.SG_SaveRestoreDB)

Synthesizing module: top1 (elaborated name: top1) ... (Module
6 of total 7) done

Synthesizing module: top (elaborated name: top) ... (Module 7
of total 7) done
```

Design File Change

Run 1 (save1)

See *Run 1 (save1)*

Run 2 (save2)

See *Run 2 (save2)*

Run 3 (restore)

The `mid1.v` is changed across save-restore run and hence, netlist is synthesized as shown below:

```
Synthesizing module: bottom2 (elaborated name: bottom2) ...
(Module 1 of total 7) done (restored from dbdir save2/
project/top2/.SG_SaveRestoreDB)

Synthesizing module: mid2 (elaborated name: mid2) ... (Module
2 of total 7) done (restored from dbdir save2/project/top2/
.SG_SaveRestoreDB)

Synthesizing module: top2 (elaborated name: top2) ... (Module
3 of total 7) done

Synthesizing module: bottom1 (elaborated name: bottom1) ...
(Module 4 of total 7) done (restored from dbdir save1/
project/top1/.SG_SaveRestoreDB)

Synthesizing module: mid1 (elaborated name: mid1) ... (Module
5 of total 7) done

Synthesizing module: top1 (elaborated name: top1) ... (Module
6 of total 7) done

Synthesizing module: top (elaborated name: top) ... (Module 7
of total 7) done
```

Module-Specific Option Change

Run 1 (save1)

See *Run 1 (save1)*

Run 2 (save2)

See *Run 2 (save2)*

Run 3 (restore)

A parameter, `bottom2_p1`, defined in `bottom2` is changed in restore run as shown below:

```
set_option param bottom2.bottom2_p1=4
```

Now, the modules are synthesized in incremental fashion as shown below:

```
Synthesizing module: bottom2 (elaborated name: bottom2) ...
(Module 1 of total 7) done

Synthesizing module: mid2 (elaborated name: mid2) ... (Module
2 of total 7) done

Synthesizing module: top2 (elaborated name: top2) ... (Module
3 of total 7) done

Synthesizing module: bottom1 (elaborated name: bottom1) ...
(Module 4 of total 7) done (restored from dbdir save1/
project/top1/.SG_SaveRestoreDB)

Synthesizing module: mid1 (elaborated name: mid1) ... (Module
5 of total 7) done (restored from dbdir save1/project/top1/
.SG_SaveRestoreDB)

Synthesizing module: top1 (elaborated name: top1) ... (Module
6 of total 7) done
```

```
Synthesizing module: top (elaborated name: top) ... (Module 7
of total 7) done
```

Global Option Change

Run 1 (save1)

See *Run 1 (save1)*

Run 2 (save2)

See *Run 2 (save2)*

Run 3 (restore)

An option having global impact is changed in restore run as shown below:

```
set_option ignoredu mid1
```

In this case, the entire netlist is resynthesized as shown below:

```
Synthesizing module: bottom2 (elaborated name: bottom2) ...
(Module 1 of total 7) done

Synthesizing module: mid2 (elaborated name: mid2) ... (Module
2 of total 7) done

Synthesizing module: top2 (elaborated name: top2) ... (Module
3 of total 7) done

Synthesizing module: bottom1 (elaborated name: bottom1) ...
(Module 4 of total 7) done

Synthesizing module: mid1 (elaborated name: mid1) ... (Module
5 of total 7) done

Synthesizing module: top1 (elaborated name: top1) ... (Module
6 of total 7) done

Synthesizing module: top (elaborated name: top) ... (Module 7
of total 7) done
```

NOTE: *The I command is not recommended to be used in the project file. For details, see Options Not Recommended. However, you should specify the directory paths using the -I option on the command-line itself while invoking console or sg_shell.*

Save Restore Sensitive Options

Following table contains the list of save-restore sensitive options having global impact i.e., if any of these options change across DBs and the current run, then nothing gets picked up and it'll be a case of fresh synthesis

87	ovl_verilog	allow_incr_save_restore	ovl_vhdl
allow_celldefine_as_top	prefer_tech_lib	blackboxdir	resetall
compilelibs	sfcu	convert_udp_to_latch	sgsyn_clock_gating
def	sgsyn_clock_gating_threshold	stop/stopdir	sgsyn_enable_two_pass_flow
ignoredu	stopfile	sglib/gateslib	stopmodule
mthresh	synth_bypass	dw	synth
disable_gateslib_autocompile	synth_netlist	disable_handlememory	synth_vhdl_bypass
disableSV/enableSV	synth_vhdl_netlist	disablev2k	vlog2001_generate_name
enable_hbo	ignorefile	enable_hdl_encryption	libmap
enable_pgnetlist	macro_synthesis_off	enable_sbo	netlist
gateslibtype	no_celldefine_messages	handlememory	nopreserve
hdlin_synthesis_off_skip_text	no_rcheck_celldefine	hdlin_translate_off_skip_text	disableSV09/enableSV09
disable_amg/enable_amg	pragma	disable_hdlin_translate_off_skip_text	disable_hdlin_synthesis_off_skip_text

Handling Built-in Messages during Save-Restore Flow

During the save run, SpyGlass reports various built-in messages, such as parsing, elaboration, and synthesis. These messages are saved as a part of the saved design database.

To view these messages during restore run, the *Enable Save Restore for BuiltIn Rules* and *Enable Save Restore Flow* fields (under the *Set Read Options* tab) should be set to *Yes* during the restore run.

NOTE: *By default, the Enable Save Restore for BuiltIn Rules field is automatically set to Yes when you set the Enable Save Restore Flow field to Yes during the restore run.*

If you set the *Enable Save Restore for BuiltIn Rules* field to *No*, SpyGlass reports a message to indicate that built-in messages are not restored as the corresponding field is set to *No*.

Handling Builtin Messages Based on the Type of Current Run

Based on whether the current run is full restore or partial restore, built-in messages are handled in the following ways:

Full restore	In this case, no parsing, elaboration, or synthesis occurs. Therefore, all the messages reported by each of these stages are restored if you set the *Enable Save Restore for BuiltIn Rules* field to *Yes* under the *Set Read Options* tab.
Partial restore	In this case, parsing and elaboration occurs but no synthesis occurs. Therefore, only the messages reported during synthesis (during save run) are restored if you set the *Enable Save Restore for BuiltIn Rules* field to *Yes* under the *Set Read Options* tab. The parsing and elaboration messages are reported by the respective engines.

Impact of addrules/ignorerules Options

If you enable a built-in message during the restore run by using the *addrules* command, but that message was disabled during the save run, SpyGlass will not display that message during the restore run even if you have set the *Enable Save Restore for BuiltIn Rules* field to *Yes* under the *Set Read Options* tab.

In such cases, you should save the database forcefully by using the *addrules* command to make that message available in the restore run. If you do not save the database forcefully, SpyGlass reports a message in the restore run to indicate that the specified built-in message has actually not run in the restore run and the user needs to save the database again.

If you disable a built-in rule during the restore run by using the *ignorerules* command, but that rule was enabled during the save run, messages of such rules are not displayed during the restore run if you set the *enable_save_restore_builtin* command to `yes` command in a project file.

The following table describes the impact of *addrules*/*ignorerules* commands on built-in messages during save-restore runs:

Built-in Rule Type	**Built-in Message Reported During Save run?**	**Built-in Message Reported During Restore run?**
Off by default	No	no
Off by default	Yes (*addrules* is specified)	yes (*addrules* specified)
Off by default	No (*addrules* not specified)	no (even if *addrules* is specified)
On by default	Yes	yes
On by default	Yes	no (*ignorerules* specified)
On by default	No (*ignorerules* specified)	no (even if *ignorerules* is not specified)

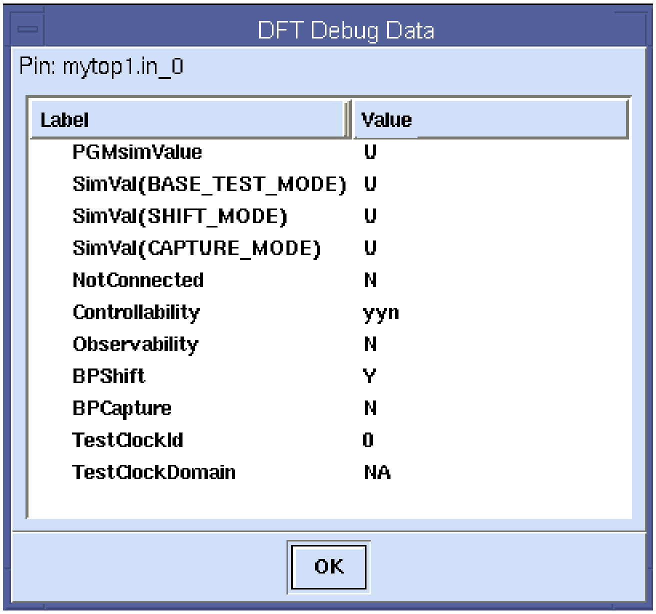
SpyGlass DFT Solution Back Annotation Feature

The Back Annotation feature of the SpyGlass DFT solution enables you to view debugging information in the schematics (Modular Schematic and Incremental Schematic) at the following two levels:

- Terminal level
- Instance level

To view SpyGlass DFT solution debugging information, you need to do the following:

1. Set the value of the `dftDebugData` rule parameter to `on`.
2. Start SpyGlass analysis.
3. Double-click the violation message for which the data can be viewed in the schematic.
4. Click the Schematic view or the Incremental Schematic view icon to open the schematic.
5. To view the SpyGlass DFT solution debug data for a terminal, right-click the terminal and select the *Show Debug Data-> DFT* context-menu option. This displays the *DFT Debug Data* window, as shown in the following figure:



DFT Debug Data

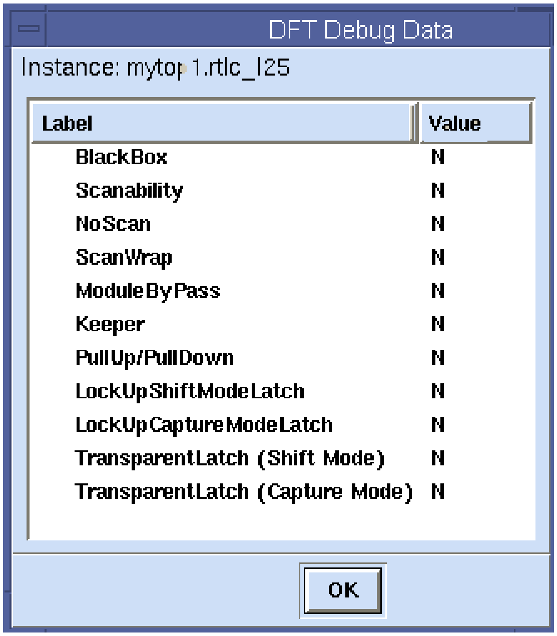
Pin: mytop1.in_0

Label	Value
PGMsimValue	U
SimVal(BASE_TEST_MODE)	U
SimVal(SHIFT_MODE)	U
SimVal(CAPTURE_MODE)	U
NotConnected	N
Controllability	yyn
Observability	N
BPShift	Y
BPCapture	N
TestClockId	0
TestClockDomain	NA

FIGURE 1. DFT Debug Data for a Terminal

SpyGlass DFT solution debug data for a terminal includes information, such as the simulation value in the shift and capture modes, controllability, blocked path due to the terminal in shift mode (BPShift), and blocked path due to the terminal in the capture mode (BPCapture).

6. To view SpyGlass DFT solution debug data for an instance, right-click the instance and select the *Show Debug Data-> DFT* menu option from the context menu. This displays the *DFT Debug Data* window, as shown in the following figure:



DFT Debug Data

Instance: mytop1.rtlc_I25

Label	Value
BlackBox	N
Scanability	N
NoScan	N
ScanWrap	N
ModuleByPass	N
Keeper	N
PullUp/PullDown	N
LockUpShiftModeLatch	N
LockUpCaptureModeLatch	N
TransparentLatch (Shift Mode)	N
TransparentLatch (Capture Mode)	N

FIGURE 2. DFT Debug Data for an Instance

SpyGlass DFT solution debug data for an instance includes information, such as:

- Scannability information of a flip-flop
- Module by pass information that is applied through SGDC commands on a black box
- Re-timing information on a latch in the shift mode or capture mode
- Transparency information for a latch in the shift mode or capture mode
- Scan-wrap information applied on a black box instance

Support for Verilog Configuration

Verilog Configuration provides the ability to specify design configurations. It is consists of a set of rules to apply when searching for library cells to which instances are to be bind. A configuration may change the binding of a module, primitive, interface or program instance. However, the configuration can not change the binding of a package. Configurations mainly specify the set of libraries from which master of an instance is to be bind. A library can contain modules, interfaces, packages and other configurations.

This section explains the following topics:

- *Related Commands*
- *Related Reports*

Related Commands

The support for Verilog configuration is enabled through the following commands:

- *enable_vlog_config*: Enables the support for verilog configuration
- *strict_vlog_config*: Converts the instances, which are unable to bind, using the verilog configuration rules, into black box and assigns the ELAB error to them
- *write_vlog_config_report*: Enables reporting of instances, which are bound using the configuration rules.

Related Reports

The *write_vlog_config_report* command generates a binding report, ReportConfiguration.log, which lists all the instances that are bound using the configuration rules. The following is the a sample snippet of the ReportConfiguration.log file:

```
###################################################

## Instance                          : top.I2.I3.I5
## Configuration Binding             : L5.superbottom
## Configuration Binding Rule        : instance top.I2.I3.I5
```

```
                                       use L5.superbottom ;
## Configuration Line Number        : 12
## Configuration File Name          : config.v

###################################################
Details :
Instance in report suggests hierarchical name for the
instance from the top.

Configuration binding suggests masterName and library Name to
which the instance is bound.

Configuration binding Rule suggests the configuration rule
using which the instance is bound.
Configuration Line number suggests the line number of binding
rule
Configuration File Name suggests the binding rule
configuration filename.

There may be cases where binding for all the instances down
the hierarchy of a particular instance are shared. In that
case, the report dumps only the binding information of
hierarchy's top instance. The dump will look like:
###################################################
## Note      : Complete Hierarchy of top.I2.I4 instance will
be shared with top.I1.I4 instance's Hierarchy
###################################################
```

Record-Mapping to Flattened Bus

When you specify VHDL records, Atrenta Console internally flattens the corresponding record elements into bus.

Consider the following VHDL record:

```
type rec1 is
record
  field1 : std_logic_vector(2 downto 0);
  field2 : std_logic;
  field3 : std_logic;
  field4 : std_logic_vector(0 to 2);
end record;
signal sig1 : rec1; --Flattened bus: sig1[7:0]
```

For each record element, the following table displays the flattened bus and its mapping details:

Record Element	Flattened Bus	Mapping
field1	sig1[2:0]	sig1[2] <=> sig1.field1(2), sig1[0] <=> sig1.field1(0)
field2	sig1[3]	sig1[3] <=> sig1.field2
field3	sig1[4]	sig1[4] <=> sig1.field3
field4	sig1[7:5]	sig1[7] <=> sig1.field4(0), sig1[5] <=> sig1.field4(2)

NOTE: *Arrays with range defined as (n downto 0) or (0 to n) in RTL maps to (0 to n) or (n downto 0) order at the flattened level.*

Handling Complex VHDL Records

SpyGlass handles complex VHDL records, such as a record within another record or an array of records during post-synthesis.

After generating a netlist of the design containing complex VHDL record(s), you can apply constraints on records.

Specifying Constrains on Record Bits

Consider the following VHDL code snippet containing a complex record:

```
type simple_record is
  record
    data4 : std_logic;
    data5 : std_logic_vector(7 downto 0);
  end record;

type record_data_type is
  record
    data1 : std_logic;
    data2 : std_logic_vector(7 downto 0);
    data3 : simple_record;
  end record;
```

Also consider that you specify the following internal signal:

```
signal sig1:record_data_type;
```

The naming convention for record bits is <record-instance>_<field-name>.

If the record field is an array, refer the field name through its indices.

The following example applies constraints on record sig1 bits:

```
// applying set_case_analysis constraint on record bit
// sig1.data3.data4
set_case_analysis -name sig1_data3_data4 -value 1

// applying set_case_analysis constraint on record bit
// sig1.data3.data5(0)
set_case_analysis -name sig1_data3_data5[0] -value 1
```

Specifying Constrains on Array of Records

Consider the following VHDL code snippet containing an array of records:

```
type simple_record is
record
  data1 : std_logic;
  data2 : std_logic_vector(1 downto 0);
end record;


type array_data_type is array (1 downto 0) of
simple_record ;
signal sig1:array_data_type
```

The naming convention for an array records is <record-instance>[<index>].

SpyGlass refers an array of records as a one-dimensional net-bundle, as given below:

```
sig1(0).data1        sig1[0]
sig1(0).data2[0]     sig1[1]
sig1(0).data2[1]     sig1[2]
sig1(1).data1        sig1[3]
sig1(1).data2[0]     sig1[4]
sig1(1).data2[1]     sig1[5]
```

All record elements in the net-bundle are accessed using indices while applying constraints.

The following example applies constraint on an array of record sig1 bits:

```
// Applying the set_case_analysis constraint on the
// record array sig1(0).data1
set_case_analysis -name sig1[0] -value 1
```

Specifying Constraints through GUI

To apply constraints on record bits through GUI, perform the following steps:

1. Select the required net by following the net-naming convention in the schematic.
1. Right-click on the required net.
2. Select the *Set SGDC Constraints* option from the shortcut menu.

This displays the *Constraints Editor* window.

3. In the *Constraints Editor* window, specify the required constraints.

This is the recommended way to specify constraints for records bits as you can view the naming convention of nets directly in the schematic, and apply constraints accordingly.

To view the naming convention of a net in the schematic, move the cursor over that net. A tool-tip appears displaying the net name, as shown in the following figure:

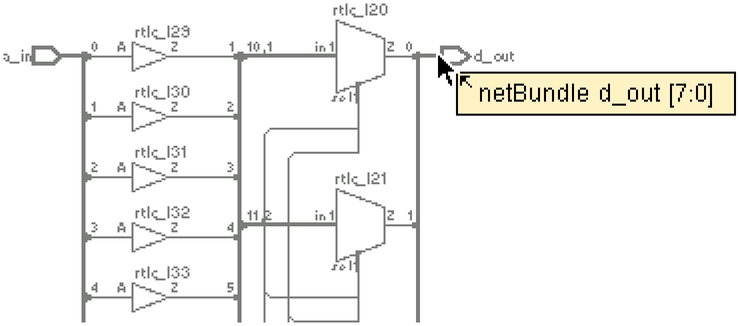


FIGURE 3. Tool-Tip for a Net Name

Incremental Schematic Abstraction Feature

A schematic highlights connectivity in a design while focusing on the design part that contains a violation. However, in large designs, highlighted paths are quite lengthy. As a result, it becomes difficult to visualize the connectivity and this takes away the focus from main issue being reported.

To solve this problem, the incremental schematic abstraction feature abstracts different types of logic into abstract groups to make a complex schematic appear less cluttered. This enables a user to focus only on relevant objects/paths in the schematic.

Combinational Logic Abstraction

Incremental schematic abstraction feature clubs all combinational logic present within registers in a combinational cloud. This enables a user to focus only on registers in the schematic and not on the combinational logic present inside registers.

The following figure shows the schematic in which combinational logic within each register is clubbed in a combinational cloud:

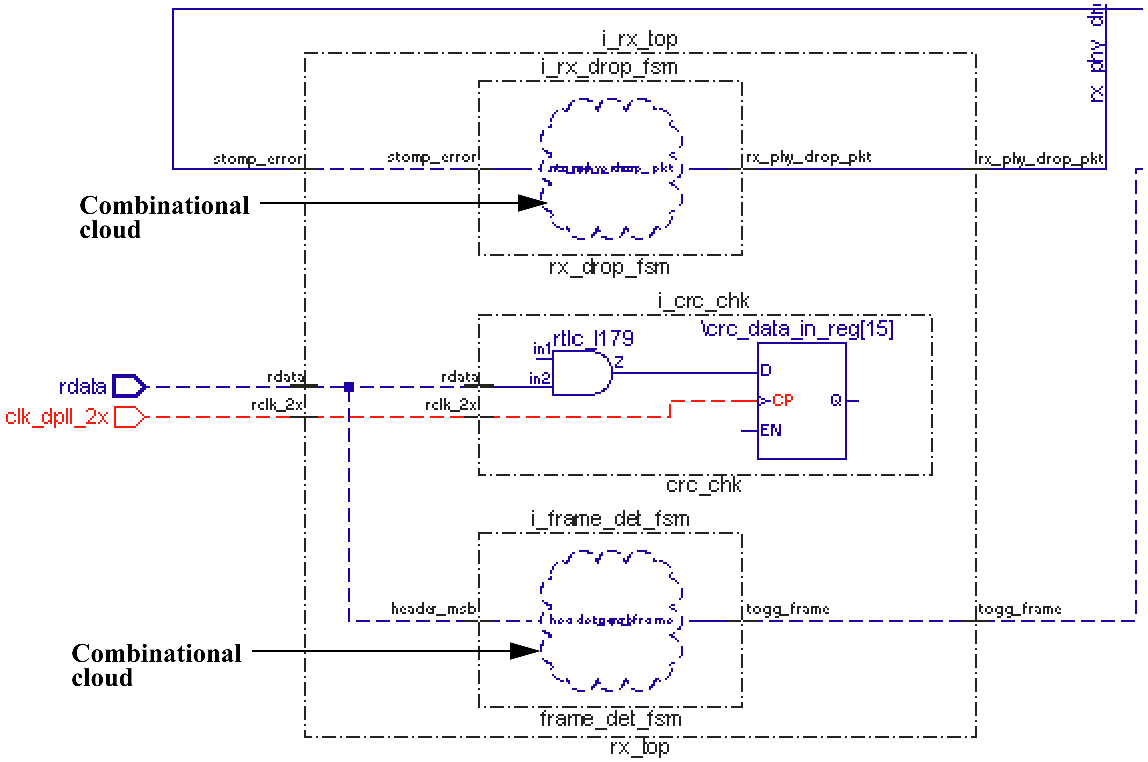


FIGURE 4. Combinational Logic Clubbed in a Combinational Cloud

If you want to view combinational logic within registers, click the button in the *Incremental Schematic* window. The schematic now changes to the following:

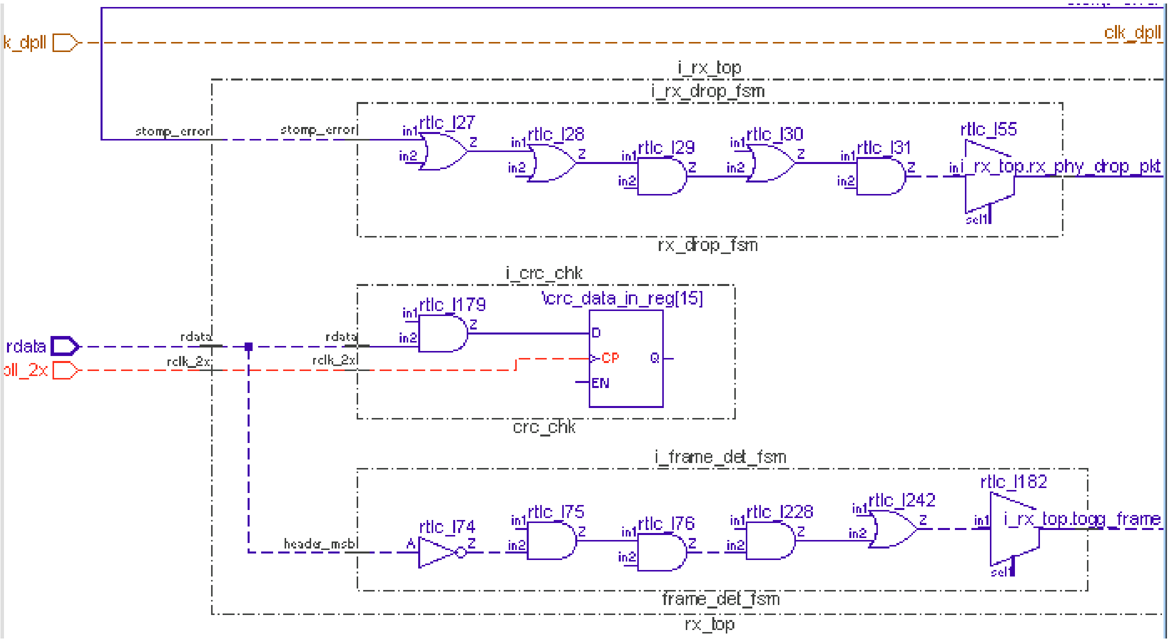


FIGURE 5. Combinational Logic in a Register

You can again club all combinational logic within registers back into a combinational cloud by clicking the button in the *Incremental Schematic* window.

Buffer Chains Abstraction

Incremental schematic abstraction feature replaces a chain of buffers or inverters with a single buffer or inverter in a schematic.

A chain of buffers or inverters generally does not have any logical significance from the point of view of debugging.

The following figure shows the schematic in which a chain of buffers has been replaced with a single buffer:

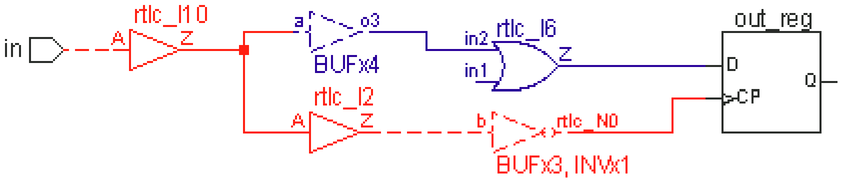


FIGURE 6. Chain of Buffers Replaced with a Single Buffer

NOTE: *Only buffers that are in a straight chain can be abstracted as a single group. In case of any branching, this feature creates abstract groups for each sub chain.*

If you want to view the complete chain of buffers, click the ▷ button in the *Incremental Schematic* window. The schematic now changes to the following

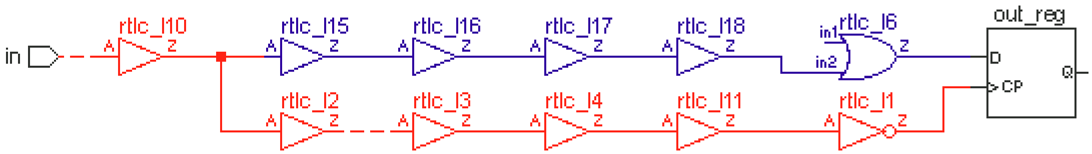


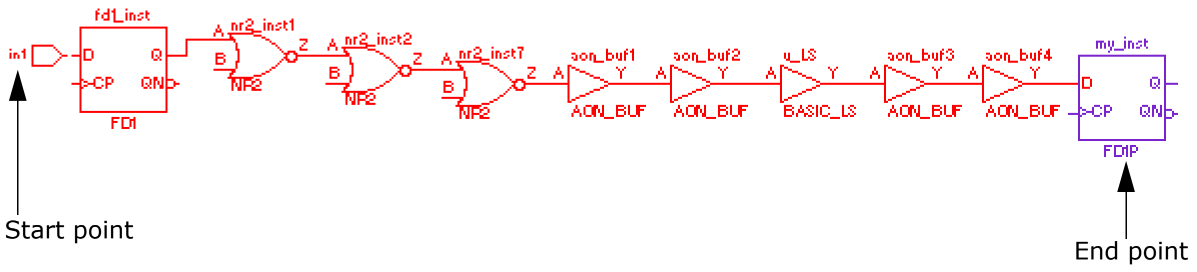
FIGURE 7. Complete Chain of Buffers

To replace the above chain back with a single buffer, click the ▷ button in the *Incremental Schematic* window.

Abstraction between Start and End Points

Incremental schematic abstraction feature abstracts logic between start and end points in the schematic.

For example, consider the following figure:



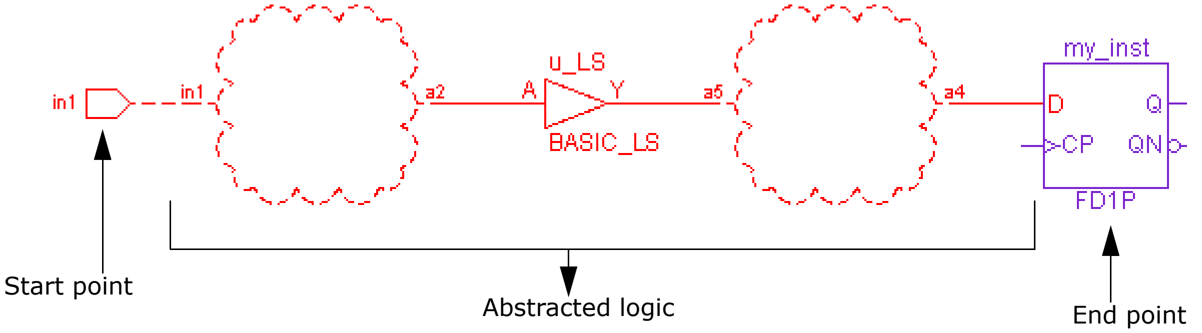
Start point

End point

FIGURE 8. Start and End Point

In the above schematic, click the button to abstract logic between the start and end points.

The following figure shows the modified schematic:



Start point

Abstracted logic

End point

FIGURE 9. Abstracted Logic Between Start and End Points

Based on the functionality of a rule, certain logic between start and end points are not abstracted. For example, in the above figure, the level shifter is not abstracted.

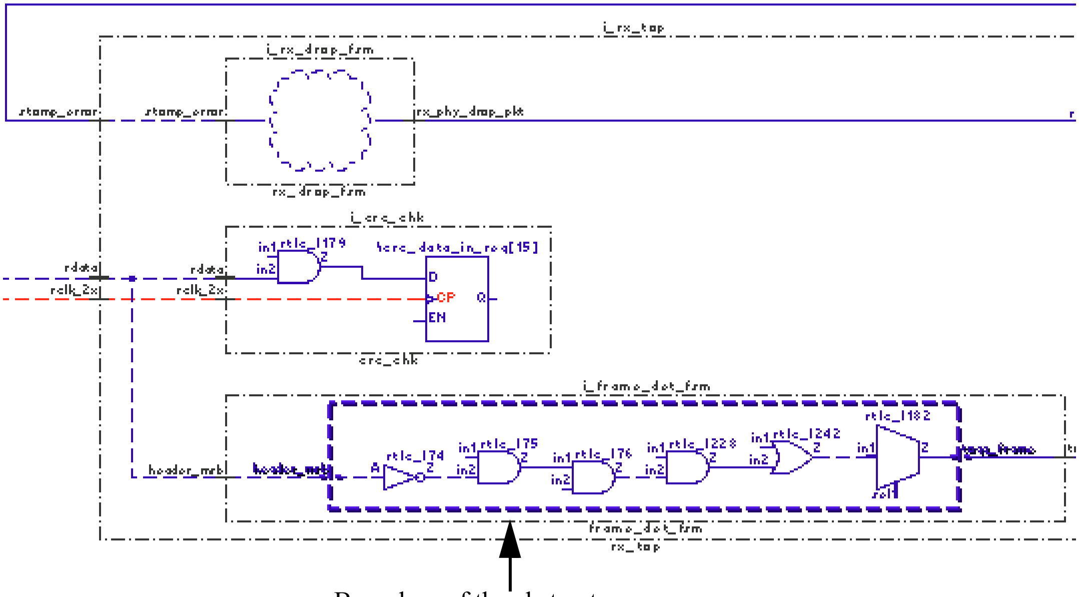
Viewing Logic within an Abstract Group

To view logic within a particular abstract group, perform any of the following actions:

- Double-click on the required group.

- Right-click on a group and select the *Expand Group* option from the shortcut menu.

Atrenta Console then displays all the logic within the boundary of that abstract group, as shown in the following figure:



Boundary of the abstract group

FIGURE 10. Schematic Showing Boundary of Abstract Group

To collapse the expanded group, right-click on that group and select the *Collapse Group* option from the shortcut menu.

Viewing or Hiding Pins of an Abstracted Group

To view all pins of an abstracted group, right-click on that group and select the *Show All Pins* option from the shortcut menu.

To hide pins that have not been probed or traced, right-click on the corresponding abstract group and select the *Hide Untraced Pins* option from the shortcut menu.

Removing All Abstract Groups

To remove all abstract groups in the schematic, click the *Disable Groups* button ( ) in the toolbar.

The Batch Mode in Atrenta Console

Overview

To use Atrenta Console in the batch mode, specify the `-batch` command-line option along with the other commands, as shown below:

`spyglass -batch [-project <project-file>] [` *Atrenta Console-Specific Options* `] [` *Console Batch, Goal, and Source File List Command-Line Options* `]`

NOTE: *If you do not specify a project file in the batch mode, Atrenta Console runs in a normal batch mode in which Atrenta Console-Specific Options do not work.*

This chapter provides details on various command-line options used in Atrenta Console.

Atrenta Console-Specific Options

Following are the Atrenta Console-specific command-line options:

-goals <goals>

Use this command to run the specified goals.

Please note the following points:

- Atrenta Console searches goals in the default methodology directory as set in the project file.
- You can specify goal names by using wildcard expressions. For details on wildcard search, see *Pattern Matching Across Features*.
- If you do not specify a valid goal, SpyGlass uses the best possible matching alternatives to the given goal name.

Consolidated Goal Results Summary

When you run multiple goals by using the `-goals` command-line option, then at the end of the run, Atrenta Console displays a consolidated results summary that contains the run result, message count per severity class, and the total messages reported, waived and generated for each goal run. In addition, the consolidated goals results summary displays the paths pointing to the spyglass.log file and various reports.

The sample consolidated goal results summary generated by Atrenta Console is as follows:

```
------------------------------------------------------------
Consolidated Results Summary for multi-goal run
------------------------------------------------------------
Goal : Initial_RTL/Connectivity     Status : PASS
=====================================================
    3 ERROR Severity Messages
    24 WARNING Severity Messages
    15 INFO Severity Messages

    Total Messages Generated        :        42
    Total Messages Waived           :        0
    Total Messages Reported         :        42
```

```
Run LogPath : /delsoft/test/VI-27088/Project-410/
Initial_RTL/Connectivity/spyglass.log

Run Results : /delsoft/test/VI-27088/Project-410/
Initial_RTL/Connectivity/spyglass_reports/

Goal : Initial_RTL/Simulation     Status : PASS
=====================================================
  4 ERROR Severity Messages
  157 WARNING Severity Messages
  17 INFO Severity Messages


  Total Messages Generated        :       178
  Total Messages Waived           :       0
  Total Messages Reported         :       178

 Run LogPath : /delsoft/test/VI-27088/Project-410/
 Initial_RTL/Simulation/spyglass.log


 Run Results : /delsoft/test/VI-27088/Project-410/
 Initial_RTL/Simulation/spyglass_reports/
```

-designread

Use this command to perform design read in the batch run of Atrenta Console.

NOTE: *When the -designread option is given along with the -goals option, design read is performed first followed by goal run.*

Example

```
%> spyglass -project myproject.prj -batch -designread or
%> spyglass -project myproject.prj -batch -designread
-goals="Initial_RTL/Connectivity"
```

-showgoals

Use this command to print all the goals in the default methodology directory along with their recommended setup status and run status.

For regression goals, the `-showgoals` command prints the name of the regression goals in the Goal Status Summary report. To view the details of the regression goals, use the `-verbose` command.

If the goal run is complete, this command prints the corresponding message count by severity (Error/Info/Warning counts of messages per goal is displayed in the goal summary).

The following is the sample goal summary:

```
=======================================================================
Goal Status Summary
======================================================================
Goal                            Setup Status   Run Status    Results
                                                             by Severity
======================================================================
initial_rtl/lint/connectivity   Setup Optional Run Complete  (0/0/2/3)
initial_rtl/lint/simulation     Setup Optional Not Run Yet
initial_rtl/lint/synthesis      Setup Optional Not Run Yet
initial_rtl/lint/structure      Setup Optional Not Run Yet
initial_rtl/audit/block_profile Setup Optional Not Run Yet
initial_rtl/audit/rtl_audit     Setup Optional Not Run Yet
myReg3                              -                -        Regression
                                                              Goal
myReg1                               -               -        Regression
                                                              Goal
=======================================================================
Note: The format of the "Results By Severity" column is: ( Fatal / Errors
      / Warnings / Info )
      It contains "Regression Goal" keyword for regression goals, use
      -verbose option to get details on these goals.
* -  Goal(s) has one or more prerequisite goals which must be run
     prior to running this goal. The batch process has been configured to
     enforce this and will indicate which, if any, goals need to be run
    first. For details on the prerequisites requirements, please see the
     documentation for this methodology.
```

```
=======================================================================
```

NOTE: *If you have selected the Show Optional Goals option in the Preferences dialog, the* `-showgoals` *option also prints the optional goals along with their recommended setup status and run status.*

-verbose

Regression goals summary is not shown by default with -showgoals. Specify this option to get Setup, Run status and Result summary for each goal specified as part of regression goals. This option is valid with -showgoals option only.

Following is a sample output from -showgoals in batchConsole run without -verbose option:

```
=======================================================================
Goal Status Summary
=======================================================================
Goal        Setup Status       Run Status     Results by Severity
=======================================================================
testgoal0   Setup Optional    Not Run Yet
testgoal1   Setup Optional    Not Run Yet
testgoal2   Setup Optional    Not Run Yet
testgoal3   Setup Optional    Not Run Yet
myReg3      -                 -               Regression Goal
myReg1      -                 -               Regression Goal
=======================================================================
Note: The format of the "Results By Severity" column is: ( Fatal / Errors
     / Warnings / Info )
     It contains "Regression Goal" keyword for regression goals, use
     -verbose option to get details on these goals.
* - Goal(s) has one or more prerequisite goals which must be run
    prior to running this goal. The batch process has been configured
    to enforce this and will indicate which, if any, goals need to be run
    first. For details on the prerequisites requirements, please see the
    documentation for this methodology.

=======================================================================
```

If you specify -verbose option with the -showgoals option, then status is shown for each goal inside regression goals. The following tables are additionally decompiled prior to **Note:** in the above report, if -verbose option is also specified.

```
Regression Goal 'myReg3' Status Summary
=======================================================================
Goal        Setup Status        Run Status     Results by Severity
=======================================================================
testgoal1   Setup Optional      Not Run Yet
testgoal2   Setup Optional      Not Run Yet
=======================================================================

Regression Goal 'myReg1' Status Summary
=======================================================================
Goal        Setup Status        Run Status     Results by Severity
=======================================================================
testgoal2   Setup Optional      Not Run Yet
testgoal3   Setup Optional      Not Run Yet
=======================================================================
```

-group

Use this command to run the goals specified using the *-goals <goals>* option.

When you use this option, combined results for all goals is generated and stored in a common project working directory Group_Run. Use this -group command-line when you want to analyze the results of different goals together.

There are some limitations when running Atrenta Console in batch mode with the -group option. These are:

- The –showgoals option does not report that the goals have been completed
- In goal selection, the goals are shown as been run, however, the summary data is not available

- Since a group has a single result directory, when analyzing the results of a goal from a group the run with show all results for that group
- The results of the group run cannot be used for the aggregated reports for a project, or consolidated management reports

-gen_aggregate_report

Use this command to generate aggregated report, such as project summary, DataSheet, or DashBoard report, in the batch mode. You can generate only one type of aggregated report at a time by using this command.

Following is the syntax of using the `-gen_aggregate_report` command:

```
-gen_aggregate_report <project_summary | datasheet |
dashboard>
-config_file <config-file> [-reportdir <dir>] [-DEBUG]
[ -LICENSEDEBUG ] [-project <prj-file>]
```

The above command requires you to specify two mandatory arguments, report type (`project_summary`, `datasheet`, or `dashboard`) and the configuration file (`-config_file` *`<config-file>`*).

Various options of this command are described in the following table:

TABLE 1

Option name	Description
-config_file	Specifies the name of a configuration file that contains a list of projects and run directories generated by goal/ source files or GUI. For the *Project Summary Report*, you can only specify a list of project files, because this report is specific to projects only. For the rest of the aggregated reports, such as the *DataSheet Report* and the *DashBoard Report*, you can specify a list of projects as well as run directories.
-project	(Optional) Specifies the name of a single project file. This option is considered only if you do not specify the `-config_file` option. In such cases, Atrenta Console accepts the specified project file as the input. If you have specified both the `-config_file` and `-project` options, Atrenta Console ignores the `-project` option and considers the`-config_file` option.
-reportdir	(Optional) Specifies the directory containing result files. By default, Atrenta Console considers the value of this option as the current working directory.
-DEBUG	(Optional) Prints useful debug messages on STDOUT. This information includes various details such as the current project accessing, time stamps at various stages, etc. Information printed on STDOUT is also be dumped in the log file, aggregate_reports.log.
-LICENSEDEBUG	Prints license debug information. For details on this option, refer to *Atrenta Console Reference Guide*.

-parallel_run

Use this command to run goals in parallel.

-host_config_file

Use this command to specify a host configuration file while running goals in parallel.

The host configuration file contains details, such as login type, maximum number of process that can run at a time and machine names on which goals should run.

Following is the sample host configuration file:

```
LOGIN_TYPE: lsf
MAX_PROCESSES: <num>
LSF_CMD: <bsub-command>
```

-enable_cmdline_debug

Use this command to generate the spyglass_cmdline_debug.log file that enables you to understand how SpyGlass arrives at the final set of command line options using the initial option set provided by you.

-oem_mode

Use this command to run Atrenta Console GUI in the OEM (Original Equipment Manufacturer) mode.

In the OEM mode, you will notice the following differences as compared to the normal Atrenta Console run:

- The *Run Design Read* tab is disabled.
- You cannot run non-OEM goals in this mode. If you try loading such goals, Atrenta Console reports an error and does not load these goals.
- The save/restore feature is disabled.
- Parallel goal run is not supported.
- The *Common Setup* tab under the *Goal Setup and Run* tab is disabled.

-I

See *I*.

-shell

Use this command to invoke `sg_shell` from SpyGlass as shown below:

```
% spyglass -shell
```

You can perform the following tasks while invoking `sg_shell` from SpyGlass:

- Load a project file using `sg_shell` as shown below:

  ```
  spyglass -shell -project <project_file_name>.prj
  ```

- Run one or more goals from an existing project file using sg_shell as shown below:

  ```
  % spyglass -shell -project<project_file_name.prj> -goals
    "<goal1>,<goal2>,..."
  ```

- Playback a set of Tcl commands in the sg_shell as shown below:

  ```
  % spyglass -shell -tcl <file_name>.tcl
  ```

Console Batch, Goal, and Source File List Command-Line Options

This section explains the various console batch, goal, and source file list command-line options.

These Atrenta Console-specific commands can be:

- Design-related commands which are stored in a project file.
- Rule/product specific commands that are stored in a methodology file.

These files are then used by Atrenta Console.

NOTE: *The following classic-batch options are converted into the Atrenta Console format, but are not mapped directly to the project or methodology files:*

-f	*-template <goal-names>*	*-templatedir*	*-wdir*	*-logfile*	-namevdb

Specifying Source Files through Command-Line

Provide a space-separated list of HDL files on command-line.

You can provide file names with or without a path (actual or relative).

If you provide only file name, the file is searched in the current directory.

You can also use wildcard expressions to specify file names. For example, specifying *.vhd will result in processing all files with extension .vhd in the current directory.

Specifying Rule Parameters through Command-Line

Specify rule parameters in the goal or source files in the following format:

```
-<name>=<value>
```

Where:

- *<name>* is the name of the rule parameter.
- *<value>* is the parameter value.

NOTE: *To know parameters and their corresponding usage for each product, refer to product documentation.*

If you specify a command-line option that SpyGlass does not recognize, it creates a parameter of that name. If the option has a value, it assigns that value to the parameter.

NOTE: *Specifying parameters in goal or source files is equivalent to using the following project file command:*

```
set_parameter <param-name> <param-value>
```

Classic Batch Command-Line Options and their Corresponding Project File Commands

The following table lists the classic-batch command-line options and their equivalent project file commands:

Classic-Batch Command-Line Options	Corresponding Project File Commands
-87	*87*
-addrules <rule-list>	*addrules*
-allow_celldefine_as_top	*allow_celldefine_as_top*
-allow_module_override	*allow_module_override*
-cell_library	*cell_library*
-check_celldefine	*check_celldefine*
-checkdu="<du-name>"	*checkdu*
-checkip="<IP-name>"	*checkip*
-checkTopDu <rule-names>	*checkTopDu*
-configfile <file-name>	*configfile*
-consolidate_reportdir	*consolidate_reportdir*
-convert_udp_to_latch	*convert_udp_to_latch*
-def <file-name>	*def* option of the *read_file* command
+define{+<macro-name>}	*define*
-define_incr_dirmap	*define_incr_dirmap*
-define_severity <severity-expression>	*define_severity*
-disable_encrypted_hdl_checks	*disable_encrypted_hdl_checks*

Classic-Batch Command-Line Options	**Corresponding Project File Commands**
-disable_hdllibdu_lexical_checks	*disable_hdllibdu_lexical_checks*
-disable_report	*disable_report*
-disallow_view_delete	*disallow_view_delete*
-dump_all_modes	*dump_all_modes*
-dw	*dw*
-enable_const_prop_thru_seq	*enable_const_prop_thru_seq*
-enable_hdl_encryption	*enable_hdl_encryption*
-enable_module_based_reporting	*enable_module_based_reporting*
-enable_pass_exit_codes	*enable_pass_exit_codes*
-enable_precompile_vlog	*enable_precompile_vlog*
-enable_save_restore	*enable_save_restore*
-enable_sgdc_debug	*enable_sglib_debug*
-enable_sglib_debug	*enable_sglib_debug*
-enableSV	*enableSV*
-f	*sourcelist* option of the *read_file* command
-gateslib <file-name>	*gateslib* option of the *read_file* command
-gen_hiersgdc	*gen_hiersgdc*
-handlememory	*handlememory*
-hdlin_synthesis_off_skip_text	*hdlin_synthesis_off_skip_text*
-hdlin_translate_off_skip_text	*hdlin_translate_off_skip_text*
-hdllibdu	*hdllibdu*
-higher_capacity	*higher_capacity*
-ignore_builtin_rules	*ignore_builtin_rule*
-ignore_builtin_spqdir <dir>	*ignore_builtin_spqdir*
-ignorerules {<rule-name>}	*ignorerules*
-ignorelibs	*ignorelibs*
-ignore_undefined_rules	*ignore_undefined_rules*
-ignorewaivers	*ignorewaivers*
-inferblackbox	*inferblackbox*

Classic-Batch Command-Line Options	**Corresponding Project File Commands**
-inferblackbox_iterations <int-value>	*inferblackbox_iterations*
-lef <file-name>	*lef* option of the *read_file* command
-lib <logical-lib-name> <physical-lib-name>	*lib*
[+libext{+<ext-name> }	*libext*
-LICENSEDEBUG	*LICENSEDEBUG*
-lvpr (<number>	*lvpr*
-macro_synthesis_off	*macro_synthesis_off*
-mapSuffixList <suffix-names>	*mapSuffixList*
-mapPrefixList <prefix-names>	*mapPrefixList*
-mapVirtualClkByName = <yes | no>	*mapVirtualClkByName*
-mixed	*mixed* option of the *language_mode* command
-mthresh	*mthresh*
-net_osc_count_limit <limit>	*net_osc_count_limit*
-nobb	*nobb*
-nodefparam	*nodefparam*
-noelab	*noelab*
-nopreserve	*nopreserve*
-noreport	*noreport*
-norules	*norules*
nosavepolicies <product-list> -nosavepolicy <product-name>	*nosavepolicy/nosavepolicies*
-overload {<named-overload>}	*overload*
-overloadpolicy {<product-name>	*overloadpolicy*
-overloadrules <expression>	*overloadrules*
-ovl_verilog <file-name>	*ovl_verilog*
-ovl_vhdl <file-name>	*ovl_vhdl*
--perflog	*perflog*
-plib <file-name>	*plib* option of the *read_file* command
-pragma <name>	*pragma*
-prefer_tech_lib	*prefer_tech_lib*

Classic-Batch Command-Line Options	Corresponding Project File Commands
-preserve_mux	*preserve_mux*
-print_sortorder_only	*print_sortorder_only*
-relax_hdl_parsing	*relax_hdl_parsing*
-remove_work	*remove_work*
-report { <formats> }	*report*
-report_adjustment_waiver	*report_adjustment_waiver*
-reportfile <file-name>	*reportfile*
-report_inst_backref	*report_inst_backref*
-report_ip_waiver	*report_ip_waiver*
-report_max_inst <value>	*resetall*
-report_max_size=<value>	*report_max_size*
-report_per_policy	*report_per_policy*
--report_style=<list-values>	*report_style*
+resetall	*resetall*
-rules {<rule-name>}	*rules*
-savepolicies <product-list> -savepolicy <product-name>	*savepolicy/savepolicies*
-sfcu	*sfcu*
-sdc2sgdc	*sdc2sgdc*
-sdc2sgdcfile <file-name>	*sdc2sgdcfile*
-sdc2sgdc_mode <mode-name>	*sdc2sgdc_mode*
-sgdc <file-name>	*sgdc* option of the *read_file* command
-sglib <file-name>	*sglib* option of the *read_file* command
-sgsyn_clock_gating	*sgsyn_clock_gating*
-sgsyn_clock_gating_threshold <num>	*sgsyn_clock_gating_threshold*
--sgsyn_enable_latch_removal	*sgsyn_enable_latch_removal*
--sgsyn_loop_limit <value>	*sgsyn_loop_limit*
-show_sdc_progress	*show_sdc_progress*
-show_lib	*show_lib*
-skip_rules_for_fast_restore	*skip_rules_for_fast_restore*

Classic-Batch Command-Line Options	Corresponding Project File Commands
-sort	*sort*
-sortrule	*sortrule*
-stop <module-name>	*stop*
-stopdir <dir-name>	*stopdir*
-stopfile <file-name>	*stopfile*
-support_sdc_style_escaped_name	*support_sdc_style_escaped_name*
-target <lib-name-list>	*target*
-templatedir	*active_methodology* For details on using this command, refer to Console User Guide.
-testsynth	*designread_enable_synthesis*
-top <module-name>	*top*
-use_goal_rule_sort	*use_goal_rule_sort*
-use_scan_flops	*use_scan_flops*
-v <lib-name>	*v*
-validate_hiersgdc	*validate_hiersgdc*
-verilog	`verilog` option of the *language_mode* command
-vhdl	`vhdl` option of the *language_mode* command
-w	*w*
-waiver <file-name>	*waiver* option of the *read_file* command
-wdir	*projectwdir*
-work <dir-name>	*work*
-write_sdc	*write_sdc*
-y <lib-dir-name>	*y*

Other Command-Line Options

This section describes other classic batch command-line options, such as informational options and mode-selection options.

NOTE: *The class-batch feature has entered the End-of-Life stage and will not be supported* *from now on. This section is listed only for the purpose of backward-compatibility.*

Informational Command-Line Options

The SpyGlass informational options provide information about SpyGlass software environment and do not require you to specify a design file.

NOTE: *SpyGlass does not create a violation database file and log file when you invoke SpyGlass with any of the informational options.*

NOTE: *You do not need to specify the* `-batch` *command-line option with the informational command-line options.*

-h

(Optional) Prints site-specific SpyGlass help (that is, information in <your-inst-dir>/SPYGLASS_HOME/doc/site-help.txt) to the stdout and exits.

You can access site-specific help from SpyGlass by entering the `-h` or *-help* option as in the following example:

```
spyglass -h
```

```
spyglass -help
```

The information you see depends on what customer-specific SpyGlass help information has been set up for your company. If no such help has been set up, a message appears saying site-specific help does not exist.

-help

Same as the *-h* option.

-quickstart

(Optional) Prints the SpyGlass Quick Start help (that is, information in <your-inst-dir>/SPYGLASS_HOME/doc/quickstart.txt) to the stdout and exits.

-usage

(Optional) Prints the help of various options based on the mode (Atrenta Console GUI or batch) in which you want to run SpyGlass.

- If `-usage` is specified alone, Atrenta Console help is shown and a reference of getting batch help is shown.

- If `-usage` is specified with any Atrenta Console-related option, Atrenta Console help is shown.
- If `-usage` is specified with any Atrenta Console related options as well as a batch-related option, a message is displayed indicating that the options specified are not compatible, and only Atrenta Console-related help is shown.
- If `-usage` is specified with any batch-related option then batch-related help is displayed.

-version

(Optional) Prints the SpyGlass version to the stdout and exits.

If several versions of SpyGlass are available on your company’s network, it is important to know which version of SpyGlass you are using.

To print the release number of the SpyGlass version you are using, enter the `-version` option in a SpyGlass command line:

```
spyglass -version
```

When the optional argument `-policies` (or `-policy`) is also specified with one or more registered product names, SpyGlass prints the version and the minimum required SpyGlass version for each specified product to stdout and exits.

Mode Selection Command-Line Options

The SpyGlass mode selection command-line options decide the SpyGlass operating mode (the GUI or the batch mode) and the SpyGlass operating language.

-32bit

Specifies SpyGlass to run in 32-bit mode.

-64bit

Specifies SpyGlass to run in 64-bit mode.

By default, 64-bit SpyGlass binaries are executed on 64-bit architectures

-batch

(Optional) Specifies to run SpyGlass in batch mode.

By default, SpyGlass starts in the GUI mode.

-gui

Specifies the GUI mode of SpyGlass.

When you specify `-gui=console`, Atrenta Console GUI is invoked.

Other Command-Line Options

-fullpolicy

(Optional) Specifies to run all rules (except those rules that are controlled by parameters, which run based on the parameter values) of all selected products.

All SpyGlass standard products have been reorganized to run only a selected set of rules when you select to run the complete product (by specifying the product name using the `-policy`/`-policies` command-line option and not specifying one or more rules of that product using the `-rules` command-line option). The selected set of rules has been chosen to quickly highlight the value of the particular product.

Earlier, such specification used to run all rules of the specified product. You can still run all rules (except those rules that are controlled by parameters, which run based on the parameter values) of a product by specifying the `-fullpolicy` command-line option.

NOTE: *There is no change in SpyGlass behavior when you use the* `-rules` *command-line option to specify rules to be run or when you run a goal.*

-policies | -policy

(Optional) Specifies a comma-separated list of products to be loaded.

SpyGlass comes with a number of standard products. Each product file is named as `<product-name>-policy.pl`. where `<product-name>` is the product name and is located in its own directory <your-inst-dir>/SPYGLASS_HOME/policy/<product-name>. For example, the product file for SpyGlass lint solution is lint-policy.pl, and is located in the <your-inst-dir>/

SPYGLASS_HOME/policy/lint directory.

NOTE: *If you specify a policy.pl file at the command-line that has an error, a fatal error message is displayed and the SpyGlass GUI is not invoked.*

NOTE: *Use of rule parameters at the command-line may affect the functional behavior of some rules and may determine if those rules should be run or not run. Refer to the corresponding product documentation for details on the rule parameters and the effect of the parameters on some rules.*

More information is available in the respective Rule Reference document for each product that can be accessed using *The spydocviewer utility* or *The spyhelpviewer utility*.

In addition, you can create your own customized products and use them just like SpyGlass standard products.

You can tell SpyGlass to use multiple products at one time.

When you apply a product, you are telling SpyGlass to check your design source files against the rules that are in this product.

To apply one product

To apply a single product on your design files, use the `-policy` command-line option or `-policies` command-line option as follows:

```
spyglass -batch -policy=<product-name> ...
spyglass -batch -policies=<product-name> ...
```

For example, to apply SpyGlass OpenMore solution, use the following command-line:

```
spyglass -batch -policy=openmore ...
```

To apply SpyGlass STARC solution, use the following command-line:

```
spyglass -batch -policies=starc ...
```

To apply more than one product

You can apply multiple products to a design file during a single run.

To apply multiple products on your design files, use the `-policy` command-line option or `-policies` command-line option as follows:

```
spyglass -batch -policy=<product-name-list> ...
spyglass -batch -policy=<product1-name>
  -policy=<product2-name> ...
```

```
spyglass -batch -policies=<product-name-list> ...
```

Where *<product-name-list>* is a comma-delimited list of product names (**with no spaces**).

For example, to apply SpyGlass lint Solution and SpyGlass OpenMore solution, use any of the following command-line options:

```
spyglass -batch -policy=lint,openmore ...
spyglass -batch -policy=lint -policy=openmore ...
spyglass -batch -policies=lint,openmore ...
```

NOTE: *You can run SpyGlass without any products by specifying the* `none` *value (for example,* `-policies=none`*). Then, only the SpyGlass built-in rules are run on your design files (HDL files, SpyGlass Design Constraints files, Library files, etc.), that is, your design files are effectively syntax-checked.*

NOTE: *Also, see the* `norules` *command that has a similar function.*

-run

Analyzes a design with the specified settings after invoking SpyGlass GUI.

Use the `-run` command-line option when you want to work in the SpyGlass GUI and want the design to be immediately analyzed after invoking the SpyGlass GUI.

You must provide all essential SpyGlass command-line options with the `-run` command-line option. Otherwise, the design will not be analyzed.

-template <goal-names>

NOTE: *Using this command is equivalent of using the -goals <goals> command or the* `current_goal` *<goal-name> project file command.*

Runs the specified goals.

Specify a goal name in the following manner:

```
-template <method-name>/<goal-name> ...
```

Where:

- *<method-name>* is the name of the methodology.
- *<goal-name>* is the name of a goal under the *<method-name>* methodology.

To run multiple goals, use the `-template` command in any of the following ways:

- Method 1

  ```
  -template <method1-name>/<goal1-name>
  -template <method2-name>/<goal2-name> ...
  ```

- Method 2

  ```
  -template "<method1-name>/<goal1-name>
            <method2-name>/<goal2-name>"
  ```

Methodology names are the names of directories containing goal files. The goal name is the first part of the name of a goal file.

For example, `clock_reset_integrity` goals present in the SpyGlass CDC methodology (that is, the <your-inst-dir>/SPYGLASS_HOME/Methodology directory) contains the following files for each language mode:

- `clock_reset_integrity-verilog.spq`
- `clock_reset_integrity-vhdl.spq`
- `clock_reset_integrity-mixed.spq`

To run these goals, specify the following commands:

```
spyglass -batch -vhdl -template Clock-reset/
clock_reset_integrity ...
spyglass -batch -verilog -template Clock-reset/
clock_reset_integrity ...
spyglass -batch -mixed -template Clock-reset/
clock_reset_integrity ...
```

Depending on the language specification, SpyGlass selects and runs the corresponding goal file.

SpyGlass searches for the specified goals in the following order:

1. The directory specified using the *-templatedir* command-line option (overrides the *DEFAULT_TEMPLATE_DIRECTORY* configuration file setting, if any)

2. Paths specified using the *-I* command-line option
3. In the <your-inst-dir>/SPYGLASS_HOME/Methodology directory

NOTE: *If the specified goals are not found in any of the above paths, SpyGlass prints the best possible matches for the goals, which are present in the above mentioned paths.*

You can specify the `-template=none` command-line option to indicate that no goals should be run during SpyGlass analysis. All goals specified after specifying the `-template=none` option are ignored. However, the goals specified before specifying the `-template=none` option are processed as before.

-templatedir

(Optional) Specifies the directory where the goals specified using the *-template <goal-names>* command-line option are to be searched.

If the `-templatedir` command-line option is specified, SpyGlass searches for the specified goals in the following order:

1. The directory specified using the `-templatedir` command-line option
2. The paths specified using the `-I` command-line option
3. In the <your-inst-dir>/SPYGLASS_HOME/Methodology directory

If the `-templatedir` command-line option is not specified, SpyGlass searches for the specified goals first in the directory specified using the *DEFAULT_TEMPLATE_DIRECTORY* configuration file setting and then in the other directories mentioned above.

NOTE: *The goal directory structure may contain one or more methodology directories within which the goals are present. In such cases, you should set the* `-templatedir` *option one level up than the methodology directory to see goals below each methodology. For example, if a goal,* `T`*, is present under the methodology directory,* `M`*, which is further present under the directory,* `D1/D2`*, you should set the value of the* `-templatedir` *option as* `D1/D2` *and the value of the -template option as* `M/T`*.*

-classic_mode

(Optional) Executes the CLASSIC mode rules only (which run on non-optimized netlist) and disables rules of EST and ESYNTH mode.

-est_mode

(Optional) Executes the EST mode rules only (which run on optimized and technology-mapped netlist) and disables rules of CLASSIC and ESYNTH mode.

-esynth_mode

(Optional) Runs only ESYNTH mode rules (which run on optimized netlist) and disables rules of CLASSIC and EST mode.

NOTE: *SpyGlass displays an error message if you specify a combination of -est_mode, -esynth_mode, or -classic_mode options.*

NOTE: *While running SpyGlass in -enable_save_restore mode along with any of the -est_mode/-esynth_mode/-classic_mode option, it is recommended that you specify different precompile work directory (using the -lib WORK <WORK-DIR> option) for each mode. If you do not specify a different precompile work directory for each mode, SpyGlass may synthesize the design every time you run SpyGlass in a different synthesis mode and re-save the database for each synthesis mode.*

-f

NOTE: *This command is equivalent of using the following project file command:*

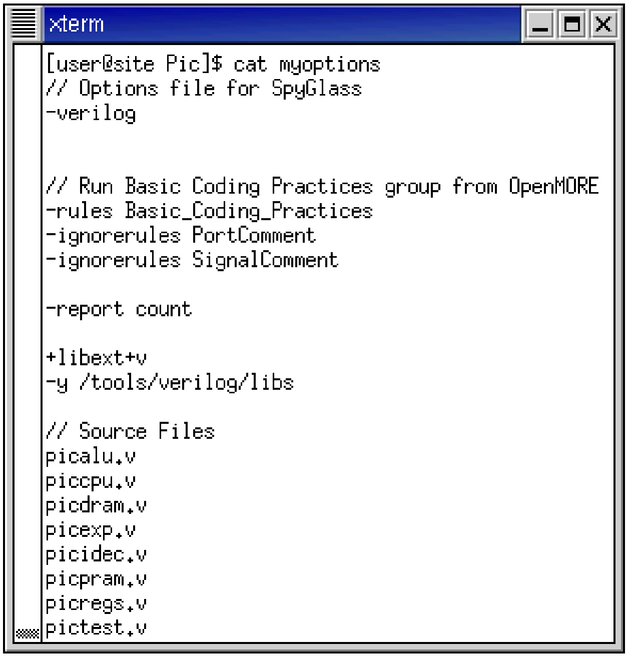
```
read_file -type sourcelist <file-names>
```

Example of Using the -f Command in Classic batch

The following example specifies the myoptions file present in the MYFILES directory:

```
spyglass -batch -f MYFILES/myoptions -verilog design.v
```

Where, myoptions contains the following details:



```
[user@site Pic]$ cat myoptions
// Options file for SpyGlass
-verilog


// Run Basic Coding Practices group from OpenMORE
-rules Basic_Coding_Practices
-ignorerules PortComment
-ignorerules SignalComment

-report count

+libext+v
-y /tools/verilog/libs

// Source Files
picalu.v
piccpu.v
picdram.v
picexp.v
picidec.v
picpram.v
picregs.v
pictest.v
```

FIGURE 1. myoptions File

-logfile

(Optional) Sets the name and location of the SpyGlass log file.

By default, the log file name is spyglass.log and it is created in the current working directory as specified with the *-wdir* command-line option (default is the current directory).

When you specify the `-logfile` command-line, the SpyGlass log file name and location are determined as follows:

- If the specified log file name is a simple file name (for example mylog.txt), the log file will be created as mylog.txt in the current working directory as specified with the *-wdir* command-line option (default is the current directory).
- If the specified log file name is an absolute file path name (for example /usr/john/logs/mylog.txt), the log file will be created as mylog.txt in the /usr/

john/logs directory irrespective of the *-wdir* command-line option specification.

NOTE: *The specified directory (the* /usr/john/logs *directory in the example) must exist.*

- If the specified log file name is a relative file path name (for example myprojects/mylog.txt), the log file will be created as mylog.txt in the myprojects directory under the current directory irrespective of the *-wdir* command-line option specification.

NOTE: *The specified directory (the* myprojects *directory in the example) must exist under the current directory.*

-rulegroup_display_depth

(Optional) Enables you to specify the number of the subgroups to be displayed in reports and message tree.

By default, the value of the `-rulegroup_display_depth` option is set to 2. You can change this value to specify different number of subgroups to be displayed.

If you specify the value of this option as 0 then all the subgroups are displayed.

NOTE: *The* `-rulegroup_display_depth` *option can be specified only if the* `display_rulegroup` *option is used.*

-wdir

NOTE: *This command is equivalent to the following project file command:*

```
set_option projectwdir <dir-name>
```

Specifies the output directory for SpyGlass if you want output to be generated at a location different from the current working directory.

Then, the SpyGlass output (SpyGlass schematics, Violation Database file, Log file, and Reports files) will be written to this new path.

Another application of the `-wdir` command-line option is to run SpyGlass from a read-only area and create the output files in another location.

If you do not specify this option, SpyGlass uses the current directory as the output directory.

NOTE: *The Verilog/VHDL precompiled libraries are always controlled by the* `-lib` *specification only.*

SpyGlass produces a number of different output files when it runs, including the Violation Database file, the log file and files for creating the schematics of the synthesized design for use in the SpyGlass (these are stored in a hidden directory <vdbfilename>.spysch). By default, all of these files are stored in the current directory from where SpyGlass was invoked.

It is possible to change the storage location of these output files using the `-wdir` option.

To change the current working directory

To have all SpyGlass output files stored in a different location from the current context, enter the `-wdir` option followed by the path to the new working directory, followed by the language, product to be checked, name of the design and any other desired options.

For example, to direct SpyGlass to store output files in directory mydir, enter:

```
spyglass -batch -wdir ../mydir -vhdl design.vhd
```

-gateslib

NOTE: *This command is equivalent to the following project file command:*

read_file `-type` *gateslib* `<file-name>`

(Optional) Specifies gate libraries containing functionality information about gate cells instantiated in a design.

NOTE: *If a file name includes wildcard characters (*, or ?), the name should be enclosed in single quotes, and the wildcard characters should be preceded by a backslash (\) to be treated as literal. For example, if your file name is '`abc*d`', you need to refer to it as '`abc\*d`'.*

However, if you want to refer to two files, for example '`abc1d`' and '`abc2d`', you can specify them using SpyGlass pattern matching support, that is, you can specify "`abc*d`" in this case. For details on pattern matching support, see Pattern Matching Across Features.

Example of Using the -gateslib Command in Classic Batch

The examples are described below. These examples are also applicable for the `spyglass_lc` utility.

- Specifying a single gate library

The following example specifies the mygates.lib Synopsys Liberty-format file to be analyzed:

```
spyglass -batch -gateslib mygates.lib
-enable_gateslib_autocompile -verilog -policy=erc
mydesign.v
```

- Specifying multiple gate libraries

  The following example specifies the mygates1.lib and mygates2.lib gate libraries:

```
// Method 1 (Using multiple -gateslib commands)

spyglass -batch -gateslib mygates1.lib \
  -gateslib mygates2.lib -enable_gateslib_autocompile
-verilog -policy=erc mydesign.v


// Method 2 (Using a single –gateslib command in which a
// space-separated list of library names are specified)

spyglass -batch -gateslib “mygates1.lib mygates2.lib” \
-enable_gateslib_autocompile -verilog -policy=erc
mydesign.v


// Method 3: Combining the above two methods:

spyglass -batch -gateslib “a.lib b.lib” \
-enable_gateslib_autocompile -verilog -policy=erc
mydesign.v -gateslib c.lib
```

- Specifying functionality information of a gate cell in a Verilog library file

  SpyGlass gives preference to a Verilog library file over Synopsys Liberty format file (.lib file) to pick the functionality information of a gate cell.

Example:

Consider the x module declared in the x.v file. The functionality information of this module is present in the lib1.v library file as well as the mylib1.lib file.

Now consider you specify the following command:

```
spyglass x.v -v lib1.v -gateslib mylib1.lib -verilog
```

In this case, SpyGlass picks the functionality information of the x module from the lib1.v library rather than the mylib1.lib file.

- Specifying the functionality information of a gate cell in a Verilog/VHDL design file

  Similar to the Verilog library file, SpyGlass gives preference to a Verilog/VHDL design file over Synopsys Liberty-format file (.lib file) to pick the functionality information of a gate cell.

  Example:

  Consider that the functionality information of the m module (declared in the m.v file) is present in the mod1.v file and mylib1.lib file.

  Now consider the following command:

  ```
  spyglass m.v mod1.v -gateslib mylib1.lib -verilog
  ```

  In this case, SpyGlass picks the functionality information of the m module (declared in the m.v file) from the mod1.v file rather than the mylib1.lib file.

  Alternatively, you can use the following command:

  ```
  spyglass_lc -gateslib <libfile-name>
     -lib <logical-lib-name> <physical-lib-name>
  ```

- Mixed-Language

  Re-run the SpyGlass Library Compiler specifying the Verilog and VHDL RTL description files directly (for both Verilog and VHDL), using the -v/-y/+libext command-line option (for Verilog), or using the -lib command-line option (for both Verilog and VHDL).

  In this case, the SpyGlass Library Compiler searches for cell descriptions first in the Verilog domain, then in the VHDL domain, and finally in the gates library files.

In case, the functional description for a cell is available both in the library and in a user-specified HDL source file, SpyGlass stores the functional view as follows:

Library	HDL	Functional View
Available and can be translated	Not specified	Library description is stored
Available and cannot be translated	Not specified	Only port interface from the library is stored
Available and can be translated	Specified	HDL description is stored
Available and cannot be translated	Specified	HDL description is stored
Available and can be translated	Specified but is not synthesizable	Library description is stored
Available and cannot be translated	Specified but is not synthesizable	Only port interface from the HDL is stored

-param

Specifies parameters used during Verilog/VHDL analysis run.

NOTE: *For Atrenta Console-specific usage of this command, see Set HDL Parameter(s) Value.*

Defining a Value for a Generic/Parameter

To set a value for a parameter, you must specify an instance of the parameter you wish to define together with the value you wish to set. In VHDL, this means the `entity_name.generic_name`, while in Verilog it means the `module_name.parameter_name`.

For example, to set the VHDL generic `width` in entity `control` to 8, enter the following as part of the SpyGlass invocation:

```
-param control.width=8
```

To set the Verilog parameter `limit` in module `block1` to 4, enter the following as part of the SpyGlass invocation:

```
-param block1.limit=4
```

Overriding the Value of a Generic/Parameter

In batch, you can override a generic/parameter value present in RTL by using the `-param` command-line option.

In case if any issues are encountered while overriding generic/parameter values, SpyGlass reports appropriate messages, as shown below:

```
Incorrect argument passed to -param. Correct usage is : '-param
<key>=<value>', where <key> is either <ent>.<gen> or
<mod>.<param>
```

```
'-param' option specified multiple times on the command line
for the entity/module "<entity/module-name>" generic/param
"<generic/parameter-name>" using the last value: '<value>'
```

Other than the above messages, SpyGlass also reports violation of the *CMD_param01*, *CMD_param02*, *CMD_param03*, *CMD_param04*, *CMD_param05*, and *CMD_param06* rules depending upon different cases. For details on these rules, refer to *BuiltIn Rules Reference Guide*.

Overriding Generics in VHDL

In VHDL, you can override the following type of values:

- Integer, positive, or a natural value, as shown in the following examples:

Integer value	`-param entity_name.generic_name=20`
Binary value	`-param entity_name.generic_name="2\"10\""`
Octal value	`-param entity_name.generic_name="8\"76\""`
Decimal value	`-param entity_name.generic_name="10\"93\""`
Hexadecimal value	`-param entity_name.generic_name="16\"AF\""`

- Boolean/enum value, as shown in the following examples:

Boolean value	`-param entity_name.generic_name=true`
Enum value	`-param entity_name.generic_name=red`

- Bits, as shown in the following examples:

Bit value	`-param 'entity_name.generic_name="0"'` Note that if single quotes around "0" are not present, the value is considered as the decimal value 0.
bit_vector value	`-param 'entity_name.generic_name="0000"'`

- std_logic, as shown in the following examples:

std_logic	`-param 'entity_name.generic_name="1"'` Note that if single quotes around "1" are not present, the value is considered as the decimal value 1.
std_logic_vector	`-param 'entity_name.generic_name="0000"'`

- String value, as shown in the following example:

String Value	`-param entity_name.generic_name=ABC`

- Aggregate value, as shown in the following example:

Aggregate Value	`-param entity_name.generic_name="others=>\"0\""`

Overriding Parameters in Verilog

In Verilog, you can override parameter values with the following type of values:

- Integer

  When you override a parameter value with an integer value or value in a valid based number format, it is considered as overriding the parameter value with an integer value.

  Following are some examples:

Integer	`-param "module_name.parameter_name=123"`
Binary	`-param "module_name.parameter_name=16'b10101010"`
Octal	`-param "module_name.parameter_name=32'o657"`

Decimal	`-param "module_name.parameter_name=16'd94"`
Hexadecimal	`-param "module_name.parameter_name=32'hAB4"`

NOTE: *If you pass a value in an invalid based number format, SpyGlass reports a violation.*

- String

  When you override a parameter with a value containing alphabets and/or special characters (with or without numerical values), it is considered as overriding with a string value.

  Following are some examples:

```
-param "module_name.parameter_name=12ab3"
-param module_name.parameter_name=abc
-param "module_name.parameter_name=abc"
```

-testsynth

NOTE: *This command is equivalent to the designread_enable_synthesis project file command:*

(Optional) Causes SpyGlass to elaborate and synthesize the design and report elaboration and synthesis messages.

Use the `-testsynth` command-line option to check the design for elaboration/synthesis issues without running any rules.

Thus, you must specify the norules command along with the `-testsynth` command-line option.

-debug_proc

(Optional) Dumps procedure call trace information for errors inside the procedure definition.

By default, the value of this command-line option is set to `no`, and SpyGlass does not dump any procedure call trace information.

To dump the procedure call trace information, specify `-debug_proc=yes` or `-debug_proc` on command-line.

Consider the following example (with line numbers highlighted):

```
1   # file1.sdc
2   proc proc2 { period_arg2 } {
```

```
3   create_clock -name CLK1 in1 $period_arg2
4   create_clock -name CLK2 in1
5   }
```

```
1   # file2.sdc
2   proc proc1 { period_arg1 } {
3   proc2 $period_arg1
4   }
5   proc proc0 { period_arg0 } {
6   # First proc
7   proc1 $period_arg0
8   }
9   proc0 10
```

For the above example, SpyGlass flags the following SDC errors for the `create_clock` command in the moresimple report:

```
SDC_106  Error    file1.sdc    3      10    Incorrect argument
"10" for "create_clock" (too many arguments?)
```

```
SDC_145  Error    file1.sdc    4       10    -period value
missing
```

However, if you want more information about the procedure call-trace, use the `-debug_proc` option. When you specify this option, SpyGlass generates the debugProcInfo file in the <wdir>/spyglass_spysch/spyglass_sdc/ directory and dumps the procedure call-trace information in this file in the following format:

```
SDC_106 File/Line : file1.sdc/3 (Procedure Trace : proc0
(file2.sdc,9) --> proc1 (file2.sdc,7) --> proc2 (file2.sdc,3)
)
```

```
SDC_145 File/Line : file1.sdc/4 (Procedure Trace : proc0
(file2.sdc,9) --> proc1 (file2.sdc,7) --> proc2 (file2.sdc,3)
)
```

+incdir

Searches the specified path for include files.

NOTE: *This command is equivalent to the incdir project file command.*

Examples of Using +incdir Command in Classic-Batch

Following are some examples:

- To specify directories containing `'include` files, use the `+incdir` option followed by the paths to the directories separated by + signs, the `-verilog` option and your design file name.

  For example, to specify the global directory, use the following command:

  ```
  +incdir+/u/<user_name>/global -verilog mydesign.v
  ```

- If the name of an `'include` directory contains the + character, specify that directory by using the `-incdir` command instead of using the `+incdir` command. For example, if the directory name is `abc+bcd`, specify this directory by using the following command:

  ```
  -incdir abc+bcd
  ```

- To specify multiple directories by using the `-incdir` command, specify a space-separated list of directories, as shown in the following example:

  ```
  -incdir "abc+bcd xyz"
  ```

- You can use the `+incdir` and `-incdir` commands together, as shown in the following example:

  ```
  -incdir abc+bcd +incdir+xyz+
  ```

NOTE: *If your directory name includes wildcard characters (*, or ?), the name should be enclosed in single quotes, and the wildcard characters should be preceded by a backslash (\) to be treated as literal. For example, if your directory name is '`abc*d`', you need to refer to it as '`abc\*d`'.*

However, if you want to refer to two directories, for example '`abc1d`' and '`abc2d`', you can specify them using SpyGlass pattern matching support, that is, you can specify "`abc*d`" in this case. For details on pattern matching support, see Pattern Matching Across Features.

`+define`

Adds the specified macro definitions.

NOTE: *This command is equivalent to the define project file command.*

Examples of Using +define Command in Classic-Batch

Following are some examples:

- To set the value of a ‘define macro, specify the `+define` command followed by the macro name and a value (separated by a + sign), the `-verilog` option and the design file name.

  For example, to set the macro State0 equal to 3, enter:

  ```
  spyglass -batch +define+State0=3 -verilog mydesign.v
  ```

- The following command sets the value of the `State0` and `State1` macros to `3` and `5`, respectively:

  ```
  +define+State0=3+State1=5 -verilog <file-name>
  ```

- The ‘define macros can also be included in the *.v files. However, these files must be analyzed by SpyGlass first before analyzing the remainder design files. Therefore, such files must be listed first on the command-line. For example, consider a define.v file that has the following format:

  ```
  // comment
  ‘define State0 3
  ‘define State1 5
  etc.
  ```

  In this case, define.v file must be listed first on the SpyGlass command-line, as shown below:

  ```
  spyglass -batch -verilog define.v design.v
  ```

--gdb

(Optional) Invokes GDB (GNU Project Debugger) during the SpyGlass run.

Use the `--gdb` option to debug your custom rules.

You must have the GDB tool suite installed in your file system.

The GDB tool suite is searched in the following order:

1. Path to the GDB executable set using the `SPYGLASS_GDB_PATH` environment variable
2. The /usr/local/bin directory
3. The /usr/bin directory
4. Path set in your `PATH` environment variable

SpyGlass Configuration File Setting Override Options

SpyGlass has the Configuration File feature using which you can specify configuration settings like default startup mode (the SpyGlass GUI or batch), default product to be run, default language setting, default report format etc.

The Configuration File settings can be overridden by specifying certain command-line options directly on the command-line or indirectly in a command file as follows:

Configuration File Setting	Value	Overriding Command-line Option
DEFAULT_STARTUP_MODE	gui	*-batch*
	batch	-gui
USE_32_BIT_EXECUTABLE_ONLY	no	
	yes	-32bit
DEFAULT_LANGUAGE_MODE	VHDL	-verilog, -def, or -mixed
	Verilog	-vhdl, -def, or -mixed
	Mixed	-verilog, -vhdl, or -def
	DEF	-verilog, -vhdl, or -mixed
	none	-verilog, -vhdl, -def, or -mixed
DEFAULT_TEMPLATE	Any	-template
DEFAULT_POLICY_FOR_SPYEXPLAIN	Any	-policies | -policy
DEFAULT_REPORT_FORMAT	default, <report-name>	-report_per_policy, -noreport
	none	-report_per_policy
DEFAULT_PRAGMA	default, <pragma-name-list>	-pragma=<pragma-name-list>, -pragma=nopragma
	none	-pragma=<pragma-name-list>
VHDL_LIB_MAP	Any	-lib

Configuration File Setting	Value	Overriding Command-line Option
COMMAND_OPTION_FILENAME		Additive effect hence not possible
COMMAND_FILE_ARGS		Additive effect hence not possible
SYSTEMVERILOG_SUPPORT	no	-enableSV
	yes	-disableSV
AUTOENABLE_INFERBLACKBOX	no	-inferblackbox, +libext
	yes	+libext, -disable_inferblackbox
	yes_netlist	-disable_inferblackbox
	yes_rtl	-inferblackbox, -disable_inferblackbox
AUTOENABLE_VHDL_SORT	no	-sort
	yes	-disable_sort
DEFAULT_VHDL_SORT_METHOD	lexical	-sort
	no argument	-sort=lexical

NOTE: *The command-line options that do not have an associated description have been provided only to override the Configuration File settings.*

Command-Line Utilities

The spyexplain Utility

SpyGlass provides the `spyexplain` utility that displays information about specified rules or rule parameters. Sometimes, you will need to find which product contains which rules, for instance. Moreover, you may not know exactly what rules are available for checking clocks in a particular product. Then, you can use the `spyexplain` utility to display a report that shows where rules are defined and gives a brief description of their function.

NOTE: `spyexplain` *is a separate utility with its own options and it is NOT an option to the* `spyglass` *application.*

Searching Rules

The `spyexplain` utility has a number of options, allowing you to define the HDL language, the product (or products you wish to search) and a keyword for which you wish `spyexplain` to search in the rule description and name.

The syntax of using the `spyexplain` command to search rules is as follows:

```
spyexplain
  -verilog | -vhdl | -mixed | -def
  [ -policies | -policy = {<product-name>,} ]
  [ { <rule-name> } ]
    | [ -k <search-string> [ -searchlonghelp ] ]
  [ -I <path> ]
  [ -include_builtins ]
```

Where the `-policy`, `-policies`, and `-I` command-line options work same as the corresponding SpyGlass command-line options, *`<rule-name>`* is a rule name or rule alias name (case-insensitive), and *`<search-string>`* is a valid string.

If you do not specify a product in the command-line, the spyexplain utility searches the installed products.

Use the -include_builtins argument to search the SpyGlass Built-in rules (HDL Parsing rules, SpyGlass Design Constraints file Parsing rules, and Library (.lib) Files Parsing rules).

The spyexplain utility searches both the rule name and short help message fields for matches but does not search extended help message fields unless the -searchlonghelp command-line option is supplied. The search mechanism is not case-sensitive, but partial words are also located during the search.

The spyexplain utility reports the following information for each rule that matches the specified search criteria:

- The rule name
- The language to which that rule applies
- The product in which the rule is described
- A short description of the rule
- An extended description of the rule

Examples of Searching Rules

For example, to see the rules in the VHDL SpyGlass OpenMore solution, use the following command-line:

```
spyexplain -vhdl -policies=openmore
```

To see the rules in the Verilog SpyGlass lint solution, use the following command-line:

```
spyexplain -verilog -policies=lint
```

To find out about a particular rule

For example, to see information about the W703 rule in a Verilog SpyGlass lint solution, use the following command-line:

```
spyexplain -verilog -policies=lint W703
```

To locate a particular rule in a product

To locate a rule in a product or to search for all rules in a product that relate to a required check (for example, all clock-related rules), you need to search for a word that is contained in the rule name or the short help message.

To search for a specific word in a product, enter the `-k` (keyword) command-line option of the `spyexplain` utility.

For example, to search for the keyword `Reset` in the Verilog SpyGlass lint solution, use the following command-line:

```
spyexplain -k Reset -verilog -policies=lint
```

To search for the keyword `Clock` in the VHDL SpyGlass lint solution, use the following command-line:

```
spyexplain -k Clock -vhdl -policies=lint
```

Searching Rule Parameters

The `spyexplain` utility has a number of options, allowing you to search a rule parameter in a specified product (or products).

The syntax of using the `spyexplain` command to search rule parameters is as follows:

```
spyexplain
  -verilog | -vhdl | -mixed | -def
  [ -policies|-policy={<product-name>,} ]
  [ -I <path> ]
  [ -param <param-name-list> ]
```

Where the `-policy`, `-policies`, and `-I` command-line options work same as the corresponding SpyGlass command-line options and *`<param-name-list>`* is a space-separated list of valid strings.

If you do not specify a product using the `-policies/-policy` option, the `spyexplain` utility searches all products. If you specify a product name using the `-policies/-policy` option and specify the `-param` option without any string, the `spyexplain` utility reports all rule parameters in the specified product.

NOTE: *You must specify at least one of the* `-policies/-policy` *and* `-param` *options.*

The `spyexplain` utility searches both the rule parameter name and short help message fields for matches. The search mechanism is not case-sensitive, but partial words are also located during the search.

The `spyexplain` utility reports the following information for each rule parameter that matches the specified search criteria:

- The rule parameter name
- The language to which that rule parameter applies
- The product in which the rule parameter is described
- The description of the rule parameter

Searching Goals

You can also list the description of all rules in a specified goal using the `-template` option of the `spyexplain` utility.

The syntax of using the `spyexplain` command to list description of rules in a goal is as follows:

```
spyexplain
  -verilog | -vhdl | -mixed | -def
  [ -I <path> ]
  [ -template <methodology-name>/<goal-name> ]
```

Where the `-I` command-line option works same as the corresponding SpyGlass command-line option, *<methodology-name>* is the name of the methodology and *<goal-name>* is the name of the goal.

For example, if you want the description of all rules in the SpyGlass standard Block-Design/Creation goal then specify as follows (for different languages):

```
spyexplain -template Block-Design/Creation -verilog
spyexplain -template Block-Design/Creation -vhdl
spyexplain -template Block-Design/Creation -mixed
```

The `spyexplain` utility searches for the specified goal in the user-specified -I paths.

Searching SpyGlass Design Constraints

You can search the description of standard SpyGlass Design Constraints using the spyexplain utility.

The syntax of using the spyexplain command to search standard SpyGlass Design Constraints is as follows:

```
spyexplain
  [ -policies|-policy={<product-name>,} ]
  [ -I <path> ]
  [ -sgdc <constraint-name-list> ]
```

Where the -policy, -policies, and -I command-line options work same as the corresponding SpyGlass command-line options and *<constraint-name-list>* is a space-separated list of valid strings.

If you do not specify a product using the -policies/-policy option, the spyexplain utility searches all products. If you specify a product name using the -policies/-policy option and specify the -sgdc option without any string, the spyexplain utility reports all SpyGlass design constraints in the specified product.

NOTE: *You must specify at least one of the* -policies/-policy *and* -sgdc *options.*

The spyexplain utility first searches for the exact match by the design constraint name. If not found, the utility searches in the registration and long help of all design constraints in the specified products for matches. The search mechanism is not case-sensitive, but partial words are also located during the search.

The spydocviewer utility

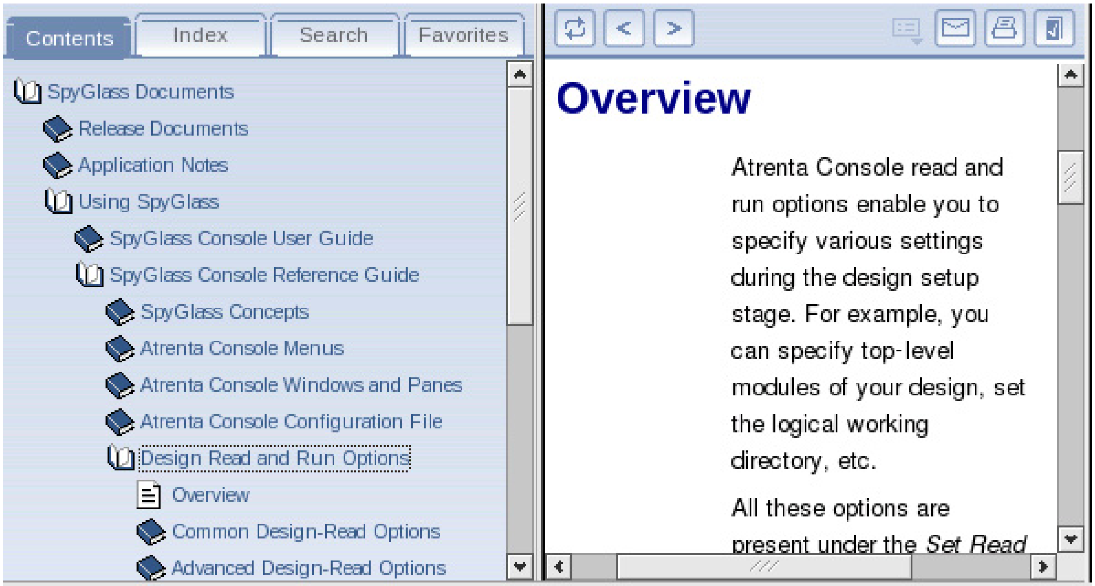
The spydocviewer utility displays the SpyGlass documentation in a tree format for easy access. In addition, the SpyGlass standard Rule-Primitive documentation (in text format) is also accessible.

By default, the spydocviewer utility searches for the acroread or xpdf executables in your machine's path for displaying the PDF files. Set the SG_PDF_VIEWER environment variable to set your PDF viewer.

The spyhelpviewer utility

The spyhelpviewer utility displays the SpyGlass documentation in HTML format arranged in a tree format for easy access.

By default, the spyhelpviewer utility searches for the netscape executable in your machine's path for displaying the HTML files. Use the SG_HTML_BROWSER environment variable to set your HTML Browser.


Contents
Index
Search
Favorites
SpyGlass Documents
Release Documents
Application Notes
Using SpyGlass
SpyGlass Console User Guide
SpyGlass Console Reference Guide
SpyGlass Concepts
Atrenta Console Menus
Atrenta Console Windows and Panes
Atrenta Console Configuration File
Design Read and Run Options
Overview
Common Design-Read Options
Advanced Design-Read Options

Overview

Atrenta Console read and run options enable you to specify various settings during the design setup stage. For example, you can specify top-level modules of your design, set the logical working directory, etc.

All these options are present under the *Set Read*

FIGURE 2. SpyGlass HTML-based On-line Help System

The HTML-based online Help system has the following:

- The Contents tab that shows all topics arranged in a hierarchical tree
- The Index tab that has the index entries for the complete documentation set
- The Search tab for search across the complete documentation set
- The Favorites tab for collecting a set of related topics for viewing

- The Topic display page

System Requirements

The HTML-based On-line Help system works on a UNIX computer running version 4 or later of Internet Explorer or Netscape or a current version of Mozilla, or Safari. The underlying engine has also been tested with Mozilla. JavaScript must be enabled in the user's browser.

To view the Java implementation of the Help system, Java must be enabled in the user's browser.

Known Limitations

1. Netscape 6.0 is not supported on any platform; Netscape 6.1 and later are supported.
2. The underlying engine may also work with Opera and other browsers, but it has been tested only with Internet Explorer, Netscape, Mozilla, and Safari.

Reports Generated in Atrenta Console

Overview

When SpyGlass analysis is complete, the following types of reports are generated:

- *General Reports*
- *Default Reports*
- *Custom Reports*
- Product-specific reports

  These reports are generated based on selected goals. Refer to the respective product documentation for details on product-specific reports.

You can open the required report to view the summary of design issues, or you can redirect these reports in separate files to review them later.

Generating Reports

To generate a report, perform any of the following actions:

- Select the *Tools -> Report* menu option.

  A sub menu appears listing different categories of reports. Select a report from a category. The selected report appears in a new window.

- Select the *Reports* option or the 💾 icon from the *Results Pane*. A menu appears listing different categories of reports. Select a report from a category. The selected report appears in a new window.
- Specify the report to be generated by using the *report* command in a project file.
- Specify the report to be generated by using the *Reports Name* option under the *Set Read Options* tab.

General Reports

Atrenta Console generates the following reports when you specify them by using the *report* command:

count report	*inline report*
moresimple_filesort report	*moresimple_rulesort report*
moresimple_sevclass report	*score report*
sign_off report	sign_off_sevlabel report
simple report	*summary report*

count report

The count report lists the number of times SpyGlass found each type of message. It also displays the total number of messages of rules you previously waived or excluded.

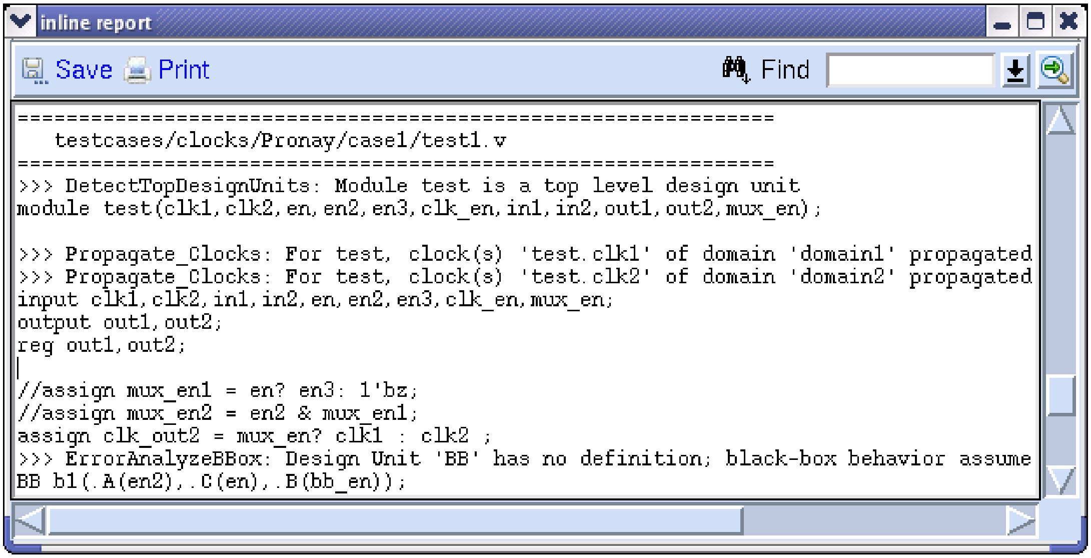
inline report

The inline report displays the contents of the RTL source file annotated with violation messages. These messages are displayed above the violating line in the source code.

To help you spot the messages, Atrenta Console prefixes these messages with >>>.

NOTE: *The Inline report is not generated properly when the compressed netlist is used in the design.*

Following figure shows a sample inline report.



```
==============================================================
   testcases/clocks/Pronay/case1/test1.v
==============================================================
>>> DetectTopDesignUnits: Module test is a top level design unit
module test(clk1,clk2,en,en2,en3,clk_en,in1,in2,out1,out2,mux_en);

>>> Propagate_Clocks: For test, clock(s) 'test.clk1' of domain 'domain1' propagated
>>> Propagate_Clocks: For test, clock(s) 'test.clk2' of domain 'domain2' propagated
input clk1,clk2,in1,in2,en,en2,en3,clk_en,mux_en;
output out1,out2;
reg out1,out2;
|
//assign mux_en1 = en? en3: 1'bz;
//assign mux_en2 = en2 & mux_en1;
assign clk_out2 = mux_en? clk1 : clk2 ;
>>> ErrorAnalyzeBBox: Design Unit 'BB' has no definition; black-box behavior assume
BB b1(.A(en2),.C(en),.B(bb_en));
```

FIGURE 1. Example of inline Report

moresimple_filesort report

The moresimple_filesort report is similar to the moresimple report except that the rule messages are sorted in the following order:

1. File name for a given file
2. Severity
3. Weight
4. Rule
5. Line number

moresimple_rulesort report

The moresimple_rulesort report is similar to the moresimple report except that the rule messages are sorted by the rule name first and for a given rule, by their file and line.

moresimple_sevclass report

The moresimple_sevclass report is similar to the moresimple report with additional information displaying the severity class.

score report

SpyGlass provides a built-in scoring system for code checks. The report generator assigns a weight to each rule message based on the severity of the message. The score report displays the score for a design as the sum of the weights assigned to all the rule messages that are detected during a SpyGlass Analysis run. The overall score is an indicator of the relative quality of your design.

sign_off report

The sign_off report lists summary and detailed information about the SpyGlass analysis run.

This report has the following sections:

- The header section that displays SpyGlass version, products versions, user name, top-level design unit name(s) with file name and line number information, goals used, if any, the current working directory, counts of messages generated, messages waived, if any, messages reported, and messages suppressed, if any.
- The Rule Setup Info section that displays a list of SpyGlass commands with their actual values used in the analysis run passed. The source file details are printed at the end of the list followed by unknown options, if any.
- The Policy Info section has the names, versions, and locations of selected products.
- The Constraints Info section has the details of user-specified SpyGlass Design Constraints.
- The Parameter Info section has the list of user-specified rule parameters with their applied values and origin (command file, goal/session file, or SpyGlass Configuration file).

- The Waiver Info section has the list of applied waivers with waiver comment, if any, waiver expression, and the number of messages waived. This section also contains the spg_backref field. This field is used to specify the back reference information about the waiver command, that is, the file and line number of the waiver file in which the waiver command was specified.
- The Violation Info section has two sub-sections:
  - ❒ The Summary Status sub-section has severity class, rule name, message count, and rule short description for each rule that has flagged a message. The list is sorted by the severity class.
  - ❒ The Detailed Status sub-section has same details as that of the moresimple report.

sign_off_sevlabel report

The sign_off_sevlabel report is similar to the sign_off report, but the Summary Status sub-section of the Violation Info section displays rule information along with severity label instead of severity class.

simple report

The simple report truncates long names and messages to fit the contents in the report’s layout.

This report displays the following information:

- ❒ Name of the rule violated
- ❒ Name of the file where SpyGlass found the message
- ❒ Line number of the code where the corresponding rule violation occurred
- ❒ Severity level of the message
- ❒ Message explaining why SpyGlass logged the message

Note that the severity level and message are contained in the rule definition.

summary report

The Summary report displays a summary list of message counts by each particular rule type along with the severity class and rule short help.

The rules are grouped based on the default grouping criteria. Within each group, the data is first sorted by the severity class and then by rule name.

summary_sevlabel report

The summary_sevlabel report is similar to summary report, but it displays summary list of message counts by each particular rule type along with severity label instead of severity class.

Default Reports

Atrenta Console generates the following default reports:

ignore_summary report	*moresimple report*
moresimple_warning report	*moresimple_error report*
no_msg_reporting_rules report	*stop_summary report*
elab_summary report	*waiver report*

ignore_summary report

The ignore_summary report lists design units/files that are ignored in the current run.

These design units/files are specified by the *ignoredu* and *ignorefile* project file commands.

Following is the sample ignore_summary report:

```
Design units ignored by ignoredu/ignorefile options
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Object Name     Object Type      Ignore specification
===========================================================
mid             MODULE           ignoredu     "mid"
other_bot       MODULE           ignorefile     "src/bot.v"

Files ignored by ignorefile options
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
src/bot.v (ignorefile "src/bot.v")

Unused ignorefile/ignoredu specification(s)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ignoredu "no_such_dir"
```

moresimple report

The moresimple report is similar to the *simple report*. However, it does not truncate long names and messages.

The moresimple report displays the following information:

- Name of the rule violated
- Alias of the rule

NOTE: *An alias refers to an alternative name. It is a cross-reference to a standards document, such as the lint standard or in-house coding standard or guideline.*

- Severity level of the message
- Name of the file where SpyGlass found the message
- Line number of the code containing the message
- Short help and rule name for each rule group

NOTE: *Set the report_style option in project file to include the short help and rule name for each rule group as shown below:*

```
set_option report_style {group_moresimple_by_rule}
```

- Weight of the message

NOTE: *The weight is a number assigned based on the severity of the rule message. That is, higher the severity of the message, higher will be the weight assigned. It is used when SpyGlass calculates lint assessment of your design.*

- A message explaining why SpyGlass logged the message

NOTE: *The alias, severity level, weight, and message are all contained in the rule definition.*

Messages in the moresimple report are sorted in the order of Severity Class, Severity Label, Rule Name, File Name, and Line Number.

NOTE: *SpyGlass does not elaborate the dead code of a module, but performs parsing of such code. However, violations due to parsing are not reported in the moresimple report. Such violations are captured in the spyglass.log file.*

The SDC mode grouping in the moresimple report is switched off, by default. To enable the SDC mode, specify the following command:

```
set_option report_style display_sdcgroup
```

Whenever you run SpyGlass analysis without specifying a report format to be generated, SpyGlass automatically generates the *moresimple report*. You can change the automatic report format by specifying the name of the report format you want to create to the DEFAULT_REPORT_FORMAT key.

moresimple_warning report

The moresimple_warning report is similar to the moresimple report and displays all the Info and Warning violations in the design run. Use the following command in the project file to generate the moresimple_warning report:

```
set_option report {moresimple_warning}
```

moresimple_error report

The moresimple_error report is similar to the moresimple report and displays all the Error and Fatal violations in the design run. Use the following command in the project file to generate the moresimple_error report:

```
set_option report {moresimple_error}
```

no_msg_reporting_rules report

The no_msg_reporting_rules report displays a list of rules that did not report any violation or waived during SpyGlass run

spyglass_violations report

The spyglass_violations report is similar to the *moresimple report* with the following distinguishing features:

- The spyglass_violations report does not show `Alias` (Rule alias name) and `SEV_CLASS` (Rule Severity Class name).
- The spyglass_violations report does not show messages with the severity as `Info`.
- Optionally, set the value of the `sg_viol_rpt_no_warnings` parameter to `1` to eliminate messages of `Warning` severity from the `spyglass_violations` report.

  For example, to eliminate warning messages and display only high severity messages, such as, Fatal and Error messages in the `spyglass_violations` report, specify the following command in the project file in the global scope:

```
set_parameter sg_viol_rpt_no_warnings 1
```

stop_summary report

The stop_summary report lists design units and files that are skipped from SpyGlass checking in the current run.

These design units and files are specified by the *stop*, *stopdir*, and *stopfile* project file commands.

Following is the sample stop_summary report:

```
Design units stopped by stop/stopfile/stopdir options
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Object Name      Object Type   Stop specification
===========================================================
sr               MODULE       stopfile"bench/benchcell.v"
reorder_bits     MODULE       stopfile "bench/benchcell.v"
clcell           MODULE       stopfile "bench/benchcell.v"
clcell2          MODULE       stopfile "bench/benchcell.v"
expreg_dec       MODULE       stopfile "bench/benchcell.v"
exple_dec        MODULE       stopfile "bench/benchcell.v"

Files stopped by stopfile/stopdir options
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
bench/benchcell.v (stopfile "bench/benchcell.v")

Unused stop/stopfile/stopdir specification(s)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
stopdir "no_such_dir"
```

elab_summary report

The elab_summary report shows the following design information:

- RTL design unit names
- Elaborated names

- Parameter values

waiver report

A waiver report is generated when messages are waived during SpyGlass run.

This report has the following three sections:

- The *Design Issues Waiver Report* section has actual waived messages waived due to arguments other than the `-ignore` argument or the `-ip` argument of the `waive` constraint
- The *IP/Legacy Waiver Report* section has the waived message count due to the `-ip` argument of the `waive` constraint and the actual waived messages waived if the *report_ip_waiver* command is specified.
- The *Adjustments Waiver Report* section has the waived message count due to the `-ignore` argument of the `waive` constraint and the actual waived messages waived if the *report_adjustment_waiver* option is specified.

The Waiver report also contains the spg_backref field. This field is used to specify the back reference information about the waiver command, that is, the file and line number of the waiver file in which the waiver command was specified.

Report Generated When a Module is Instantiated in Multiple IPs

If SpyGlass reports a violation on a module instantiated in multiple IPs and you have set the *use_du_sch_hier* command to `yes`, you must specify the `waive -ip` command for the all IPs containing its instantiation to waive such violations.

In this case, the waiver report contains the following two sections for the `waive -ip` command:

- The first section shows violations completely waived by the current `waive` command.
- The second section shows violations waived by the current `waive` command along with the other `waive -ip` commands.

NOTE: *Imported waive commands are not displayed in a separate section in the waiver report.*

For example, consider a design containing the `clcell` module that is instantiated in the `mid` and `mid2` IPs. In addition, the design contains a black box instance of the `reorder_bits` module.

Now, when you specify the `waive -ip mid` and `waive -ip mid2 -rule ErrorAnalyzeBBox` commands, the following waiver report is generated:

```
waive -ip mid
#spg_backref : "test.swl" 1

1 message(s) waived by current waive -ip command, 1 message(s) waived
along with other waive -ip commands:
================= MESSAGES WAIVED BY CURRENT WAIVE IP COMMAND===========
Index Rule            Severity File      Line Wt  Message
=======================================================================
[2]   WarnAnalyzeBBox Warning  test.v 1  10       Design Unit 'IBUF'
                                                  has empty definition

============== MESSAGES WAIVED ALONG WITH OTHER WAIVE IP COMMANDS=======
Index Rule              Severity  File       Line Wt  Message
=======================================================================
[4]*  ErrorAnalyzeBBox  Error     test.v 49  10       Design Unit
                                                     'reorder_bits'
                                                     (elaborated name
                                                     'reorder_bits_7')
                                                     has no definition;
                                                     black-box behavior
                                                     assumed
=======================================================================


waive -ip mid2 \
      -rule ErrorAnalyzeBBox
#spg_backref : "test.swl" 2

1 message(s) waived along with other waive -ip commands:

============ MESSAGES WAIVED ALONG WITH OTHER WAIVE IP COMMANDS=========
Index Rule              Severity  File    Line Wt  Message
=======================================================================
[4]*  ErrorAnalyzeBBox  Error     test.v  49   10  Design Unit
                                                   'reorder_bits'
```

```
                                                                     (elaborated name
                                                                     'reorder_bits_7') has
                                                                      no definition;
                                                                      black-box behavior
                                                                      assumed
==========================================================================
```

Custom Reports

In addition to the standard reports, Atrenta Console also provides custom reports that are designed for specific customer requirements but can be used by all SpyGlass users.

You can generate these reports by using the following commands in the project file:

```
set_option report <custom-report-name>
set_option I {space-separated list of directory-name}
```

NOTE: *The I command is not recommended to be used in the project file. For details, see Options Not Recommended. However, you should specify the directory paths using the -I option on the command-line itself while invoking console or sg_shell.*

You can view the custom reports by selecting the *Report > Default* menu option.

NOTE: *Custom reports are available on AS IS basis.*

The count_sevsort Report

The count_sevsort report is an enhanced version of the *count report* and has the rule message count for each rule of each severity label grouped under their severity class.

For each rule, the severity label name, the rule name, and the rule message count are reported.

The groupings are reported in decreasing severity class (FATAL, ERROR, WARNING, and INFO). Under each severity class, the rule message count is sorted based on the severity label (alphabetically), and then on Rule name (alphabetically).

The moresimple_csv Report

The moresimple_csv report has the same details as the *moresimple report* but in a comma-separated format without the header/footer lines.

In turbo mode, SpyGlass generates the `moresimple_report.csv` report, which is similar to the `moresimple_turbo` report, but in comma-separated format. The `moresimple_report.csv` report includes the parent ID of the secondary violations. For primary violations, SpyGlass reports the parent ID is 'N.A.'.

You can open the generated moresimple_csv.rpt file in a spreadsheet program, such as Microsoft Excel.

The score_detail Report

The score_detail report is an enhanced version of the *score report* and has the following sections:

1. The first section has the detailed status showing the rule name, the rule severity label, the rule message count multiplied by the rule weight, and the rule score. The details are listed sorted by the severity class (not reported) and alphabetically within the severity class.
2. The second section has the summary status showing number of rules flagged under each severity class and the total score for the severity class.

The report has the grand total score for the design at the end.

Block Dependency Report

The block dependency report provides a complete list of files and options that are required to perform successful SpyGlass analysis of a particular design unit. The generated list of options can be used with other synthesis and simulation tools as well.

For each design unit, this report is split in two different files. For details, see *Files Generated in this Flow*.

NOTE: *Although there are related reports, such as elab_summary and Audit that generate a hierarchical view of a design along with their file name, however, none of these reports gives a complete list of dependent files and options for a design unit. To get this information, you can use the block dependency report.*

Generating the Block Dependency Report

To generate the block dependency report for certain design units, use the following command in the project file:

```
set_option gen_block_options { <list-of-blocks> }
```

The following example shows the usage of this command:

```
set_option gen_block_options { top DU1 }
```

Files Generated in this Flow

When you specify design units by using the gen_block_options command, Atrenta Console generates the following report files for each specified design unit:

- /spyglass_reports/<design-unit-name>/std_opts.f

  This file contains standard options, such as v, y, libext, define, incdir, libhdlfiles, libhdlf, lib, and libmap.

  You can use the std_opts.f file in SpyGlass run as well as use it across other tools.

- /spyglass_reports/<design-unit-name>/sg_opts.f

  This file contains the following type of commands:

  - Design read options that were specified to provide various other files and directories. These options include param, pragma, stop, and top.
  - Rule-checking options, such as sgdc and waivers.

  This file is meant strictly for SpyGlass and cannot be used with other tools.

Usability Aspects of Dependency Reports

The following points discuss some usability aspects of the block dependency report:

- All rule-checking options, such as waiver and sgdc are generated in the sg_opts.f file as it is. You may have to edit some of these options in this file in the subsequent SpyGlass runs.
- Options, such as libhdlfiles, libhdlf, and lib are decompiled as is in the std_opts.f file because they do not have any negative impact on the analysis of blocks specified by using the gen_block_options command.
- You should specify the *top* command when using the internally generated std_opts.f file for the top-level design unit in subsequent SpyGlass runs.
- Design read options that impact any design unit within the hierarchy of the block specified by using the gen_block_options command should not be changed while using the internally generated std_opts.f dependency file for a subsequent SpyGlass run.

For example, consider design two units du1 and du2, and du1 contains an instance of du2. Now consider that in the original SpyGlass run, you specify the following commands in the project file:

```
set_option gen_block_options du1
set_option stop du2
```

In this case, the <du1>/std_opts.f file is generated.

Now if you again run SpyGlass and pass the <du1>/std_opts.f file in this run, do not change the stop command specification. This is because the dependencies were generated with respect to the netlist without du2, and therefore, the dependencies of du2 will not be considered in the original run and not generated in the <du1>/std_opts.f file.

Consider another example in which you want to generate dependencies for the du1 block and it has the parameter P1. In this case, ensure that the value of P1 should remain same across original SpyGlass run and subsequent SpyGlass runs, as shown below:

Original SpyGlass run	`set_option gen_block_options du1`
	`set_option param "du1.P1=8"`

Subsequent SpyGlass run	`read_file -type sourcelist <du1>/ std_opts.f`
	`set_option param "du1.P1=8"`

Specifying different values for P1 might activate different branches of code within du1 and not all dependencies might be accounted for a particular value of P1.

Directory Structure of Generated Reports

After SpyGlass run, Atrenta Console creates the following two parallel directory structures:

- The directory structure containing all the generated reports.
- The consolidated directory structure containing links to the reports present in the above structure.

  This directory structure is a simple and a compact structure as compared to the above directory structure. For example, when you run multiple goals, the above directory structure generated is very nested and the user has to search for a report deep down the hierarchy. To solve this problem, SpyGlass provides the consolidated directory structure that is a simplified flattened version of this directory structure.

  By default, Atrenta Console creates this structure at the following path:

  <project-working-directory>/<project-name>/consolidated_report/

  To specify a different path, use the *consolidate_reportdir* project file command.

Directory Structure Generated After Design-Read

When you run the design read process, Atrenta Console generates reports under the following directory structure:

<project-working-directory>/<project-name>/Design_Read/spyglass_reports/

Path in the Consolidated Directory Structure

The following directory structure contains links to the reports:

<project-working-directory>/<project-name>/consolidated_report/Design_Read/

Directory Structure Generated when a Top-Level Module is Specified

If you specify a top-level module, Atrenta Console generates reports under the following directory structure:

<project-working-directory>/<project-name>/<top-module-name>/
Design_Read/spyglass_reports/

NOTE: *You can specify a top-level module in the Top Level Design Unit field under the Set Read Options tab or by using the top project file command.*

Path in the Consolidated Directory Structure

The following directory structure contains links to the reports:

<project-working-directory>/<project-name>/consolidated_report/
<top-module>_Design_Read/

The following example shows the directory structure generated when you specify the top-level module as `moduleA`:

case1/Project-1/consolidated_report/moduleA_Design_Read/

Directory Structure Generated After a Goal Run

When you run goals, Atrenta Console generates reports for each goal run under the following directory structure:

<project-working-directory>/<project-name>/<goal-path>/
spyglass_reports/<product-name>/

The following example shows the directory structures generated when you run the initial_rtl/lint/structure goal:

case1/Project-2/initial_rtl/lint/structure/spyglass_reports/lint/

Path in the Consolidated Directory Structure

The following directory structure contains links to the report:

<project-working-directory>/<project-name>/consolidated_report/
<goal-path-name>/

The following example shows the directory structure generated when you run the initial_rtl/lint/structure goal:

case1/Project-1/consolidated_report/initial_rtl_lint_structure/

Directory Structure Generated when a Top-Level Module is Specified

If you specify a top-level module, Atrenta Console generates reports for each goal run under the following directory structure:

<project-working-directory>/<project-name>/<top-module-name>/<goal-path>/spyglass_reports/<product-name>/

NOTE: *You can specify a top-level module in the Top Level Design Unit field under the Set Read Options tab or by using the top project file command.*

Path in the Consolidated Directory Structure

The following directory structure contains links to the reports:

<project-working-directory>/<project-name>/consolidated_report/<top-module>_<goal-path-name>/

The following example shows the directory structure generated when you specify the top-level module as `moduleA` and run the initial_rtl/lint/structure goal:

Project-1/consolidated_report/moduleA_initial_rtl_lint_structure/

Directory Structure Generated After a Scenario Run

If you run a scenario, Atrenta Console generates reports for each scenario run under the following directory structure:

<project-working-directory>/<project-name>/<goal-path>/<scenario-name>

The following example shows the directory structure generated for the `S1` scenario created from the initial_rtl/cdc_exhaustive/cdc_verif_strict goal:

case1/Project-1/top/initial_rtl/cdc_exhaustive/cdc_verif_strict/S1/spyglass_reports/

Path in the Consolidated Directory Structure

The following directory structure contains links to the reports:

<project-working-directory>/<project-name>/consolidated_report/<goal-path-name>@<scenario-name>/

The following example shows the directory structure generated for the `S1` scenario created from the initial_rtl/lint/structure goal:

Project-1/consolidated_report/initial_rtl_lint_structure@S1/

Directory Structure Generated when a Top-Level Module is Specified

If you specify a top-level module, Atrenta Console generates reports for each scenario run under the following directory structure:

<project-working-directory>/<project-name>/<top-module>/<goal-path-name>/<scenario-name>/spyglass_reports/

NOTE: *You can specify a top-level module in the Top Level Design Unit field under the Set Read Options tab or by using the top project file command.*

Path in the Consolidated Directory Structure

The following directory structure contains links to the reports:

<project-working-directory>/<project-name>/consolidated_report/<top-module>_<goal-path-name>@<scenario-name>/

The following example shows the directory structure generated for the `S1` scenario created from the initial_rtl/cdc_exhaustive/cdc_verif_strict goal when you specify the top-level module as `moduleA`:

case1/Project-1/consolidated_report/moduleA_initial_rtl_lint_structure@S1/

Grouping in Standard SpyGlass Reports

The following standard SpyGlass reports display messages grouped into useful groups:

- *simple report*
- *moresimple report*

Grouping in Batch Reports

In batch reports, SpyGlass groups rule messages in the following order:

1. New, pre-existing, and fixed/missing messages

   This grouping is enabled in the incremental mode.

NOTE: *No further grouping occurs within the fixed/missing group.*

2. Built-in messages as compare to non built-in messages

   This grouping criterion is always enabled.

3. Rule goals

   This grouping criterion is enabled when you run a goal.

Grouping in GUI Reports

In GUI reports, SpyGlass groups rule messages in the following order:

1. New, pre-existing, and fixed/missing messages

NOTE: *No further grouping occurs within the fixed/missing group.*

2. Built-in messages as compare to non built-in messages
3. User-specified tags

   For details on adding tag to a message, see *Tag*.

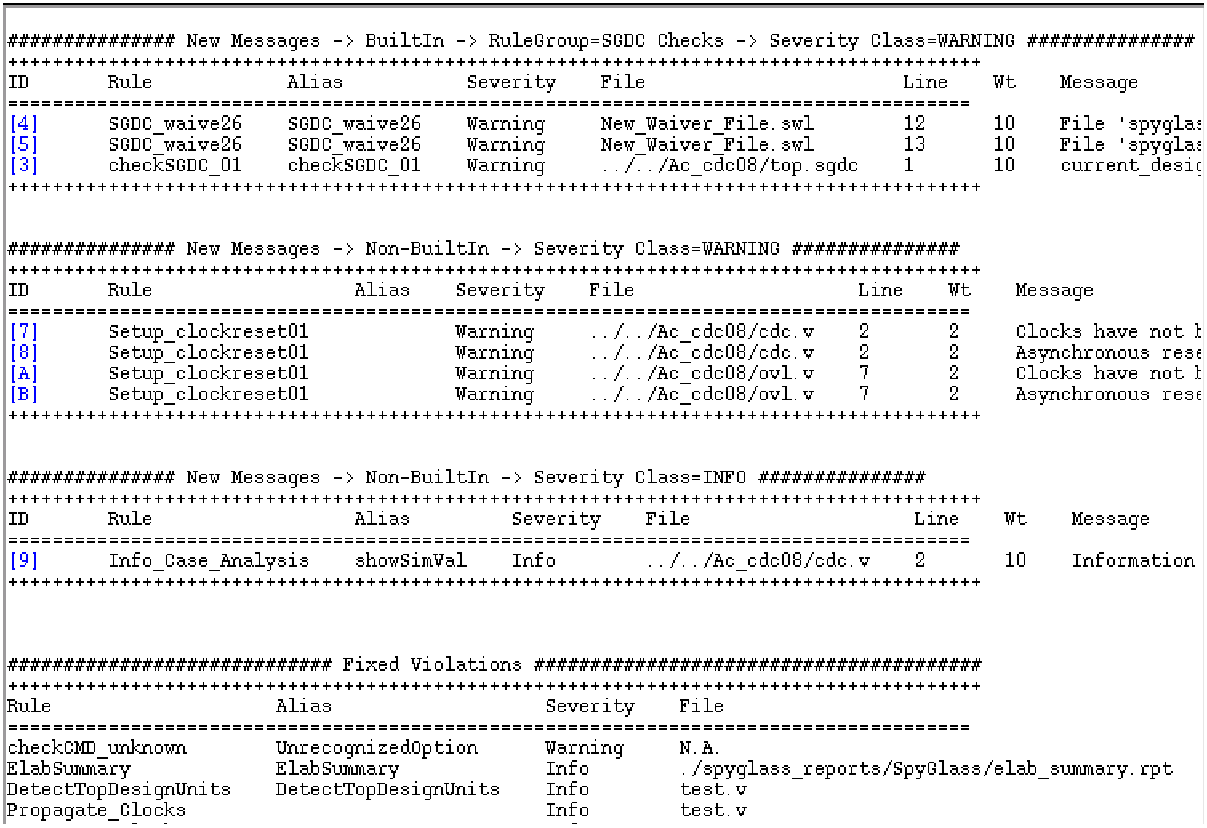
4. Rule goals

NOTE: *The grouping order of the rule messages in GUI reports is the same as the grouping order selected by you. Refer to the Message Grouping section for details about the configuration of the above grouping order.*

The generated reports contain different sections for leaf-level messages within each of the above grouping orders.

Sample Report

The following figure illustrates a sample moresimple report:



```
############### New Messages -> BuiltIn -> RuleGroup=SGDC Checks -> Severity Class=WARNING ###############
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID       Rule            Alias           Severity    File                        Line    Wt    Message
=======================================================================================
[4]      SGDC_waive26    SGDC_waive26    Warning     New_Waiver_File.swl         12      10    File 'spyglas
[5]      SGDC_waive26    SGDC_waive26    Warning     New_Waiver_File.swl         13      10    File 'spyglas
[3]      checkSGDC_01    checkSGDC_01    Warning     ../../Ac_cdc08/top.sgdc     1       10    current_desig
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


############### New Messages -> Non-BuiltIn -> Severity Class=WARNING ###############
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID       Rule                  Alias    Severity    File                     Line    Wt    Message
=======================================================================================
[7]      Setup_clockreset01             Warning     ../../Ac_cdc08/cdc.v     2       2     Clocks have not b
[8]      Setup_clockreset01             Warning     ../../Ac_cdc08/cdc.v     2       2     Asynchronous rese
[A]      Setup_clockreset01             Warning     ../../Ac_cdc08/ovl.v     7       2     Clocks have not b
[B]      Setup_clockreset01             Warning     ../../Ac_cdc08/ovl.v     7       2     Asynchronous rese
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


############### New Messages -> Non-BuiltIn -> Severity Class=INFO ###############
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID       Rule                  Alias        Severity    File                     Line    Wt    Message
=======================================================================================
[9]      Info_Case_Analysis    showSimVal    Info        ../../Ac_cdc08/cdc.v     2       10    Information
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



############################# Fixed Violations ########################################
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Rule                    Alias                   Severity    File
=======================================================================================
checkCMD_unknown        UnrecognizedOption      Warning     N.A.
ElabSummary             ElabSummary             Info        ./spyglass_reports/SpyGlass/elab_summary.rpt
DetectTopDesignUnits    DetectTopDesignUnits    Info        test.v
Propagate_Clocks                                Info        test.v
```

FIGURE 2.

Sorting Messages in SpyGlass Reports

SpyGlass generates the following reports that are sorted for better usability:

count report	*moresimple report*	*simple report*
summary report	*Default Reports*	

SpyGlass sorts the messages in these reports and the Message Tree by the following criteria (provided the criterion is applicable to the report):

- Severity Class (decreasing from FATAL, Error, Warning, and Info)
- Rule Name (alphabetical)
- Source HDL Filename (alphabetical)
- Line number (ascending)

You can modify the default sorting order of rule messages in BATCH reports and Message Tree. To display the sorting criteria with the severity label and rule weight, you can specify the following command in the project file:

```
set_option report_style sort_sevlabel_wt
```

When this option is specified, SpyGlass sorts the messages in reports and Message Tree by the following criteria:

- Severity Label (alphabetical)
- Rule Weight (descending)
- Severity Class (decreasing from FATAL, Error, Warning, Info)
- Rule Name (alphabetical)
- Source HDL filename (alphabetical)
- Line number (ascending)

User-Defined Message Sorting in SpyGlass Reports

SpyGlass provides a predefined sorting order for messages in SpyGlass reports. You can customize this sorting order as per your requirement.

Default Sorting Order

By default, messages are sorted in the following order:

1. Sorting based on severity
2. Sorting based on rule name
3. Sorting based on file and line number for a given rule

Customizing the Sorting Order

You can specify your own message sorting order based on message components by using the *sortrule* command.

Following is the syntax of the *sortrule* command:

```
set_option sortrule <language>+<rule-name>+<sort-order>
```

NOTE: *There are no spaces between any of the values in the above syntax.*

Details of Arguments of the sortrule Command

The following table describes the details of various arguments of the *sortrule* command:

Argument	Description
<language>	Specifies the rule language. You can specify the language as Verilog, VHDL, or Mixed. The rule for which message sorting order is being defined must be registered for the specified language. If the rule is specified for both languages, you can optionally specify only one of the languages if you want to specify the message sorting order for only that language.
<rule-name>	Specifies the rule name
<sort-order>	Specifies a user-defined sort order in the following format: <arg-number><arg-type><arg-sort-order>
Arguments of <*sort-order*>	

Argument	Description
<arg-number>	Specifies the argument number. To get the argument number, refer the rule message goal in the Product rule-deck file. For example, the LPFSM16 rule of the SpyGlass Power Verify solution has the following message goal: Attribute '%1' found on enumerated type '%2' used for encoding FSM states Thus, the first argument is the attribute name and is specified as 1. The second argument is the state variable name and is specified as 2. Refer the corresponding rules reference document for explanation of the rule message arguments.
<arg-type>	Specifies the argument type. In the above example, the argument type for both arguments is string and is specified as s. The other possible argument types are numerals (specified as n) and enumerated types (specified as e)
<arg-sort-order>	Specifies the argument value sorting order as ascending (specified as a) or descending (specified as d).

Example of Using the sortrule Option

Consider the following example:

```
set_option sortrule Verilog+R1+2sa+1nd+3e/val1/val2/val3
```

The above specification defines message sorting order of the R1 rule in Verilog mode. Further, the messages are sorted in the following order:

1. Messages sorted by 2sa, that is, sorted by the value of the second argument (2) which is a string argument (s) in ascending order (a).
2. For messages with the same second argument value, sorting is done by 1nd, that is, sort by the first argument (1) which is a numeral argument (n) in descending order (d).
3. For messages with the same first argument value, sorting is done by 3e/val1/val2/val3, that is, sort by the third argument (3) which is an enumerated type argument (e) based argument values val1, val2, and val3 in that order.

In addition to the argument-based sorting orders described above, you can specify message sorting order by file (specified as f) and by line number (specified as l), both in either ascending order (specified as a) or descending order (specified as d). Thus, fd means that sort the messages by file name in descending order. la means that sort by line number in ascending order.

Consider the sortrule specification in the above example with addition values as follows:

```
set_option sortrule Verilog+R1+2sa+1nd+3e/val1/val2/
val3+fd+la
```

This specification means that any sorting after the argument-based sorting will be done first by file names in descending order and then by line numbers in ascending order.

By default, the argument values are sorted in a case-sensitive manner. Specify i (for ignore case) to indicate that the argument values are to be sorted in a case-insensitive manner. Consider the following example:

```
set_option sortrule Verilog+R1+2sai+1nd+3e/val1/val2/
val3+fdi+la
```

The above specification indicates that argument-based sorting indicated by 2sa and file-based sorting indicated by fd should be performed in a case-insensitive manner.

Appendix

Supported Options of the set_option and/or set_goal_option Commands

Following table contains various options supported by the `set_option` and/or `set_goal_option` command:

Option Name	**Supported by set_option command?**	**Supported by set_goal_option command?**
abstract_file_name_style	Yes	No
abstract_searchpath	Yes	No
addrules	No	Yes
aggregate_report	Yes	No
aggregate_reportdir	Yes	No
aggregate_report_config_file	Yes	No
allow_celldefine_as_top	Yes	No
allow_fatal_downgrade	Yes	No
allow_recursion_limit	Yes	No

Option Name	Supported by set_option command?	Supported by set_goal_option command?
allow_module_override	Yes	No
block_abstract_directory	Yes	Yes
cachedir	Yes	No
cell_library	Yes	Yes
check_celldefine	Yes	Yes
checkdu	Yes	No
checkip	Yes	No
consolidate_reportdir	Yes	No
convert_udp_to_latch	Yes	No
debug_comments	No	Yes
DEBUG	Yes	Yes
default_waiver_file	Yes	Yes
define	Yes	No
define_incr_dirmap	Yes	Yes
define_severity	No	Yes
designread_enable_synthesis	Yes	No
designread_synthesis_mode	Yes	No
disable_encrypted_hdl_checks	Yes	Yes
disable_hdllibdu_lexical_checks	Yes	Yes
disable_hdlin_translate_off_skip_text	Yes	No
disable_hdlin_synthesis_off_skip_text	Yes	No
disable_html_report	Yes	No
disable_report	No	Yes
disallow_view_delete	Yes	No
dump_all_modes	Yes	No
dump_precompile_builtin	Yes	Yes
dw	Yes	No
elab_precompile	Yes	No

Option Name	**Supported by set_option command?**	**Supported by set_goal_option command?**
enable_abstract_blocks_schematic	Yes	Yes
enable_const_prop_thru_seq	Yes	Yes
enable_module_based_reporting	Yes	No
enable_pass_exit_codes	Yes	Yes
enable_physical_aware_pe	Yes	No
enableSV	Yes	No
enable SVA	Yes	No
enable_gateslib_autocompile	Yes	No
enable_hdl_encryption	Yes	No
enable_inactive_rtl_checks	Yes	Yes
enable_pgnetlist	Yes	No
enable_precompile_vlog	Yes	No
enable_save_restore	Yes	No
enable_save_restore_builtin	Yes	Yes
enable_sglib_debug	Yes	Yes
force_compile	Yes	No
force_gateslib_autocompile	Yes	No
gen_blk_sgdc	Yes	Yes
gen_hiersgdc	Yes	No
handlememory	Yes	No
hdlin_synthesis_off_skip_text	Yes	No
hdlin_translate_off_skip_text	Yes	Yes
hdllibdu	Yes	Yes
higher_capacity	Yes	No
ignoredu	Yes	No
ignorefile	Yes	No
ignorelibs	Yes	Yes
ignore_reference_project_sgdc	No	Yes
ignorerules	Yes	Yes

Option Name	Supported by set_option command?	Supported by set_goal_option command?
ignorewaivers	Yes	Yes
ignore_undefined_rules	No	Yes
incdir	Yes	No
inferblackbox	Yes	No
inferblackbox_iterations	Yes	No
lang	Yes	No
language_mode	Yes	No
lib	Yes	No
libmap	Yes	No
libext	Yes	No
libhdlf	Yes	No
libhdlfiles	Yes	No
LICENSEDEBUG	Yes	No
lvpr	Yes	Yes
mapSuffixList	Yes	Yes
mapPrefixList	Yes	Yes
mapVirtualClkByName	Yes	Yes
mthresh	Yes	No
net_osc_count_limit	Yes	No
nobb	Yes	No
nodefparam	Yes	No
noelab	Yes	No
nopreserve	Yes	No
noreport	Yes	Yes
norules	No	Yes
nosavepolicy/nosavepolicies	Yes	Yes
old_vdbfile	No	Yes
operating_mode	Yes	Yes
overload	No	Yes
overloadpolicies	No	Yes

Option Name	Supported by set_option command?	Supported by set_goal_option command?
overloadpolicy	No	Yes
overloadrules	Yes	Yes
ovl_verilog	Yes	No
ovl_vhdl	Yes	No
param	Yes	No
perflog	Yes	Yes
physical_dbdir	Yes	No
portparam	Yes	No
pragma	Yes	No
prefer_tech_lib	Yes	No
preserve_mux	Yes	Yes
print_sortoder_only	Yes	No
projectcwd	Yes	No
projectwdir	Yes	No
project_read_only	Yes	No
reference_design_projectfile	No	Yes
reference_design_sgdc	No	Yes
reference_design_sources	No	Yes
relax_hdl_parsing	Yes	No
remove_work	Yes	No
report	Yes	Yes
report_incr_messages	Yes	Yes
report_inst_backref	Yes	No
report_ip_waiver	Yes	Yes
report_max_size	Yes	Yes
report_style	Yes	Yes
reportfile	Yes	Yes
resetall	Yes	No
rules	No	Yes
savepolicies	Yes	No

Option Name	Supported by set_option command?	Supported by set_goal_option command?
savepolicy	Yes	No
sdc2sgdc	Yes	Yes
sdc2sgdc_mode	Yes	Yes
sdc2sgdcfile	Yes	Yes
scfu	Yes	No
sgdc_check_severity	Yes	No
sgsyn_clock_gating	Yes	Yes
sgsyn_clock_gating_threshold	Yes	Yes
sgsyn_loop_limit	Yes	No
show_lib	Yes	No
skip_rules_for_fast_restore	Yes	No
sort	Yes	No
sortrule	Yes	Yes
stop	Yes	No
stopdir	Yes	No
stopfile	Yes	No
support_sdc_style_escaped_name	Yes	Yes
target	Yes	Yes
top	Yes	No
use_du_sch_hier	Yes	Yes
use_goal_rule_sort	Yes	Yes
use_scan_flops	Yes	Yes
v	Yes	No
validate_hiersgdc	Yes	No
verbosity	Yes	No
w	Yes	Yes
work	Yes	No
y	Yes	No

Option Name	Supported by set_option command?	Supported by set_goal_option command?
vlog2001_generate_name	Yes	No
vlog2005_lrm_naming	Yes	No

The following table lists the details of all the options of the `set_option` and `set_goal_option` commands:

Option Name	Details
87	Description Checks the design for IEEE standard 1076-1987 compliance.
	Syntax `set_option 87 <yes	no>`
	Example `set_option 87 yes`
abstract_file_name_style	Description Controls the naming style of the abstract files generated by SpyGlass.
	Syntax `set_option abstract_file_name_style <compat	short>`
	Example `set_option abstract_file_name_style short`
abstract_searchpath	Description Enables you to locate block SGDC files during abstract view use run in the SoC flow.
	Syntax `set_option abstract_searchpath { <dir_path1> <dir_path2> }`
	Example `set_option abstract_searchpath {./block1_sgdc/}`

Option Name	Details
active_methodology	Description Specifies the active methodology when a project is being loaded. This option tracks the last methodology which user selected before closing the project, and restores it back when the project is loaded again.
	Syntax `set_option active_methodology <methodology-name>`
	Example `set_option active_methodology M1`
addrules	Description Selectively adds a rule in current goal.
	Syntax `set_goal_option addrules <rule-names>`
	Example `set_goal_option addrules myRule23`
aggregate_report	Description Specifies the type of aggregate report to be generated.
	Syntax `set_option aggregate_report { <report-type-list> }`

	Example
	`set_option aggregate_report dashboard`
	`set_option aggregate_report { project_summary dashboard }`

aggregate_reportdir	Description Specifies the directory path where data for aggregate report should be generated.
	Syntax `set_option aggregate_reportdir <report-dir-path>`
	Example `set_option aggregate_reportdir default_dir`

Option Name	Details
aggregate_report_config_file	Description Specifies a configuration file path containing project/dump directory names where data for aggregate report is present.
	Syntax `set_option aggregate_report_config_file <config-file-path>`
	Example `set_option aggregate_report_config_file config_file`
allow_celldefine_as_top	Description Specifies to perform rule-checking on 'celldefine module top's hierarchy.
	Syntax `set_option allow_celldefine_as_top <yes | no>`
	Example `set_option allow_celldefine_as_top yes`
allow_fatal_downgrade	Description Cancels overloading of the SGDC rules, which are not built-in and have a FATAL severity. This option is available for backward compatibility only.
	Syntax `set_option allow_fatal_downgrade <yes | no | 1 | 0 >`
	Example `set_option allow_fatal_downgrade no`
allow_module_override	Description Allows duplicate module/UDP definitions.
	Syntax `set_option allow_module_override <yes | no>`
	Example `set_option allow_module_override yes`
allow_recursion_limit	Description Specifies recursion limit for a VHDL function.
	Syntax `set_option allow_recursion_limit <recursion_limit>`
	Example `set_option allow_recursion_limit 1095`

Option Name	Details
block_abstract_directory	Description Specifies a directory in which the abstract view of a block should be saved.

	Syntax
	`set_option block_abstract_directory <directory>`
	`set_goal_option block_abstract_directory <directory>`

	Example `set_option block_abstract_directory dir1`
cachedir	Description Specifies the directory where library compilation will be performed
	Syntax `set_option cachedir <cachedir-name>`
	Example `set_option cachedir /myDir1/cache`
cell_library	Description Skip rule-checking on design units loaded from precompiled libraries

	Syntax
	`set_option cell_library <library-name>`
	`set_goal_option cell_library <library-name>`

	Example `set_option cell_library { "L2" "L4" }`
check_celldefine	Description Turns on the rule-checking on all the 'celldefine modules

	Syntax
	`set_option check_celldefine <yes | no>`
	`set_goal_option check_celldefine <yes | no>`

	Example `set_option check_celldefine yes`
checkdu	Description Specifies the design hierarchy (level) for rule-checking.
	Syntax `set_option checkdu <du-name>`
	Example `set_option checkdu modA`

Option Name	**Details**
checkip	Description Specifies the design units for rule-checking.
	Syntax `set_option checkip <ip-name>`
	Example `set_option checkip mod`
consolidate_reportdir	Description Specifies the directory path containing reports and data in a consolidated user-friendly directory structure.
	Syntax `set_option consolidate_reportdir <directory-path>`
	Example `set_option consolidate_reportdir /abc/reports/`
convert_udp_to_latch	Description Enables SpyGlass to infer UDP as a latch while translating the UDP with both edge and level sensitiveness. By default, SpyGlass infers a UDP as a flip-flop in such cases.
	Syntax `set_option convert_udp_to_latch <yes | no>`
	Example `set_option convert_udp_to_latch yes`
DEBUG	Description Decompiles debug information about various stages during a SpyGlass run.

	Syntax
	`set_option DEBUG <stage>`
	`set_goal_option DEBUG <stage>`
	Where, *`<stage>`* can be `default`, `analysis`, `elaborator`, sgom, and sgdc.

	Example `set_option DEBUG { "elaborator" "sgom" "sgdc" }` The above example generates debug information for elaborator, sgom, and sgdc.

Option Name	**Details**
debug_comments	Description Specifies the debug comments to identify a goal/scenario in the correlation view
	Syntax `set_option debug_comments {"<text>"}`
	Example `set_option debug_comments {"Goal to test"}`
default_waiver_file	Description Specifies a default waiver file for saving interactive waiver commands. If you do not specify a default waiver file by using this command, Atrenta Console considers the *<project-wdir>/<project_name>.swl* file as the default waiver file.
	Syntax `set_option default_waiver_file <file-name>`
	Example `set_option default_waiver_file A.swl`
define	Description Adds the specified macro definitions.
	Syntax `set_option define <macro-def>=<value>`
	Example The following command sets the value of the `State0` macro to 3: set_option define "State0=3" The following command sets the value of macros, `State0` and `State1`, to 3 and 5, respectively: `set_option define { "State0=3" "State1=5" }`
define_incr_dirmap	Description Provides mapping for different locations of RTL files.

	Syntax
	`set_option define_incr_dirmap <dir1> <dir2>`
	`set_goal_option define_incr_dirmap <dir1> <dir2>`

	Example `set_option define_incr_dirmap { "case01/v_1" "case01/v_2" }`

Option Name	Details
define_severity	Description Defines a user-specified severity label.
	Syntax `set_goal_option define_severity <product-name>+<severity-label>+<severity-class>`
	Example `set_goal_option define_severity lint+myFATAL+FATAL`
designread_enable_synthesis	Description Enables synthesis during the design read process.
	Syntax `set_option designread_enable_synthesis <yes | no>`
	Example `set_option designread_enable_synthesis yes`
designread_synthesis_mode	Description Specifies the type of synthesis to be performed during the design read process.
	Syntax `set_option designread_synthesis_mode <mode>` Where, *<mode>* can be `base`, `opt`, or `techmap`.
	Example `set_option designread_synthesis_mode opt`
disable_encrypted_hdl_checks	Description Disables RTL rule-checking on the encrypted design units.

	Syntax
	`set_option disable_encrypted_hdl_checks <yes|no>`
	`set_goal_option disable_encrypted_hdl_checks <yes|no>`

	Example `set_option disable_encrypted_hdl_checks yes`
disable_hdllibdu_lexical_checks	Description Disallows lexical rule checking on precompiled libraries.

	Syntax
	`set_option disable_hdllibdu_lexical_checks <yes|no>`
	`set_goal_option disable_hdllibdu_lexical_checks <yes|no>`

	Example `set_option disable_hdllibdu_lexical_checks yes`

Option Name	Details
disable_hdlin_translate_off_skip_text	Description Disables interpretation of VHDL design code between Synopsys translate_off/translate_on pragma pair as comments.
	Syntax `set_option disable_hdlin_translate_off_skip_text <yes|no>`
	Example `set_option disable_hdlin_translate_off_skip_text yes`
disable_hdlin_synthesis_off_skip_text	Description Disables interpretation of VHDL design code between Synopsys synthesis_off/synthesis_on pragma pair as comments.
	Syntax `set_option disable_hdlin_synthesis_off_skip_text <yes|no>`
	Example `set_option disable_hdlin_synthesis_off_skip_text yes`
disable_html_report	Description Disables automatic generation of the specified reports. You can specify more than one options as a space-separated list
	Syntax `set_option disable_html_report <dashboard|datasheet|html>`
	Example `set_option disable_html_report {datasheet}`
disable_report	Description Disables the generation of the specified reports that are generated by default.
	Syntax `set_goal_option disable_report { <report-names> }`
	Example `set_goal_option disable_report {R1 R2}`

Option Name	Details
disallow_view_delete	Description Disables the Large Design Processing Mode.
	Syntax `set_option disallow_view_delete <yes | no>`
	Example `set_option disallow_view_delete yes`
dump_all_modes	Description Enable both 32-bit and 64-bit precompiled HDL sources creation
	Syntax `set_option dump_all_modes <yes | no>`
	Example `set_option dump_all_modes yes`
dump_precompile_builtin	Description Enables reporting of same built-in message while using precompiled RTL dump that were reported when that dump was being generated.

	Syntax
	`set_option dump_precompile_builtin <yes |no>`
	`set_goal_option dump_precompile_builtin <yes |no>`

	Example `set_option dump_precompile_builtin yes`
dw	Description Enables DesignWare component support
	Syntax `set_option dw <yes | no>`
	Example `set_option dw yes`
elab_precompile	Description Enables elaboration in single step precompilation
	Syntax `set_option elab_precompile <yes | no>`
	Example `set_option elab_precompile yes`

Option Name	Details
enable_abstract_blocks_schematic	Description Enables schematic debugging of abstracted modules.
	Syntax `set_option enable_abstract_blocks_schematic <abstract_module_names>`
	Example `set_option enable_abstract_blocks_schematic { mts_* }`
enable_const_prop_thru_seq	Description Allows constant propagation beyond sequential elements during logic simulation.

	Syntax
	`set_option enable_const_prop_thru_seq <yes | no>`
	`set_goal_option enable_const_prop_thru_seq <yes|no>`

	Example
	`set_option enable_const_prop_thru_seq yes`
	`set_goal_option enable_const_prop_thru_seq yes`

enable_module_based_reporting	Description Enables module name based reporting in SpyGlass reports.
	Syntax `set_option enable_module_based_reporting <yes | no>`
	Example `set_option enable_module_based_reporting yes`
enable_pass_exit_codes	Description Provides exit code as 0, 11, and 12 for runs with INFO, WARNING, and ERROR, respectively. If this option is turned off, the exit code is 0 in all the above cases. For more details, see *SpyGlass Exit Status*.

	Syntax
	`set_option enable_pass_exit_codes <yes |no>`
	`set_goal_option enable_pass_exit_codes <yes |no>`

	Example `set_option enable_pass_exit_codes yes`

Option Name	Details
enable_physical_aware_pe	Description Enables Physical aware power estimation flow and checks out required licenses
	Syntax `set_option enable_physical_aware_pe <yes | no>`
	Example `set_option enable_physical_aware_pe yes`
enableSV	Description Enables parsing of SystemVerilog constructs
	Syntax `set_option enableSV <yes | no>`
	Example `set_option enableSV yes`
enable SVA	Description Enables parsing of SystemVerilog constructs and also handles the SV Assert logic.
	Syntax `set_option enableSVA <1 | 0>`
	Example `set_option enableSVA 1`
enable_gateslib_autocompile	Description Enables automatic compilation of Synopsys Liberty™ files (.lib files) to a SpyGlass-compatible library file form (.sglib files).
	Syntax `set_option enable_gateslib_autocompile <yes | no>`
	Example set_option enable_gateslib_autocompile yes
enable_hdl_encryption	Description Enables encryption of VHDL/Verilog libraries during compilation.
	Syntax `set_option enable_hdl_encryption <yes | no>`
	Example `set_option enable_hdl_encryption yes`

<table>
<tr><th>Option Name</th><th>Details</th></tr>
<tr><td rowspan="3"><em>enable_inactive_rtl_checks</em></td><td>Description
Enables semantic checking capability in SpyGlass</td></tr>
<tr><td>Syntax
<code>set_option enable_inactive_rtl_checks <yes|no></code>
<code>set_goal_option enable_inactive_rtl_checks <yes|no></code></td></tr>
<tr><td>Example
<code>set_option enable_inactive_rtl_checks yes</code></td></tr>
<tr><td rowspan="3"><em>enable_pgnetlist</em></td><td>Description
Enables power and ground pin information to be taken into consideration from specified physical libraries.</td></tr>
<tr><td>Syntax
<code>set_option enable_pgnetlist <yes |no></code></td></tr>
<tr><td>Example
<code>set_option enable_pgnetlist yes</code></td></tr>
<tr><td rowspan="3"><em>enable_precompile_vlog</em></td><td>Description
Enables the Precompiled Verilog library feature.</td></tr>
<tr><td>Syntax
<code>set_option enable_precompile_vlog <yes | no></code></td></tr>
<tr><td>Example
<code>set_option enable_precompile_vlog yes</code></td></tr>
<tr><td rowspan="3"><em>enable_save_restore</em></td><td>Description
Enables the Design Save/Restore feature.</td></tr>
<tr><td>Syntax
<code>set_option enable_save_restore <yes | no></code></td></tr>
<tr><td>Example
<code>set_option enable_save_restore yes</code></td></tr>
<tr><td rowspan="3"><em>enable_save_restore_builtin</em></td><td>Description
Enables reporting of same built-in messages during the restore run that were reported during the save run.</td></tr>
<tr><td>Syntax
<code>set_option enable_save_restore_builtin <yes | no></code>
<code>set_goal_option enable_save_restore_builtin <yes | no></code></td></tr>
<tr><td>Example
<code>set_option enable_save_restore_builtin yes</code></td></tr>
</table>

Option Name	**Details**
enable_sglib_debug	Description Provides SpyGlass Library Compiler debug Information

	Syntax
	`set_option enable_sglib_debug <yes | no>`
	`set_goal_option enable_sglib_debug <yes | no>`

	Example `set_option enable_sglib_debug yes`
force_compile	Description Forces compilation of files.
	Syntax `set_option force_compile <yes | no>`
	Example `set_option force_compile yes`
force_gateslib_autocompile	Description Forces automatic compilation of Synopsys Liberty™ files (.lib files) to a SpyGlass-compatible library file form (.sglib files).
	Syntax `set_option force_gateslib_autocompile <yes|no>`
	Example `set_option force_gateslib_autocompile yes`
gen_blk_sgdc	Description Enables generation of an abstract view of a block.

	Syntax
	`set_option gen_blk_sgdc <yes | no>`
	`set_goal_option gen_blk_sgdc <yes | no>`

	Example `set_option gen_blk_sgdc yes`
gen_hiersgdc	Description Enables hierarchical migration of SGDC commands from block to chip level.
	Syntax `set_option gen_hiersgdc <yes|no>`
	Example `set_option gen_hiersgdc yes`

Option Name	Details
handlememory	Description Specifies to process memories in an optimized manner.
	Syntax `set_option handlememory <yes | no>`
	Example `set_option handlememory yes`
hdlin_synthesis_off_skip_text	Description Interpret VHDL design code between Synopsys synthesis_off/ synthesis_on pragma pairs as comments.
	Syntax `set_option hdlin_synthesis_off_skip_text <yes|no>`
	Example `set_option hdlin_synthesis_off_skip_text yes`
hdlin_translate_off_skip_text	Description Interpret VHDL design code between Synopsys translate_off/ translate_on pragma pair as comments.

	Syntax
	`set_option hdlin_translate_off_skip_text <yes|no>`
	`set_goal_option hdlin_translate_off_skip_text <yes|no>`

	Example `set_option hdlin_translate_off_skip_text yes`
hdllibdu	Description Enables RTL and lexical rule-checking on precompiled Verilog/ VHDL design units.

	Syntax
	`set_option hdllibdu <yes | no>`
	`set_goal_option hdllibdu <yes | no>`

	Example `set_option hdllibdu yes`
higher_capacity	Description Disables rules designed for SpyGlass version 3.2.0.
	Syntax `set_option higher_capacity <yes | no>`
	Example `set_option higher_capacity yes`

Option Name	Details
ignoredu	Description Ignores specific VHDL design units or Verilog modules at the design read (parsing) stage.
	Syntax `set_option ignoredu { <design-unit-names> }`
	Example `set_option ignoredu {block1 block2}`
ignorefile	Description Ignores a specific file from SpyGlass analysis
	Syntax `set_option ignorefile <file-name>`
	Example `set_option ignorefile myaddlfile.v`
ignorelibs	Description Skips the rule-checking for modules in the library files specified through v/y option

	Syntax
	`set_option ignorelibs <yes | no>`
	`set_goal_option ignorelibs <yes | no>`

	Example `set_option ignorelibs yes`
ignore_reference_project_sgdc	Description Ignores SGDC files specified in the project file specified by the *reference_design_projectfile* command.
	Syntax `set_goal_option ignore_reference_project_sgdc <yes | no>`
	Example `set_goal_option ignore_reference_project_sgdc yes`

Option Name	Details
ignorerules	Description Specifies the rule names or rule group names to be ignored in rule checking

	Syntax
	`set_option ignorerules { rule-names }`
	`set_goal_option ignorerules { rule-names }`

	Example `set_option ignorerules R1`
ignorewaivers	Description Causes SpyGlass to ignore waivers supplied as embedded SpyGlass Waiver pragmas

	Syntax
	`set_option ignorewaivers <yes | no>`
	`set_goal_option ignorewaivers <yes | no>`

	Example set_option ignorewaivers yes
ignore_undefined_rules	Description Continue after issuing a warning message if an undefined rule is specified
	Syntax `set_goal_option ignore_undefined_rules <yes | no>`
	Example `set_goal_option ignore_undefined_rules yes`
incdir	Description Searches the specified path for include files.
	Syntax `set_option incdir <directory-path>`

	Example
	`set_option incdir "abc bcd xyz"`
	`set_option incdir /u/<user_name>/global`

Option Name	Details
inferblackbox	Description Infers black box module interface based on the black box instances in the synthesized netlist.
	Syntax `set_option inferblackbox <yes | no>`
	Example `set_option inferblackbox yes`
inferblackbox_iterations	Description Specifies the effort in terms of the total number of iterations that SpyGlass should make before finalizing port directions for black boxes.
	Syntax `set_option inferblackbox_iterations <int-value>`
	Example `set_option inferblackbox_iterations 2`
lang	Description Specifies the display language for messages and waivers.
	Syntax `set_option lang <language>`
	Example `set_option lang ja`
language_mode	Description Specifies language specification to use.
	Syntax `set_option language_mode <value>` Where, *`<value>`* can be `verilog`, `vhdl`, `mixed`, `def`, or `lef`.
	Example `set_option language_mode verilog`
lib	Description Defines the logical to physical mapping for referenced libraries.
	Syntax `set_option lib <logical-lib-name> <physical-path>`
	Example `set_option lib MATH ../lib/MATH`

Option Name	Details
libmap	Description Specifies mapping between logical libraries and an intermediate library
	Syntax `set_option libmap <logical-lib-name> <intermediate-lib-name>`
	Example `set_option libmap L1 IP1`
LICENSEDEBUG	Description Prints license debug information on the screen.
	Syntax `set_option LICENSEDEBUG <yes | no>`
	Example `set_option LICENSEDEBUG yes`
libext	Description Specifies library file extensions.
	Syntax `set_option libext {space-separated list of extensions }`
	Example `set_option libext .v`
libhdlf	Description Specifies mapping between specified logical library and source-files for precompiled library use.
	Syntax `set_option libhdlf <logical-library-name> {space separated source file list}`
	Example `set_option libhdlf L1 {S1.f S2.f}`

Option Name	Details
libhdlfiles	Description Specify mapping between specified logical library and HDL files for precompiled library use.
	Syntax set_option libhdlfiles <logical_library_name> { file-list }

	Example
	set_option libhdlfiles L1 {rtl_1.v rtl_2.v}
	set_option libhdlfiles L3 {rtl_dir/*}

lvpr	Description Specifies the maximum number of messages to be reported per rule.
	Syntax set_option lvpr <rule-name>=<num> set_option lvpr <num> set_goal_option lvpr <rule-name>=<num> set_goal_option lvpr <num>
	Example set_option lvpr rule1=20 set_goal_option lvpr 20
mapSuffixList	Description Specifies a comma-separated list of suffix strings used for mapping virtual clocks to real clocks.
	Syntax set_option mapSuffixList <list> set_goal_option mapSuffixList <list>
	Example set_option mapSuffixList virtual,virt,VIRT,io set_goal_option mapSuffixList virtual,virt
mapPrefixList	Description Specifies a comma-separated list of prefix strings used for mapping virtual clocks to real clocks.
	Syntax set_option mapPrefixList <list> set_goal_option mapPrefixList <list>
	Example set_option mapPrefixList virtual,virt,VIRT,io set_goal_option mapPrefixList virtual,virt

Option Name	Details
mapVirtualClkByName	Description Specifies a manner in which virtual-to-real clock mapping should be done.

	Syntax
	`set_option mapVirtualClkByName <yes | no>`
	`set_goal_option mapVirtualClkByName <yes | no>`

	Example
	`set_option mapVirtualClkByName yes`
	`set_goal_option mapVirtualClkByName yes`

mthresh	Description Specifies the bit-count threshold for the compilation of net/ variables in a design unit.
	Syntax `set_option mthresh <num>`
	Example `set_option mthresh 1024`
net_osc_count_limit	Description Specifies the number of oscillations allowed to get a stable value on a particular net within SpyGlass logic evaluator.
	Syntax `set_option net_osc_count_limit <num>`
	Example `set_option net_osc_count_limit 200`
nobb	Description Forces SpyGlass to exit without processing if any black box is found in the design.
	Syntax `set_option nobb <yes | no>`
	Example `set_option nobb yes`

Option Name	**Details**
nodefparam	Description Ignores explicit parameter re-definition given by the `defparam` Verilog construct
	Syntax `set_option nodefparam <yes | no>`
	Example `set_option nodefparam yes`
noelab	Description Exit after design analysis and without elaborating the design.
	Syntax `set_option noelab <yes | no>`
	Example `set_option noelab yes`
nopreserve	Description Forces SpyGlass to remove hanging/unconnected instances and nets.
	Syntax `set_option nopreserve <yes | no>`
	Example `set_option nopreserve yes`
noreport	Description Suppresses report generation.

	Syntax
	`set_option noreport <yes | no>`
	`set_goal_option noreport <yes | no>`

	Example `set_option noreport yes`
norules	Description Suppresses rule-checking.
	Syntax `set_goal_option norules <yes | no>`
	Example `set_goal_option norules yes`

Option Name	Details
nosavepolicy/nosavepolicies	Description Specifies the product/products that should not be saved during design save.

	Syntax
	`set_option nosavepolicy <product_name>`
	`set_option nosavepolicies <product-list>`

	Example `set_option nosavepolicy P1`
old_vdbfile	Description Specifies path of previous violation database file for consideration in incremental mode.
	Syntax `set_goal_option old_vdbfile <file>`
	Example `set_goal_option old_vdbfile file1.vdb`
operating_mode	Description Enables setting value of special variable SG_OPERATING_MODE for SGDC conditional compilation.

	Syntax
	`set_option operating_mode <values>`
	`set_goal_option operating_mode <values>`

	Example `set_option operating_mode "x y z x1 x2"`
overload	Description Runs the specified named overloads for all specified products.
	Syntax `set_goal_option overload <named-overload>`
	Example `set_goal_option overload 'CAD CAD1 METEOR'`

Option Name	Details
overloadpolicies	Description Runs the specified products with the overloaded components.
	Syntax `set_goal_option overloadpolicies <product-names>`

	Example
	`set_option overloadpolicies a,b`
	`set_option overloadpolicies none`
	`set_option overloadpolicies all`

overloadpolicy	Description Runs the specified products with the overloaded components.
	Syntax `set_goal_option overloadpolicy <product-name>`

	Example
	`set_option overloadpolicy a,b`
	`set_option overloadpolicy none`
	`set_option overloadpolicy all`

overloadrules	Description Overloads the severity or weight of a rule.

	Syntax
	`set_option overloadrules <rule-name>+<lang>+severity=<severity-label>+weight=<value>`
	`set_goal_option overloadrules <rule-name>+<lang>+severity=<severity-label>+weight=<value>`

	Example
	`set_goal_option overloadrules W402b+Verilog+severity=Warning`
	`set_goal_option overloadrules InstNameLength+Verilog+weight=10`
	set_goal_option overloadrules TA_01+severity=Info+weight=10

Option Name	Details
ovl_verilog	Description Specifies an Open Verification Library (OVL) file written in Verilog.
	Syntax `set_option ovl_verilog { OVL-file> }`
	Example `set_option ovl_verilog { attach_ovl_verilog.ovl }`
ovl_vhdl	Description Specifies an Open Verification Library (OVL) file written in VHDL.
	Syntax `set_option ovl_vhdl { OVL-file> }`
	Example `set_option ovl_vhdl { attach_ovl_vhdl.ovl }`
param	Description Sets the new user-specified values of the VHDL generics and Verilog parameters.
	Syntax `set_option param <list>`

	Example
	`set_option param "control.width=8"`
	`set_option param "block1.limit=4"`

perflog	Description Specifies to generate the SpyGlass performance log.

	Syntax
	`set_option perflog <yes | no>`
	`set_goal_option perflog <yes | no>`

	Example `set_option perflog yes`
physical_dbdir	Description Specifies the design database directory path for save and restore runs.
	Syntax `set_option physical_dbdir <direcory_path>`
	Example `set_option physical_dbdir /myDir1/abc`

Option Name	Details
portparam	Description Overrides simple type parameters of top module ports only.
	Syntax `set_option portparam { <list-of-space-separated-strings> }`
	Example `set_option portparam {"test.P1.W1=7" "top.P3.W1=1" "top.P2.W2=4'b0011" "top.P1.W0=3" "top.P1.W2[0]=1'b1" "top.P1.W3=4" "top.P1.W1=xxxx"}`
pragma	Description Specifies a prefix-string for synthesis directives.
	Syntax `set_option pragma <values>`

	Example
	`set_option pragma nopragma`
	`set_option pragma { "quickturn" "synopsys" }`

prefer_tech_lib	Description Sets higher preference for technology library definitions over HDL definitions while resolving master of instances.
	Syntax `set_option prefer_tech_lib <yes | no>`
	Example `set_option prefer_tech_lib yes`
preserve_mux	Description Enables SpyGlass to pick mux cells from the technology library.

	Syntax
	`set_option preserve_mux <yes | no>`
	`set_goal_option preserve_mux <yes | no>`

	Example `set_option preserve_mux yes`
print_sortoder_only	Description Prints the list of sorted VHDL files and exits
	Syntax `set_option print_sortorder_only <yes | no>`
	Example `set_option print_sortorder_only yes`

Option Name	Details
projectcwd	Description Specifies the directory in which a project is created.
	Syntax `set_option projectcwd <dir-name>`
	Example `set_option projectcwd prjCWD`
projectwdir	Description Specifies the output directory of a project file.
	Syntax `set_option projectwdir <dir-name>`
	Example `set_option projectwdir prjDir`
project_read_only	Description Makes the project file as read only.
	Syntax `set_option project_read_only <yes | no>`
	Example `set_option project_read_only yes`
relax_hdl_parsing	Description Performs relaxed VHDL semantic checking.
	Syntax `set_option relax_hdl_parsing <yes | no>`
	Example `set_option relax_hdl_parsing yes`
reference_design_projectfile	Description Specifies a project file containing details, such as design files, design options, and SGDC for the reference design while running a DDR (Dual Design Read) goal.
	Syntax `set_goal_option reference_design_projectfile <file-name>`
	Example `set_goal_option reference_design_projectfile ref.prj`

Option Name	Details
reference_design_sgdc	Description Specifies an SGDC file that contains constraints for reference design while running a DDR (Dual Design Read) goal.
	Syntax `set_goal_option reference_design_sgdc <file-name>`
	Example `set_goal_option reference_design_sgdc ref.sgdc`
reference_design_sources	Description Specifies design files and options for a reference design through `.f` file while running a DDR (Dual Design Read) goal.
	Syntax `set_goal_option reference_design_sources <file-name>`
	Example `set_goal_option reference_design_sources ref_sources.f`
remove_work	Description Deletes the contents of the WORK directory.
	Syntax `set_option remove_work <yes | no>`
	Example `set_option remove_work yes`
report	Description Specifies the report format type to be generated.

	Syntax
	`set_option report <report-name>`
	`set_goal_option report <report-name>`

	Example `set_option report { "summary" }`
report_inst_backref	Description Prints back-reference information, such as file name and line number containing definition for a design and its instances in the elab_summary.rpt report
	Syntax `set_option report_inst_backref <yes | no>`
	Example `set_option report_inst_backref yes`

Option Name	Details
report_ip_waiver	Description Modifies the way violations are displayed in the waiver report.

	Syntax
	`set_option report_ip_waiver <yes | no>`
	`set_goal_option report_ip_waiver <yes | no>`

	Example
	`set_option report_ip_waiver yes`
	`set_goal_option report_ip_waiver yes`

report_max_size	Description Specifies the maximum number of messages for sorted reports.

	Syntax
	`set_option report_max_size <value>`
	`set_goal_option report_max_size <value>`

	Example `set_option report_max_size 60000`
report_style	Description Enables customization of report format

	Syntax
	`Set_option report_style <value>`
	`set_goal_option report_style <value>\`
	
	Where, <value> can be flat, grouped, display_msgid, hide_msgid, display_rulegroup, or `hide_rulegroup`.

	Example `set_option report_style hide_msgid`
reportfile	Description Sets the name and location of the SpyGlass report file.

	Syntax
	`set_option reportfile <name>`
	`set_goal_option reportfile <name>`

	Example `set_option reportfile myoutput`

Option Name	Details
report_incr_messages	Description Enables incremental reporting of messages

	Syntax
	`set_option report_incr_messages <yes | no>`
	`set_goal_option report_incr_messages <yes | no>`

	Example `set_option report_incr_messages yes`
resetall	Description Resets the Verilog compiler directive default_nettype to language default which is wire.
	Syntax `set_option resetall <yes | no>`
	Example `set_option resetall yes`
rules	Description List of rules or rule group names for which rule checking should be done.
	Syntax `set_goal_option rules <rule-names>`
	Example `set_goal_option rules { "SigName" "InstName"}`
savepolicies	Description Specifies the product/products that run during design restore.
	Syntax `set_option savepolicies <products-list>`
	Example
savepolicy	Description Specifies the product/products that run during design restore.
	Syntax `set_option savepolicy <product-name>`
	Example `set_option savepolicy P1`

Option Name	Details
sdc2sgdc	Description Translates design attributes from SDC format to SGDC format.

	Syntax
	`set_option sdc2sgdc <yes | no>`
	`set_goal_option sdc2sgdc <yes | no>`

	Example `set_option sdc2sgdc yes`
sdc2sgdc_mode	Description Specifies the mode of the SDC file to be translated to SGDC.

	Syntax
	`set_option sdc2sgdc_mode <mode_name>`
	`set_goal_option sdc2sgdc_mode <mode_name>`

	Example `set_option sdc2sgdc_mode one`
sdc2sgdcfile	Description Specifies the file to save the output of SDC-to-SGDC translation.

	Syntax
	`set_option sdc2sgdcfile <file-name>`
	`set_goal_option sdc2sgdcfile <file-name>`

	Example `set_option sdc2sgdcfile F1`
scfu	Description Enables each file to be compiled as a separate compilation unit.
	Syntax `set_option sfcu <yes | no>`
	Example `set_option sfcu yes`
sgdc_check_severity	Description Modifies the severity and/or provides severity control for the *checkSGDC_existence* rule (built-in) and other non-built in rules.
	Syntax `set_option sgdc_check_severity <COMPAT | DEFAULT | ERROR | FATAL | WARNING | IGNORE>`
	Example `set_option sgdc_check_severity {"compat"}`

Option Name	Details
sgsyn_clock_gating	Description Specifies whether clock gating should be enabled or disabled.

	Syntax
	`set_option sgsyn_clock_gating <1 | 0>`
	`set_goal_option sgsyn_clock_gating <1 | 0>`

	Example `set_option sgsyn_clock_gating 1`
sgsyn_clock_gating_threshold	Description Specifies the threshold of the clock gating logic.

	Syntax
	`set_option sgsyn_clock_gating_threshold <num>`
	`set_goal_option sgsyn_clock_gating_threshold <num>`

	Example `set_option sgsyn_clock_gating_threshold 20`
sgsyn_loop_limit	Description Specifies a loop-rolling limit during design synthesis.
	Syntax `set_option sgsyn_loop_limit <value>`
	Example `set_option sgsyn_loop_limit 2000`
show_lib	Description Enables generation of messages for each library module.
	Syntax `set_option show_lib <yes | no>`
	Example `set_option show_lib yes`
skip_rules_for_fast_restore	Description Enables skip of design's re-parsing and/or re-synthesis during design restore.
	Syntax `set_option skip_rules_for_fast_restore <yes | no>`
	Example `set_option skip_rules_for_fast_restore yes`

Option Name	Details
sort	Description Sorts and prints the design files before analyzing.
	Syntax `set_option sort <yes | no>`
	Example `set_option sort yes`
sortrule	Description Specifies the sort order for messages in SpyGlass reports.

	Syntax
	`set_option sortrule <language>+<rule-name>+<sort-order>`
	`set_goal_option sortrule <language>+<rule-name>+<sort-order>`

	Example
	`set_option sortrule Verilog+R1+2sa+1nd+3e/val1/val2/val3`
	`set_option sortrule Verilog+R1+2sai+1nd+3e/val1/val2/val3+fdi+la`

stop	Description Skips rule-checking on the specified VHDL design unit or Verilog module.
	Syntax `set_option stop <du-name>`

	Example
	`set_option stop block1`
	`set_option stop 'block1 block2'`
	`set_option stop "lib1.*"`

stopdir	Description Skips rule-checking on all design units located in a specified directory.
	Syntax `set_option stopdir <directory-name>`
	Example `set_option stopdir myaddldir`

Option Name	**Details**
stopfile	Description Skips rule-checking on all design units described in a specified file.
	Syntax `set_option stopfile <file-name>`

	Example
	`set_option stopfile myaddlfile.v`
	`set_option stopfile "dir?/*"`

support_sdc_style_escaped_name	Description Enable Synopsys-style escaped names in SpyGlass Design Constraints files.

	Syntax
	`set_option support_sdc_style_escaped_name <yes | no>`
	`set_goal_option support_sdc_style_escaped_name <yes | no>`

	Example set_option support_sdc_style_escaped_name yes
target	Description Specifies libraries to be used for technology-mapping out of the specified .sglib libraries.

	Syntax
	`set_option target <lib-name-list>`
	`set_goal_option target <lib-name-list>`

	Example `set_option target tech1` **NOTE:** In the above example, tech1 is considered as the library name given in `tech1.sglib`.
top	Description Specifies top of the design.
	Syntax `set_option top <du-name>`
	Example `set_option top modA`

Option Name	Details
use_du_sch_hier	Description Enables SpyGlass to use schematic information to waive violations on a design unit.

	Syntax
	`set_option use_du_sch_hier <yes | no>`
	`set_goal_option use_du_sch_hier <yes | no>`

	Example `set_option use_du_sch_hier yes`
use_goal_rule_sort	Description Sorts violation messages in the moresimple report based on the order of rules specified in a goal file.

	Syntax
	`set_option use_goal_rule_sort <yes | no>`
	`set_goal_option use_goal_rule_sort <yes | no>`

	Example `set_option use_goal_rule_sort yes`
use_scan_flops	Description Enables SpyGlass to pick scan flops from the technology library.

	Syntax
	`set_option use_scan_flops <yes | no>`
	`set_goal_option use_scan_flops <yes | no>`

	Example `set_option use_scan_flops yes`
v	Description Specifies the library file used in the source design.
	Syntax `set_option v <file-names>`
	Example `set_option v /u/<user_name>/libfile.v`
validate_hiersgdc	Description Enables validation of hierarchically migrated block-level SGDC commands to the chip-level.
	Syntax `set_option validate_hiersgdc <yes | no>`
	Example `set_option validate_hiersgdc yes`

Option Name	Details
verbosity	Description Controls the verbosity of log file. You can control the extent of the information present in the spyglass.log file
	Syntax `set_option verbosity <0 | 1 | 2 | 3>`
	Example `set_option verbosity 2`
w	Description Turns on generation of warnings for PERL-level compilation.

	Syntax
	`set_option w <yes | no>`
	`set_goal_option w <yes | no>`

	Example `set_option w yes`
work	Description Specifies the logical library directory for compilation of Verilog/ VHDL libraries.
	Syntax `set_option work <work-dir-name>`
	Example `set_option lib work2 /u/<user_name>/my_work_lib`
y	Description Specifies the library directory containing libraries.
	Syntax `set_option y { space-separated path-names of directories }`
	Example `set_option y /u/<user_name>/libdir`
vlog2001_generate_name	Description Specifies if all Verilog generate statements in the Verilog 2001 syntax should be unrolled.
	Syntax `set_option vlog2001_generate_name <yes | no>`
	Example `set_option vlog2001_generate_name yes`

Option Name	Details
vlog2005_lrm_naming	Description Specifies if all Verilog generate statements in the Verilog 2005 syntax should be unrolled.
	Syntax `set_option vlog2005_lrm_naming <yes | no>`
	Example `set_option vlog2005_lrm_naming_name yes`

Deprecated Command Names and Their Corresponding New Commands

The following table lists the old commands and their corresponding new commands in Atrenta Console:

Old Command	New Command
set_option current_methodology	current_methodology
set_option stop_module	set_option stop
set_option define_macro	set_option define <macro>
set_option include_file_search_paths	set_option incdir <path>
set_option verilog_library_extensions	set_option libext {exts}
set_option verilog_library_files	set_option v {files}
set_option verilog_library_directories	set_option y {dirs}
set_option verilog_standard	set_option enableSV <yes | no> set_option disablev2k <yes | no>
set_option allow_duplicate_modules	set_option allow_module_override <yes | no>
set_option vhdl_standard	set_option 87 <yes | no>
set_option work_logical_name	set_option work <arg>
set_option vhdl_sort_method	set_option sort <yes | no>
set_option hdl_parameter	set_option param <parameter>
set_option analyze_designware	set_option dw <yes | no>
set_option black_boxes	set_option inferblackbox <yes | no> set_option nobb <yes | no>
set_option message_printing_limit	set_option lvpr <value>
set_option define_synthesis_pragma	set_option pragma <list-of-pragmas>
set_option enable_clock_gating	set_option sgsyn_clock_gating
set_option clock_gating_threshold	set_option sgsyn_clock_gating_threshold
set_option memory_size_limit	set_option mthresh <value>

Old Command	New Command
set_option handle_large_memory	set_option handlememory <yes | no>
set_option enable_save_restore	set_option enable_save_restore
set_option precomp_lib_check	set_option hdllibdu <yes | no>
set_option cache_directory	set_option cachedir <dir-name>
add_file	read_file
create_report	set_option report
create_report -file	set_option reportfile
create_report -maxsize	set_option report_max_size
create_report -style	set_option report_style
define_precomp_lib	set_option libhdlfiles <library-name> {space separated file list }
define_lib_map	set_option lib <lib-name>
define_goal_setup	current_goal

Project File Commands in Atrenta Console

Overview

A project file stores details of a session run in Atrenta Console. These session details are in the form of project file commands. For details on these commands, see *Project File Commands*.

For details of the commands that are not recommended in the project-based flow, see Options Not Recommended.

Project File Commands

The project file commands are described below.

87

Usage: `set_option 87 <yes | no>`

For details on this command, see *Run in VHDL87 Compatibility Mode*.

abstract_file_name_style

Use this command to control the naming style of the abstract files generated by SpyGlass.

The following table describes the valid input values that you can specify, their description, and related example:

Permissible Value	Description	Example
short	Generates short file name in the *<module_name>_<product>_abstract.sgdc* format. *For example, deep_cdc_abstract.sgdc*	deep_cdc_abstract.sgdc
compat	Generates unique file names with parameter details in the *<module_name>_<parameter_details>_<product>_abstract.sgdc format.*	deep_BUS_1_WIDTH_1_cdc_abstract.sgdc

By default, the value of the *abstract_file_name_style* command is set to `compat`.

The following is the usage of the *abstract_file_name_style* command:

Usage: `set_option abstract_file_name_style <compat | short>`

abstract_searchpath

Usage: `set_option abstract_searchpath { <dir_path1> <dir_path2> }`

Use this command during the SoC flow to locate block SGDC files during abstract view use run.

If the block SGDC files are moved or relocated to some other path, use this option to specify the new block SGDC path.

The following example shows the usage of this command:

```
set_option abstract_searchpath {./block1_sgdc/}
```

active_design

Usage: `set_goal_option active_design <reference | implement>`

Use this command to specify the type of design, `reference` or `implement`.

In the scope of the current goal, when you set this command to `reference` and specify a source list file by using the `reference_design_sources` command, SpyGlass considers that source list file for the current goal run and ignores the files specified by the `read_file` command. However, if you set the `active_design` command to reference outside the scope of the current goal or do not specify this command at all, SpyGlass ignores the source list file specified by the `reference_design_sources` command and considers the files specified by the `read_file` command.

For example, consider the following commands in a project file:

```
read_file -type verilog test.v
read_file -type sgdc test.sgdc

current_goal G1
set_goal_option active_design reference
set_goal_option reference_design_sources a.f
```

```
current_goal G2
set_goal_option reference_design_sources b.f
```

In the above example, SpyGlass picks the a.f file for the G1 goal. However, it picks the test.v and test.sgdc file for the G2 goal.

However, in the scope of the current goal, when you set this command to `implement` and specify a source list file by using the `implement_design_sources` command, SpyGlass considers that source list file in the current goal run and ignores the files specified by the `read_file` command at the global level. However, if you set the `active_design` command to implement outside the scope of the current goal, SpyGlass ignores the source list file specified by the `implement_design_sources` command and considers the files specified by the `read_file` command.

addrules

Usage: `set_goal_option addrules <rule-names>`

Use this command to:

- Enable a rule
- Add a rule in the current goal

The following command enables the *W448* rule that is switched off by default:

```
set_goal_option addrules W448
```

allow_celldefine_as_top

Usage: `set_option allow_celldefine_as_top <yes | no>`

Use this command to perform rule-checking on the top hierarchy of a top module.

By default, a top that is inside a 'celldefine module is ignored for rule-checking.

allow_fatal_downgrade

Usage: `set_option allow_fatal_downgrade <yes | no | 1 | 0 >`

Cancels overloading of the SGDC rules, which are not built-in and have a FATAL severity. This option is available for backward compatibility only.

allow_incr_save_restore

Usage: `set_option allow_incr_save_restore <yes | no>`

Use this command to enable incremental save restore feature. This command is valid only when specified with the *enable_save_restore* command.

NOTE: *The allow_incr_save_restore command can not be used with the dw command in the designware flow.*

For details on this command, see *Incremental Save and Restore*.

allow_module_override

Usage: `set_option allow_module_override <yes |no>`

For details on this command, see *Allow Duplicate Module Names in Verilog Designs*.

allow_non_lrm

Usage: `set_option allow_non_lrm <1 | 0>`

Enables parsing of constructs, described in *Table 1*, in Verilog parser. These constructs are not standard lrm.

By default, the value of this command is 0. In this case, SpyGlass flags non-standard lrm constructs as syntax errors.

TABLE 1 Constructs Supported by the allow_non_lrm Command

Construct	Example

Support for use of port/net before its complete declaration	`module top (a, b, sel, out1, out2, out3 );`
	`input a, b, sel;`
	`output wire out3, out2;`
	`assign out3 = out2 ? a : out1 ;`
	`output wire out1;`
	`endmodule`

Support for re-declaration of genvar
STX_VE_600 for genvar re-declaration is replaced by WRN_1048
	`genvar i;`
	`genvar i;    // Warning`

Support for non-standard label style of assignment pattern
WRN_1047 is flagged for each non-standard usage
	`int a[4] = {default:0};  // Warning [non-standard]`
	`int a[4] = '{default:0}; // Fine [standard]`

Support for assert final
WRN_1046 is flagged for each assert final
	`a1: assert final (out1 == ~in1)`
	`$fatal (2,"System Task FATAL executed");`
	`  is treated as`
	`     a1: assert #0 (out1 == ~in1)`

Support for macro argument replacement within double quotes	`` `define A(a) "a" ``
	`module top;`
	`` int a = `A(X); ``
	`endmodule`
	With switch `` ` ``A(X) is replaced by "X" whereas by default it is replaced by "a"

TABLE 1 Constructs Supported by the allow_non_lrm Command

Construct	Example

Support to relax STX_VE_800 by truncating evaluated value of expression used to represent ranges if they exceed 32bit limit	`parameter NUM = 1'b0;`
	`input wire [0: 85'b0 - 1 ] q_sec;`

Removed STX_810 fatal by considering functions having a parameter affected by a defparam statement as constant	NA
Relaxing STX_299 into WRN_1059 due to same type from same name package compiled into different libraries	NA

Relaxing the sign and state mismatched layout for array assignment compatibility checks
It will result in replacement of some violations of STX_VE_279, STX_VE_467, STX_VE_292 and STX_VE_299 by WRN_1462
	`module top;`
	`    logic signed conn1[3:0];`
	`    logic conn2[3:0];`
	`    bottom inst1(.a(conn1));        //`
	`Error :`
	`   incompatible port connection due to`
	`    mismatch in sign`
	`    bottom inst2(.a(conn2));        //`
	`Fine  :`
	`    compatible port connection`
	`endmodule`
	
	`module bottom(input wire a[3:0]);`
	`endmodule`

Support to allow assignment of concatenation to unpacked type
It will result in replacement of some violations of STX_VE_462, STX_VE_467, STX_VE_292 and STX_VE_299 by INFO_994
	`int a[4] = {0,1,2,3};  // Info`
	`int a[4] = '{0,1,2,3}; // Fine`

allow_pre_packaged_goals

Usage: `set_option allow_pre_packaged_goals <yes|no>`

Use this command to run all the optional and mandatory rules of the *SpyGlass lint* product, without explicitly editing the goal spq files.

allow_recursion_limit

Usage: `set_option allow_recursion_limit <recursion_limit>`

Use this command to specify the recursion limit for a VHDL function. You can specify any positive integer value to specify the recursion limit. If the VHDL function call recursion exceeds the value specified using this parameter, the STX_VH_437 rule reports a violation and exits recursion. The default value for this option is 10420.

aggregate_report

Usage: `set_option aggregate_report {<report-names>}`

Use this command to specify the type of aggregate reports, that is, *project_summary*, *datasheet*, or *dashboard*, that you want to generate.

aggregate_report_config_file

Usage: `set_option aggregate_report_config_file <config-file-path>`

Use this command to specify a configuration file that contains a list of projects and run directories generated by batch console or GUI. This data is used to generate the specified aggregate report.

aggregate_reportdir

Usage:

```
set_option aggregate_reportdir <report-directory-path>
```

Use this command to specify a directory in which you want to generate data for the specified aggregate report.

auto_restore

Usage: `set_option auto_restore <yes |no>`

Enables restore of design query data as part of the `current_goal/` `open_project` Tcl commands. You can automatically restore the design information saved earlier by enabling this option in the project file. It is useful for executing Tcl procedures accessing this design information.

auto_save

Usage: `set_option auto_save <yes |no>`

Enables save of design query data as part of the goal execution. The design attributes are computed during the goal execution. These attributes along with the synthesized/flattened design view are saved when the auto_save option is enabled in the project file. You can retrieve this information at a later point without a need to re-run the goal.

block_abstract

Usage: `set_option block_abstract <yes | no | true | false>`

Use this command to generate an abstract view during block-level verification in the SpyGlass CDC flow.

The following example shows the usage of this command:

```
set_goal_option block_abstract  yes
```

block_abstract_directory

Usage:

```
set_option block_abstract_directory <directory>
set_goal_option block_abstract_directory <directory>
```

Use this command to specify a directory in which the abstract view of a block should be saved.

For details of this command, see *Block Abstract Directory*.

cachedir

Usage:

```
set_option cachedir <directory-name>
```

Use this command to specify a cache directory where you want library compilation to be performed.

By default, the value of this option is set to `spyglass_cache`.

cell_library

Usage:

```
set_option cell_library <library-name>
set_goal_option cell_library <library-name>
```

Use this command to skip rule-checking on design units (Verilog/VHDL) that are loaded from precompiled libraries specified by this command. However, such units are synthesized by SpyGlass.

For example, consider that you have four precompiled libraries, `L1`, `L2`, `L3`, and `L4`, and suppose that you want to skip rule-checking on design units loaded from the libraries, `L2` and `L4`. In this case, specify the following command:

```
set_option cell_library { "L2"  "L4" }
```

check_celldefine

Usage:

```
set_option check_celldefine <yes | no>
set_goal_option check_celldefine <yes | no>
```

Use this command to enable rule-checking on all the `'celldefine` modules.

By default, Verilog `'celldefine` modules are not checked by SpyGlass as they are leaf-level library cells, and any independent checks (such as style checks like indentation) on these cells are not desired by most of the users. However, if you still want to check for such modules, use the `check_celldefine` command.

If you do not use this command, SpyGlass reports a warning specifying that rule-checking for `'celldefine` modules is off.

NOTE: *Also see the* `allow_celldefine_as_top` *command.*

checkip

Usage: `set_option checkip <ip-name>`

For details on this command, see *Check IP*.

checkdu

Usage: `set_option checkip <du-name>`

For details on this command, see *Check DU*.

checkTopDu

Usage: `set_parameter checkTopDu <rule-names>`

Use this command to specify a space-separated list of rules that should run for the top-level design.

For example, consider the following command:

```
set_parameter checkTopDu "UndrivenOutPort-ML UndrivenNet-ML"
```

In the above example, the UndrivenOutPort-ML and UndrivenNet-ML rules will check top-level design unit only.

You can also set the value of the `checkTopDu` command to `yes` so that all rules check at the top level.

NOTE: *Behavior of post-flattening rules is not affected by the* `checkTopDu` *command. For such rules, create waive commands to waive hierarchical violations.*

configfile

Usage: `set_option configfile <file-name>`

Use this command to specify a configuration file with the highest priority.

See the *Overview* section of *The Configuration File in Atrenta Console* chapter.

consolidate_reportdir

Usage: `set_option consolidate_reportdir <directory-path>`

Use this command to specify a directory path where SpyGlass should create links to the reports generated after SpyGlass run.

By default, SpyGlass stores links to the reports at the following path:

<project-working-directory>/<project-name>/consolidated_report/

For details, see *Directory Structure of Generated Reports*.

convert_udp_to_latch

Usage: `set_option convert_udp_to_latch <yes | no>`

Use this command to infer UDP as a latch while translating the UDP with both edge and level sensitiveness.

By default, SpyGlass infers a UDP as a flip-flop in such cases.

debug_comments

Usage: `set_goal_option debug_comments {"<text>"}`

Use this command to specify the debug comments to identify a goal/ scenario in the correlation view. The text/debug comment specified using this option is visible in the top row of each column of trials matrix.

decompile_block_constraints

Usage: `set_option decompile_block_constraints <yes |no |1 |0>`

Use this command to decompile the block file specified using the abstract_file command.

This facilitates porting of individual abstract SGDC files from creation directory to separate directory under the source control.

default_waiver_file

Usage: `set_option default_waiver_file <file-name>`

Use this command to specify a default waiver file for saving interactive waiver commands.

If you do not specify a default waiver file by using this command, Atrenta Console considers the *<project-wdir>/<project_name>.swl* file as the default waiver file.

define

Usage: `set_option define <list-of-options>`

For details on this command, see *Enter Macros for Analysis*.

define_cell_sim_depth

Usage:

```
set_option define_cell_sim_depth
<cell_simulation_depth_depth>
```

Enables you to increase the threshold limit for the size of the macro in order to completely simulate the macro.

Synthesis Macro Logic Evaluation is done partially, if the macro size crosses the threshold limit. The default threshold limit is data bus size of 1024, and macros with data bus size > 1024 are partially simulated.

In order to completely simulate such big macros to improve the accuracy of the logic simulation, you can increase the threshold limit using this command.

This command takes the threshold limit in terms of size of data bus given as a power of 2. For example, if the value of this parameter is specified as 12, then data bus of size of 4096 (2^12) is completely evaluated.

The default value for this command is 10. You can set the value in the range of 1 to 30, both limits inclusive.

A lower value indicates a quick evaluation but improper result. However, a higher value indicates a longer evaluation period but more accurate result.

You can increase the threshold limit during evaluation for the following synthesis macros using the define_cell_sim_depth command:

- M_RTL_ARITH_SHIFT
- M_RTL_MUX_N
- M_RTL_PRIM_MUX
- M_RTL_LSHIFT
- M_RTL_RSHIFT
- M_RTL_SHIFT
- M_RTL_ROTATE

If the size of any of the above macros exceeds the threshold limit and these macros are partially simulated, then SpyGlass reports FLAT_504 message. This message indicates the value, which is required for the define_cell_sim_depth command, in the current run to completely simulate these macros.

define_incr_dirmap

Usage:
```
set_option define_incr_dirmap <dir1> <dir2>
set_goal_option define_incr_dirmap <dir1> <dir2>
```

Use this command to map different locations of RTL files. It is used by an incremental algorithm to find if RTL files in the current run are the same as the RTL files in old SpyGlass run with respect to the incremental analysis that is being performed.

define_regression

Usage:
```
define_regression <regression-name> -goals {<goals-list>}
```

For more information on the Tcl-based usage of the *define_regression* command, refer to the *define_regression* section of the *SpyGlass Tcl Shell Interface User Guide*.

Use this command to define a regression by:

- Specifying a list of goals to be run in the sequential mode, and
- Assigning a name *<regression-name>* to the goals list.

Once you define a regression, specify the regression name to the *-goals <goals>* command-line option to run the goals of that regression in the sequential mode.

This command is useful when the list of goals to be run in the sequential mode is huge. Specifying such huge list of goals every time you run the *-goals <goals>* command-line option can be time-consuming. Therefore, specify this list in one go by using the `define_regression` command and later specify the list name (*<regression-name>*) to the *-goals <goals>* command-line option.

NOTE: *If a regression name specified by the -goals <goals> batch command is not declared in the project file, SpyGlass searches the specified goal in the current methodology.*

Example of Using the define_regression Command

The following example shows the usage of this command:

```
// Project1.prj
define_regression Reg1 -goals {G1,G2,G3}
```

Now consider that you specify the above project file in batch by using the following command:

```
spyglass -projectfile Project1.prj -goals Reg1
```

When you run the above command, SpyGlass searches `Reg1` in the project file. As the project file contains the declaration of `Reg1` through the `define_regression` command, SpyGlass runs the `G1`, `G2`, and `G3` goals in the sequential mode.

NOTE: *If you want to exempt any goal from the list of goals belonging to a regression, create a new regression by using the define_regression command such that the new regression does not have that goal.*

Creating a Regression File

A regression file is a Tcl file in which you can define regressions by using the `define_regression` commands. You must keep this file parallel to the order file in the current methodology.

After creating the regression file, when you specify a regression name to the *-goals <goals>* command-line option, SpyGlass looks for that regression name in the regression file and runs the goals of that regression in the sequential mode.

Consider an example in which Methodology/New_RTL is the current methodology and you have created the following regression_run.tcl file in this methodology directory:

```
define_regression Reg1 -goals {G1,G2,G3}
define_regression Reg2 -goals {G4,G5,G6}
```

Now, when you specify `Reg1` to the *-goals <goals>* command-line option, SpyGlass searches `Reg1` in the regression_run.tcl file and runs the `G1`, `G2`, and `G3` goals in the sequential mode.

However, if a regression name defined in a regression file matches with the regression name defined in a project file, SpyGlass gives preference to the regression defined in the project file. For example, if a regression file defines the A, B, and C regressions and a project file defines the C and D regressions, SpyGlass considers all the A, B, C, and D regressions where the C regression is picked from the project file.

define_severity

Usage:

```
set_goal_option define_severity <product-name>+<severity-
label>+<severity-class>
```

Where:

- <product-name> is the product mnemonic for the product for which you are specifying the new severity label.
- <severity-label> is the new severity label being defined.
- <severity-class> is one of the pre-defined rule severity classes.

Do not add spaces between values and the intervening + characters.

Use this command to define a severity label for the current SpyGlass run.

This command is usually used in conjunction with the `overloadrules` command.

The following example defines the new severity label `myFATAL` under the pre-defined severity class `FATAL` for SpyGlass lint solution (product mnemonic is `lint`):

```
set_goal_option define_severity lint+myFATAL+FATAL
```

designread_disable_flatten

Usage:

```
set_option designread_disable_flatten <yes | no>
```

Disables flattening during the *compile_design* command in the sg_shell. Also disables flattening while opening a project, if the project was closed with a flattened view.

designread_enable_synthesis

Usage:
```
set_option designread_enable_synthesis <yes | no>
```

For details on this command, see *design-read Synthesis Flavor*.

designread_synthesis_mode

Usage:
```
set_option designread_synthesis_mode <mode>
```

For details on this command, see *design-read Synthesis Flavor*.

disable_auto_migrate_waiver

Usage:
```
set_option disable_auto_migrate_waiver
<true|false|on|off|yes|no|1|0>
```

Use this command to disables the automatic migration of waivers.

disable_encrypted_hdl_checks

Usage:
```
set_option disable_encrypted_hdl_checks <yes|no>
set_goal_option disable_encrypted_hdl_checks <yes|no>
```

For details on this command, see *Disable Encrypted HDL Checks*.

disable_hdllibdu_lexical_checks

Usage:
```
set_option disable_hdllibdu_lexical_checks <yes|no>
set_goal_option disable_hdllibdu_lexical_checks
<yes|no>
```

Use this command to disable lexical rule checking on precompiled libraries.

disable_hdlin_synthesis_off_skip_text

Usage: `set_option disable_hdlin_synthesis_off_skip_text <yes|no>`

Use this command to disable interpretation of VHDL design code between Synopsys synthesis_off/synthesis_on pragma pair as comments.

By default, SpyGlass interprets the VHDL design code between Synopsys synthesis_off/synthesis_on pragma pair as comments. This command disables this behavior, and the code between Synopsys synthesis_off/ synthesis_on pragma pair is checked by SpyGlass.

If this command is specified along with the hdlin_synthesis_off_skip_text command, then it is ignored, and the VHDL design code between Synopsys synthesis_off/synthesis_on pragma pair is treated as comments.

This option can also be specified as part of sources.f supplied with the libhdlf command, while precompiling a given logical library. It enables parsing of VHDL code between Synopsys synthesis_off/synthesis_on pragma pair for design files precompiled as part of the given logical library.

disable_hdlin_translate_off_skip_text

Usage: `set_option disable_hdlin_translate_off_skip_text <yes|no>`

Use this command to disable interpretation of VHDL design code between Synopsys translate_off/translate_on pragma pair as comments.

By default, SpyGlass interprets the VHDL design code between Synopsys translate_off/translate_on pragma pair as comments. This command disables this behavior, and the code between Synopsys translate_off/ translate_on pragma pair is checked by SpyGlass.

If this command is specified along with the hdlin_translate_off_skip_text command, then it is ignored, and the VHDL design code between Synopsys translate_off/translate_on pragma pair is treated as comments.

This option can also be specified as part of sources.f supplied with the libhdlf command, while precompiling a given logical library. It enables parsing of VHDL code between Synopsys translate_off/translate_on pragma pair for design files precompiled as part of the given logical library.

disable_html_report

Usage:
```
set_option disable_html_report
 { < datasheet | dashboard | html > }
```

Use this command to disable the auto-generation of the specified reports. You can specify 'datasheet', 'dashboard' and/or 'html' to stop auto-generation of either of the reports.

The following command disables the datasheet and dashboard reports from being generated automatically:

```
set_option disable_html_report {datasheet dashboard}
```

disable_report

Usage:
```
set_goal_option disable_report { <report-names> }
```

Use this command to disable the generation of the default reports other than the following reports:

moresimple report	*no_msg_reporting_rules report*	*stop_summary report*
elab_summary report	*ignore_summary report*	

The following command disables the waiver and summary reports:

```
set_goal_option disable_report {R1 R2}
```

NOTE: *Do not disable a report that is specified by using the report command.*

disable_infer_async_rst_latch

Usage:

```
set_option disable_infer_async_rst_latch { <module-names>
}
```

By default, Spyglass synthesis infers latches with asynchronous set/reset. When this command is set for specified modules, Spyglass synthesis infers latches without asynchronous set/reset, for such modules.

disable_sgdc_dump

Usage:

```
set_option disable_sgdc_dump <sg_multicycle_path |
false_path>
```

Use this command to disable the conversion of certain SDC commands to their respective SGDC commands in the sdc2sgdc flow.

The following table describes the valid values available for this command and the respective behavior:

TABLE 2 disable_sgdc_dump values

Value	**Behavior**	**Command**
sg_multicycle_path	Disables the conversion of the set_multicycle_path SDC constraint to the sg_multicycle_path SGDC constraint	`set_option disable_sgdc_dump sg_multicycle_path`
false_path	Disables the conversion of the set_false_path SDC constraint to false_path SGDC constraint	`set_option disable_sgdc_dump false_path`
false_path sg_multicycle_path	Disables the conversion of set_false_path and set_multicycle_path SDC constraints to false_path and sg_muticycle_paths SGDC constraints	`set_option disable_sgdc_dump {false_path sg_multicycle_path}`

disable_Vlog2005_lrm_naming

Usage: `set_option disable_Vlog2005_lrm_naming <0 | 1>`

Use this command to disable the Verilog-2005 LRM naming style.

The following example shows the usage of this command:

```
set_option disable_Vlog2005_lrm_naming 1
```

disallow_view_delete

Usage: `set_option disallow_view_delete <yes | no>`

Use this command to disable large design processing mode.

When you disable large design processing mode:

- All rules are run as in versions before SpyGlass version 3.2.0.
- Both RTL and netlist views remain in the memory for all types of rules.

NOTE: *You cannot use the* `higher_capacity` *and* `disallow_view_delete` *commands together as they are complementary.*

dnc_param

Usage: `set_option dnc_param <val_list>`

Sets the specified VHDL generics and Verilog parameters as do not care during abstract model generation.

You can specify the key name in the `module-name.parameter-name` format for Verilog and in the `entity-name.generic-name` format for VHDL. The names are case-sensitive for Verilog parameters and case-insensitive for VHDL generics.

In a normal design hierarchy, it is possible to define parameterizable design units, where the value of the parameter (in Verilog) or generic (in

VHDL) is defined when the design unit is instantiated in the hierarchy. In order to define a value for a parameter or a generic as do not care during abstract model generation, use the `dnc_param` option.

To set the `dnc_param` option for a generic or parameter, specify the instance of the parameter you wish to define.

For example, to set the VHDL generic width in entity control as do not care, specify the following command as part of the SpyGlass invocation:

```
<sg_shell>> set_option dnc_param "control.width"
```

To set the Verilog parameter limit in module block1 as do not care, specify the following command as part of the SpyGlass invocation:

```
<sg_shell>> set_option dnc_param "block1.limit"
```

dump_all_modes

Usage: `set_option dump_all_modes <yes | no>`

Use this command to create both 32-bit and 64-bit versions of the precompiled HDL sources irrespective of the architecture.

dump_report_dir

Usage: `set_option dump_report_dir <custom-dir-name>`

Use this command to specify the directory where the `spyglass_run_stat.log`, `spyglass_violations_report.xml`, and the `spyglass_rules_report.xml` reports should be saved.

dump_inactive_rules

Usage: `set_option dump_inactive_rules <yes | no>`

Use this command to print inactive rules in the spyglass_rules_report.xml file.

dw

Usage: `set_option dw <yes | no>`

For details on this command, see *Enable Analysis of Instantiated DesignWare Components*.

dw_options

Usage:

```
set_option dw_options <hide_all_dw_violations |
dont_use_tpts  | report_violations | enable_waiver>
```

Use this command to specify following options for the DesignWare components:

- **hide_all_dw_violations**: This is the default option. In this case, none of the violations, even if waived, are reported in the waiver report.
- **dont_use_tpts**: Disables use of third-party tools in the dw run.
- **report_violations**: Disables all DW waivers, so they show as any other violations.
- **enable_waiver**: Waives violations flagged on dw components. Such waived violations are reported in the waiver report.

If none of report_violations or enable_waiver options are specified, then violations on dw components are waived. Additionally, the violations are not reported in the waiver report.

elab_precompile

Usage: `set_option elab_precompile <yes | no>`

Use this command to enable elaboration in single-step precompilation.

enable_amg

Usage:

```
set_option enable_amg <yes | no>
```

Use this command to enable the AMG flow in base synthesis.

enable_abstract_blocks_schematic

Usage:

```
set_option enable_abstract_blocks_schematic
<abstract_module_names>
```

For details of this command, see *Enable Abstract Blocks Schematic*.

enable_const_prop_thru_seq

Usage:

```
set_option enable_const_prop_thru_seq <yes | no>
set_goal_option enable_const_prop_thru_seq <yes|no>
```

Use this command to enable constant propagation beyond sequential elements during logic simulation.

If you set the value of the `enable_const_prop_thru_seq` command to `yes` in the project file, the `set_case_analysis/test_mode` values propagate beyond sequential elements. Constant propagation from latch-D/flip-flop happens only if one of the following conditions is true:

- Latch/flip-flop does not have preset/clear pin.
- Data is tied to 0 and latch/flip-flop has only clear pin.
- Data is tied to 1 and latch/flip-flop has only preset pin.

NOTE: *The* `enable_case_anal_seq_prop` *option has been renamed to* `enable_const_prop_thru_seq`*.*

enable_fpga

Usage:	`set_option enable_fpga <yes	no>`

This command converts the non-synthesizable FPGA constructs into synthesizable Verilog RTL gate constructs. The following table shows the mapping of the non-synthesizable constructs and their corresponding synthesizable constructs in Verilog:

Non-synthesizable construct	Corresponding synthesizable construct
Nmos	bufif1
Pmos	bufif0
Rtran	buf1
Tri1	wire
Tri0	wire
Time	reg [63:0]

enable_non_sgdc_naming

Usage:	`set_option enable_non_sgdc_naming <0	1	no	yes>`

Use this command to specify non-SGDC style names, that is, SDC style names, in the SGDC file.

You can specify an instance, port, or, net name in an RTL in either SDC or SGDC formats.

Following table lists the key differences in the usage of the SDC and SGDC formats:

Difference	SDC Format	SGDC Format

Escape Character	For SDC compliant names, there is no need of giving an escape character for names containing special character as a part of the name.
	Example:
	`"mid/lower@leaf"`
	`(Hierarchy separator is '/')`
		An SGDC name needs an escape character ('\' backward slash in our case) plus a white space at the end to mark the end of the name.
		Example:
		`"top.mid.\lower@leaf "`
		`(Hierarchy separator is '.' And also note that '\' is added to identify '@' as a part of the name and '  ' white space at the end to mark the end of the name.)`

RTL Names	No special characters needed in the path **Example:** `gen_block[0]/U1/q_reg[0]`	Full hierarchial name specified using the special character, such as, '.' **Example:** `top.\gen_block[0] .U1.q_reg[0]`

Using the *enable_non_sgdc_naming* command to facilitate the search for the above specified SDC style naming conventions.

enable_gateslib_autocompile

Usage: `set_option enable_gateslib_autocompile <yes | no>`

The default value of *enable_gateslib_autocompile* option is yes.

For details on this command, see *Enable auto-compilation of gateslib into sglib*.

Also see *force_gateslib_autocompile*.

enable_hbo

Usage:

```
set_option enable_hbo <yes|no|1|0|on|off|true|false>
```

Use this command to enable hierarchical boundary optimization in synthesis.

NOTE: *This options is valid only for est mode of synthesis.*

By default, in optimized synthesis flow, optimizations are performed at module level only. Across module/hierarchy optimizations are not performed in default flow.

In hierarchical boundary optimization, information about constant and unloaded nets on instance interface is propagated across boundaries and corresponding logic optimization is performed.

enable_hdl_encryption

Usage:

```
set_option enable_hdl_encryption <yes | no>
```

For details on this command, see *Enable HDL Encryption*.

enable_inactive_rtl_checks

s

Usage:

```
set_option enable_inactive_rtl_check <yes | no>
set_goal_option enable_inactive_rtl_check <yes | no>
```

Use this command to enables semantic checking capability in SpyGlass.

enablePackaging

Usage:

```
set_option enablePackaging  <yes|no>
```

Use this command to enable packaging of block SGDC files along with abstract views.

enable_pass_exit_codes

Usage:
```
set_option enable_pass_exit_codes <yes |no>
set_goal_option enable_pass_exit_codes <yes |no>
```

Use this command to print detailed exit status codes and messages. For details, see *SpyGlass Exit Status*.

enable_pgnetlist

Usage:
```
set_option enable_pgnetlist <yes |no>
```

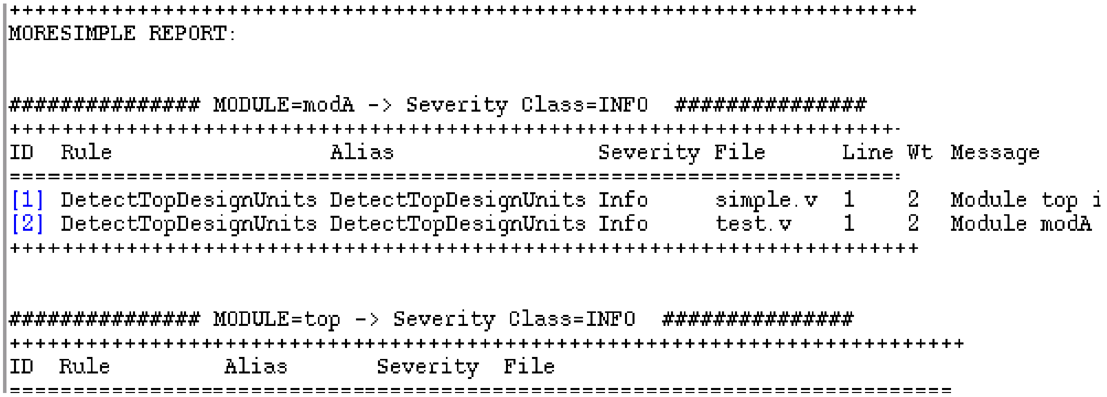
Use this command to enable power and ground pin information to consider from the specified physical libraries.

enable_module_based_reporting

Usage:
```
set_option enable_module_based_reporting <yes | no>
```

Use this command to group violations based on module names in SpyGlass reports.

The following figure shows the moresimple report in which violations are grouped based on module names when you set this command to `yes`:



```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
MORESIMPLE REPORT:


############### MODULE=modA -> Severity Class=INFO  ###############
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
ID  Rule                 Alias                Severity File      Line Wt  Message
=====================================================================:
[1] DetectTopDesignUnits DetectTopDesignUnits Info     simple.v  1    2   Module top i
[2] DetectTopDesignUnits DetectTopDesignUnits Info     test.v    1    2   Module modA
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


############### MODULE=top -> Severity Class=INFO  ###############
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID  Rule         Alias       Severity File
==========================================================================
```

enable_power_platform_flow

Use this command to enable the Power Explorer feature of the SpyGlass Power Estimation product. This command enables various views that contains data related to register and memory instance of design. By Default this feature is off.

Usage: `set_option enable_power_platform_flow <yes | no>`

enable_precompile_vlog

Usage: `set_option enable_precompile_vlog <yes | no>`

Use this command to enable the precompiled Verilog library feature.

See *Precompiling Verilog Libraries* topic of *Atrenta Console User Guide* for more details.

By default, the precompiled Verilog library feature is not enabled.

enable_physical_aware_pe

Usage: `set_option enable_physical_aware_pe <yes | no>`

Use this command to enable physical aware power estimation flow and check out required licenses. It is used in `physical_power_postfloorplan` and power_est_average goals.

By default, this option is not set and SpyGlass runs normally.

enable_rule_category_in_moresimple

Usage: `set_option enable_rule_category_in_moresimple <yes | no | 1 | 0>`

Use this command to include the `Rule Category` column in the moresimple.rpt report.

enable_rule_mnemonic

Usage: `set_option enable_rule_mnemonic <yes | no | 1 | 0>`

Use this command to enable the support for rule mnemonics in the SpyGlass run.

For more information on the support of rule mnemonics, see *Using Rule Mnemonics*.

enable_save_restore

Usage: `set_option enable_save_restore <yes | no>`

The default value of the *enable_save_restore* option is `yes`.

For details on this command, see *Enable Save Restore Flow*.

enable_save_restore_builtin

Usage: `set_option enable_save_restore_builtin <yes | no>`

The default value of the *enable_save_restore_builtin* option is yes.

For details on this command, see *Enable Save Restore for BuiltIn Rules*.

enable_sgdc_debug

Usage: `set_option enable_sgdc_debug <yes | no>`

Set this command to `yes` to generate the debug_sgdc report under the spyglass_reports/SpyGlass directory.

The debug_sgdc report contains decompiled SGDC commands with back reference information.

enable_unused_param_reporting_in_spq

Usage: `set_option enable_unused_param_reporting_in_spq <yes | no>`

Use this command to report violations for unused parameters, which are mentioned in the .spq files.

enable_sglib_debug

Usage:

```
set_option enable_sglib_debug <yes | no>
set_goal_option enable_sglib_debug <yes | no>
```

Use this command to generate the debug_sglib and sglib_version_summary reports.

The debug_sglib report contains the inferred functionality for each gate that was successfully synthesized by the SpyGlass Library Compiler.

The sglib_version_summary report lists the sglib names, their compiling library compiler version, and their status. The report also lists enhancements made in subsequent SpyGlass library compiler releases starting from the oldest version of library files used in the current SpyGlass run.

NOTE: *You must recompile your gates library files with SpyGlass 3.8.2 or higher to generate the* debug_sglib *report.*

enable_vlog_config

Enables the support for verilog configuration. For more information on verilog configuration, see *Support for Verilog Configuration*.

Usage: `set_option enable_vlog_config <yes | no>`

enableSV

Usage: `set_option enableSV <yes | no>`

For details on this command, see *Enable SystemVerilog Processing*.

enable SVA

Usage: `set_option enableSVA <1 | 0>`

Enables parsing of SystemVerilog constructs and also handles the SV Assert logic. The default value for this option is 0. Therefore, SpyGlass considers system verilog constructs as syntax errors.

enableSV09

Usage: `set_option enableSV09 <0 | 1 | yes | no>`

Use this command to enable parsing of SystemVerilog constructs.

By default, SpyGlass reports SystemVerilog constructs as syntax errors.

filter_block_violation

Usage: `set_option filter_block_violation {<space-separated-list>}`

Where <space-separated-list> can have the following:

- errors
- warnings
- waived_errors
- waived_warnings
- builtin_errors
- builtin_warnings
- builtin_errors_waived
- builtin_warnings_waived

Use this command during the SoC flow to filter violations reported during block SGDC generation.

The following example shows the usage of this command:

```
set_option filter_block_violation { errors warnings
builtin_errors builtin_warnings builtin_errors_waived
waived_errors }
```

force_gateslib_autocompile

Usage: `set_option force_gateslib_autocompile <yes|no>`

Set this command to `yes` to forcefully compile Synopsys Liberty(TM) files (.lib files) to SpyGlass-compatible format library files (.sglib files).

Using this command automatically implies the use of the *enable_gateslib_autocompile* command.

If you set the `force_gateslib_autocompile` command to `yes` to automatically compile gate libraries, any criteria for re-compilation of gate libraries is not evaluated. In such cases, the specified .lib files are always compiled and these files overwrite the existing .sglib file present in the cache directory.

gen_block_options

Usage: `set_option gen_block_options { <list-of-blocks> }`

Use this command to generate the block dependency report for certain design units.

gen_block_expand_lib_sources

Usage: `set_option gen_block_expand_lib_sources <true | false>`

Lists specification information about source files, as comments, for precompiled libs, incdir, and -y/v directories. It also filters out irrelevant .lib specifications for the specified design unit or a block. Following examples explain the difference in the information reported using the *gen_block_options* and *gen_block_expand_lib_sources* commands.

Example 1

Consider a top module, top, in a design instantiates a module, mid, which has been precompiled into library, P1. Module, mid, further instantiates two modules, bottom1 and bottom2, that have been compiled in the library P2.

When you specify the *gen_block_options* command, it de-compiles all libraries, including WORK, in the report, as shown below:

```
read_file -type verilog {top.v}
set_option lib L1 "../P1"
set_option lib L2 "../P2"
set_option lib WORK "./top/WORK"
```

However, when you specify *gen_block_expand_lib_sources* command, it reports only relevant .lib specifications as shown below:

```
read_file -type verilog {top.v}
set_option lib L1 "../P2"
# read_file -type hdl {dir1/bottom1.v}
# read_file -type hdl {dir1/bottom2.v}
set_option lib L2 "../P1"
# read_file -type hdl {dir1/mid.v}
```

Example 2

Consider a top module, top, instantiates module, mid, defined in the file, y_dir1/mid.v. The module, mid, further instantiates another module, bottom, defined in the file, y_dir2/bottom.v. Also assume that the file, bottom.v, uses a parameter 'REGSIZE' defined in inc1/include1.v.

When you specify the *gen_block_options* command, the top module generates the following report:

```
set_option define {REGSIZE=4}
read_file -type verilog {top.v}
set_option incdir {inc1/}
set_option y {y_dir1/}
set_option libext {.v}
set_option y {y_dir2}
set_option libext {.v}
```

However, when you specify the *gen_block_expand_lib_sources* command, the top module generates the following report:

```
set_option define {REGSIZE=4}
read_file -type verilog {top.v}
set_option incdir {inc1/}
# read_file -type hdl {inc1/include1.v}
set_option y {y_dir1/}
set_option libext {.v}
# read_file -type verilog {./y_dir1/mid.v}
set_option y {y_dir2}
set_option libext {.v}
# read_file -type verilog {./y_dir2/bottom.v}
```

gen_blk_sgdc

Usage:

```
set_option gen_blk_sgdc <yes | no>
set_goal_option gen_blk_sgdc <yes | no>
```

Set this command to `yes` to enable generation of an abstract view of a block.

NOTE: *This command is currently used by SpyGlass CDC solution only.*

gen_hiersgdc

Usage: `set_option gen_hiersgdc <yes | no>`

Use this command to enable hierarchical migration of block-level SGDC commands to the chip-level. See *Generating Hierarchical SGDC file* topic of *Atrenta Console User Guide* for more details.

generate_run_stat

Usage: `set_option generate_run_stat <yes | no>`

Use this command to create the `spyglass_run_stat.log` file.

generate_violations_report

Usage: `set_option generate_violations_report <yes | no>`

Use this command to create the `spyglass_violations_report.xml` file.

generate_rules_report

Usage: `set_option generate_rules_report <yes | no>`

Use this command to create the `spyglass_rules_report.xml` file, which contains all the active rules.

gensys_compatible_dump

Usage: `set_option gensys_compatible_dump <0 | 1>`

Use this command during the generation of a precompiled dump to generate extra information required by Atrenta GenSys platform.

handlememory

Usage:
```
set_option handlememory <yes | no>
```

Use this command to process memories in an optimized manner.

See the *Memory Reduction Feature* section for more details.

NOTE: *The handlememory option is ignored (for individual memories only), if the memory size is less than the specified mthresh value.*

You can also use the AUTOENABLE_MEMORY_HANDLING configuration setting to enable this feature.

hdlin_synthesis_off_skip_text

Usage:
```
set_option hdlin_synthesis_off_skip_text <yes | no>
```

For details on this command, see *Ignore VHDL code within pragma block 'synthesis'*.

hdlin_translate_off_skip_text

Usage:
```
set_option hdlin_translate_off_skip_text <yes | no>
```

For details on this command, see *Ignore VHDL code within pragma block 'translate'*.

hdllibdu

Usage:
```
set_option hdllibdu <yes | no>
set_goal_option hdllibdu <yes | no>
```

Use this command to perform RTL and lexical rule-checking on precompiled Verilog/VHDL design units that are directly or indirectly instantiated in the design being processed.

By default, this command is disabled, and SpyGlass does not perform RTL and lexical rule-checking on precompiled Verilog/VHDL design units.

When you set the hdllibdu option to yes, SpyGlass enables the lexical rule-checking on precompiled design units. This can result in an increased peak memory.

If the increase in peak memory is significant resulting in an out of-memory situation, use the *disable_hdllibdu_lexical_checks* project file command while running precompiled design units. This will disable lexical rule checking on precompiled design units.

higher_capacity

Usage: `set_option higher_capacity <yes | no>`

Use this command to:

- Disable rules that are designed for SpyGlass version 3.2.0 that require both RTL and netlist views, if any.
- Delete RTL view from memory before running rules designed for SpyGlass version 3.2.0 that require a netlist view only.

hw_variant_name

Usage: `set_option hw_variant_name <hw_variant>`

Use this command to view results of the same top module separately in DashBoard when executed under different hardware environments.

Each hardware variant may have different set of defines, file lists, parameters and may lead to a different synthesis for the same top module.

When user sets this command in the project files and the result directories are explicitly set, the SoC DashBoard shows all such hardware variant analysis results separately for tracking. Without this command, the SoC Dashboard accumulates all results of the same top module and displays them as one entity.

For example, assume that you want to execute a top module, my_top, under two different environments/projects, say, master2port and slave2port. To track these environments separately, include the following in the respective project files:

- In the `master2port` project:

  ```
  <other settings>
  set_option top my_top
  set_option hw_variant_name master2port
  set_option projectwdir ./my_top_master2port
  <other settings …>
  ```

- In the `slave2port` project:

  ```
  <other settings>
  set_option top my_top
  set_option hw_variant_name slave2port
  set_option projectwdir ./my_top_slave2port
  <other settings …>
  ```

NOTE: *To avoid overwriting results, if the project files are located in the same directory or the same project is being used for different hardware definitions, set the value of the projectwdir command.*

include_block_interface

Usage: `set_option include_block_interface <encrypted | abstract>`

Use this command to capture block interface information in the abstract model.

You can set the value of the include_block_interface command to either encrypted, or abstract, or both encrypted and abstract.

The following table describes the valid options supported by the include_block_interface command:

Option	Description	Example
abstract	Enables you to capture block interface information in the form of SGDC (abstract_interface_port, abstract_interface_param) in the abstract model	set_option include_block_interface { abstract }
encrypted	Enables you to capture block interface information encrypted in the abstract model	set_option include_block_interface { encrypted }
abstract encrypted	Enables you to capture block interface information in the form of both SGDC and encrypted information in the abstract model	set_option include_block_interface { abstract encrypted }

Recommendations when working with encrypted abstract model

Consider the following points when working with encrypted abstract model:

- In the encrypted abstract model, do not tamper or edit the encrypted part. This may corrupt the abstract model and, therefore, make it unusable for top-level SoC run.
- Avoid copy-paste of the encrypted part from one model to another model. This may corrupt the abstract model and, therefore, make it unusable for the SoC verification.
- Use the encrypted route for the HDL blocks having very complex interface.

include_opt_data

Usage: `set_option include_opt_data <true|false|yes|no>`

Use this command with the *enable_gateslib_autocompile* option to enable SpyGlass Library Compiler to generate advanced optimized data used in the SpyGlass Power Estimation for the specified gates library (.lib) files only, and store them as part of the resulting sglib (.sglib) files.

Using the `include_opt_data` command enables the preparation and storage of advanced optimized data inside the .sglib file for faster processing. It also helps in detecting errors in the .lib files early in the process resulting in better performance. Though performance improvement is specifically seen in SpyGlass power estimation flow, use of the .sglib file with advanced optimized data is not limited to this SpyGlass flow only. You can use it in all SpyGlass runs without any problem.

If SpyGlass fails to generate the advanced optimized data for the specified library file, SpyGlass does not generate the .sglib file and reports a fatal violation indicating the failure reason.

If any .sglib file used in the SpyGlass power estimation flow does not contain optimized data, that is, it was not compiled by using the `include_opt_data` command, SpyGlass reports an informational message, using the *CMD_sglib04* rule, recommending the use of this command for better performance. For more information on the *CMD_sglib04* rule, refer to the *SpyGlass Built-in Rules Reference Guide*.

ignore_builtin_rule

Usage: `set_option ignore_builtin_rules <yes | no>`

For details on this command, see *Ignore SpyGlass BuiltIn Rules*.

ignore_case_analysis

Usage:	`set_option ignore_case_analysis <yes	no>`

Use this command to ignore set_case_analysis constraint, if specified. The *checkCMD_ignore_case_analysis* rule reports an INFO message, when this command is set to yes and at least one *set_case_anlysis* constraint is specified in the design.

For more information on the *checkCMD_ignore_case_analysis* rule, refer to *SpyGlass Built-in Rules Reference Guide.*

ignoredir <dir-name>

Usage:	`set_option ignoredir { <directory-name> }`

Use this command to ignore all files in a directory and its subdirectories from SpyGlass analysis.

When you ignore a file, the design units described in that file are assumed to be black boxes for the purpose of SpyGlass analysis. For details, please refer *ignorefile*.

ignoredu

Usage:	`set_option ignoredu { <design-unit-names> }`

For details on this command, see *Ignore Design Unit(s)*.

ignore_builtin_spqdir

Usage:	`set_option ignore_builtin_spqdir <path>`

For details on this command, see *Directory Path Containing Ignore BuiltIn Files*.

ignorefile

Usage: `set_option ignorefile <file-name>`

Use this command to ignore a file from SpyGlass analysis.

When you ignore a file, the design units described in that file are assumed to be black boxes for the purpose of SpyGlass analysis.

The following example ignores the design file `myaddlfile.v`:

```
set_option ignorefile myaddlfile.v
```

NOTE: *You can specify a relative or absolute path of a file in the above command.*

When this command is run, Atrenta Console generates the *ignore_summary report*.

SpyGlass does not consider the ignorefile and ignoredu specifications, if they are specified in a source file that is passed to the set_option libhdlf <logical-library-name> <source-files> project file command.

However, SpyGlass considers the ignorefile and ignoredu specifications in the following cases:

- If they are directly specified in a project file
- If they are specified in a source file that is different from the file specified in libhdlf specification

To ignore design units from precompilation, perform the following steps:

- Perform precompilation of design files in a separate SpyGlass run along with the ignorefile and ignoredu commands.
- Use the precompiled dump for further analysis.

If the you ignore a file that includes design units that are referred in another file, SpyGlass may report an `STX_*` error.

For example, consider the following `ignorefile` specification in which the specified file, `file1.vhd`, includes an entity and the architecture of the same entity is defined in another file, `file2.vhd`:

```
set_option ignorefile file1.vhd
```

In this case, the architecture is not ignored and SpyGlass may report an STX_* error.

Using Wildcard while Ignoring Files

The following table shows some examples of using expressions with this command:

Example	Description
`set_option ignorefile {a*}`	Ignores all files (in the current directory) whose names match the wildcard a* expression. For example, a1, aa1, and abc.
`set_option ignorefile {dir1/*}`	Ignores all files in the dir1 directory
`set_option ignorefile {dir?/*}`	Ignores all files in directories that match the wildcard expression dir?. For example, dir1, dir2, and dir3.

If your filename includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your filename is "abc*d", you need to refer to it as {{abc*d}}.
However, if you want to refer to two files, for example "abc1d" and "abc2d", you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see *Pattern Matching Across Features.*

ignorelibs

Usage:

```
set_option ignorelibs <yes | no>
set_goal_option ignorelibs <yes | no>
```

Use this command to skip rule-checking on modules in library files specified by using the `v` or `y` commands.

When you set this command to `yes`, SpyGlass does not report violations (except ELAB/SYNTH/InfoAnalyzeBBox/WarnAnalyzeBBox/ErrorAnalyzeBBox/FatalAnalyzeBBox errors) on these modules.

However, functional model of these modules is synthesized/flattened and is available during any checks performed on other modules.

NOTE: *If any part of incremental schematic for a rule violation lies outside an IP boundary (module passed as a library file), SpyGlass does not waive off the violation of that rule even with the* `ignorelibs` *command.*

ignore_reference_project_sgdc

Usage:

```
set_goal_option ignore_reference_project_sgdc <yes | no>
```

Set this command to `yes` to ignore the SGDC files specified in the project file specified by the *reference_design_projectfile* command.

ignorerules

Usage:

```
set_option ignorerules { rule-names }
set_goal_option ignorerules { rule-names }
```

Use this command to specify name of rules or rule groups to be ignored in the current SpyGlass run. You can specify multiple rule names, rule group names, or a combination of both.

NOTE: *Rule and rule group names are case sensitive. You can specify multiple rules to ignore as space-separated list. The comma-separated list is not allowed.*

NOTE: *There are certain rules in SpyGlass that cannot be ignored. If you specify such rules with the* `ignorerules` *command, SpyGlass reports WARNING [38]. For example, most of the Built-In and prerequisite rules cannot be ignored.*

NOTE: *You can ignore semantic checks for SpyGlass classic-batch options and SGDC commands, even though they are mandatory checks. Some of these checks are CMD_define_severity03, checkCMD_wildcardMatch03, SGDC_clock01, and checkSGDC_existence. Do not ignore these checks unless you encounter any problems with them.*

Effect of the RULE_SELECTION_ON_CL_POSITION Command

`RULE_SELECTION_ON_CL_POSITION` is the command of the .spyglass.setup file. Based on the value of this command, the tool decides the priority of the `ignorerules` and `rules` commands.

The `RULE_SELECTION_ON_CL_POSITION` command accepts any of the following values:

- `yes`

  In this case, the `rules` command is given higher priority over the `ignorerules` command. Consider the following example:

  ```
  set_goal_option ignorerules R1
  set_goal_option rules R1
  ```

  In this example, the `R1` rule will not be ignored and will run during SpyGlass analysis.

- `no` (default)

  In this case, the `ignorerules` command is given higher priority over the `rules` command. Therefore, in the above example, the `R1` rule will be ignored and it will not run during SpyGlass analysis.

Example of using the ignorerules Command

Following are some examples:

- The following command ignores the `R1` and `R2` rules from SpyGlass analysis:

  ```
  set_option ignorerules {R1 R2}
  ```

- The following example ignores the `R1` rule of the `grp1` rule group:

```
set_goal_option rules grp1
set_goal_option ignorerules R1
```

In the above example, except the `R1` rule, all rules of the `grp1` group are run.

ignorewaivers

Usage:
```
set_option ignorewaivers <yes | no>
set_goal_option ignorewaivers <yes | no>
```

Use this command to ignore waivers supplied as embedded SpyGlass waiver pragmas.

ignore_undefined_rules

Usage:
```
set_goal_option ignore_undefined_rules <yes | no>
```

Use this command to continue SpyGlass analysis after reporting a warning message if an undefined rule is specified.

By default, SpyGlass exits with an error if you specify a rule that does not exist in any of the specified products.

incdir

Usage:
```
set_option incdir <directory-path>
```

For details on this command, see *Searches the specified paths for include files*.

inferblackbox

Usage: `set_option inferblackbox <yes | no>`

Use this command to infer black box module interface based on black box instances in the synthesized netlist and write to the sgBlackbox.v file in the current output directory.

For details, refer to the *Inferring Black Boxes* topic of *Atrenta Console User Guide*.

inferblackbox_iterations

Usage: `set_option inferblackbox_iterations <int-value>`

Use this command to specify an effort in terms of the total number of iterations that SpyGlass should make before finalizing port directions for black boxes.

In a single iteration, SpyGlass may not be able to assign a definite port direction to a black box based on connectivity with non black box instances. To achieve a definite port direction, few iterations are required.

Typically, three to four iterations are enough to converge on a good estimate of port direction.

NOTE: *You can specify the name of this command as inferblackbox_iterations or inferblackbox_iteration.*

infer_enabled_flop

Usage: `set_option infer_enabled_flop <on | off>`

Use this command to enable synthesis to create flip-flops without an enable signal.

By default, the value of this command is on. In this case, synthesis creates flip-flops with enable.

Set the value of this command to off to enable synthesis to create flip-flops without enable together with a MUX to get enable functionality.

ipdbdir

Usage: `set_option ipdbdir {list of db’s}`

Use this command to specify the list of db’s wherein each individual db corresponds to an individual IP block. Depending on the impact of change across save-restore run, the netlist modules are either restored from their respective databases or synthesized afresh incrementally in the restore run.

For more information on this option, see *Restoring From Multiple Databases*.

language_mode

Usage: `set_option language_mode <value>`

Use this command to specify the operating language for the current SpyGlass run. The allowed values are:

- `verilog`

  Specifies that the language mode as Verilog. In this case, you can process only Verilog design files.
- `vhdl`

  Specifies that the language mode as Verilog. In this case, you can process only VHDL design files.
- `mixed`

  Specifies that the language mode as mixed. In this case, you can process either Verilog designs, VHDL designs, or mixed language designs.

For details on specifying a language mode through GUI, see *Language Mode*.

lib

Usage: `set_option lib <logical-lib-name> <physical-path>`

Use this command to define a mapping between the logical name of a library and the actual physical location where the compiled library is present.

By default, SpyGlass considers the WORK directory (in the current working directory) as the path of the working library. The physical location must be an existing directory.

If you map a single logical library to multiple physical locations, the last specified location is used.

If libraries are completely debugged, there should be no need to run SpyGlass rule checks on them and the `norules` option should be used.

Examples of Using the lib Command

Following are some examples:

- The following command maps the `alu` library to the physical directory mylibs:

  ```
  set_option lib { "alu" "~libs/mylibs" }
  ```

- The following command maps the `NEW` and `MATH` libraries to their physical VHDL files

  ```
  set_option lib { "NEW" "~libs/new" }
  set_option lib { "MATH" "~libs/math" }
  ```

Handling of Incorrect Library Path Specifications

In case of incorrect library path specifications, SpyGlass behavior is as follows:

- If a library path is not specified and the library is used only in the `use` clause and not used in any design unit, SpyGlass generates a warning message and continues. If the library is being used in a design unit, SpyGlass generates additional error message and may abort depending on the criticality of library usage.

- If the library path does not exist, SpyGlass generates an error message and aborts before analysis. This is irrespective of whether the specified library is used or not.

NOTE: *The 32-bit version of user-compiled libraries are created in a sub-directory named* 32 *under the specified working directory. The 64-bit version of user-compiled libraries is created in a sub-directory named* 64 *under the specified working directory.*

Precompiled VHDL Libraries

The SpyGlass VHDL environment comes with the following precompiled libraries:

- IEEE
- STD
- SYNOPSYS

By default, these libraries are visible SpyGlass.

libhdl_extmap

Usage: `set_option libhdl_extmap <file-extension> <file-type>`

Use this command to provide mapping from extension-name to the compilation language in a project file.

This option is same as specifying the LIBHDL_EXTMAP key in the SpyGlass Configuration file. If libhdl_extmap is specified in a project file and the LIBHDL_EXTMAP key is specified in the SpyGlass Configuration file, the extension map from both will be unified for the run.

Examples for existing LIBHDL_EXTMAP in SpyGlass config file, as mentioned in Console guide:

Consider the following keys specified in the SpyGlass Configuration file:

```
LIBHDL_EXTMAP = .v   verilog
LIBHDL_EXTMAP = .v2K  verilog2000
LIBHDL_EXTMAP = .sv  systemverilog
LIBHDL_EXTMAP = .vh87  vhdl87
LIBHDL_EXTMAP = .vh93  vhdl93
```

The corresponding project file commands for the above keys is as follows:

```
set_option libhdl_extmap .v  verilog2000
set_option libhdl_extmap .v2K  verilog2000
set_option libhdl_extmap .sv  systemverilog
set_option libhdl_extmap .vh87  vhdl87
set_option libhdl_extmap .vh93  vhdl93
```

libmap

Usage: `set_option libmap <logical-lib-name> <intermediate-lib-name>`

Use this command to specify mapping between logical libraries and an intermediate library.

libext

Usage: `set_option libext <text>`

For details on this command, see *Specify library file extensions*.

library_gen_clock_naming

Usage: `set_parameter library_gen_clock_naming <yes | no>`

Use this command to switch the naming conventions for the clock names.

By default, the value of this parameter is set no. In this case, SpyGlass generates clock names as per the Liberty Reference Manual. Following is a sample clock naming scheme in SpyGlass when the value of this parameter is no:

```
cell (namestring {
  generated_clock (namestring) { // This is the name of the 
generated clock by default
  ...clock data...
   }
}
```

You can set the value of the parameter to yes to generate the clock names in accordance with other industry tools. That is, the hierarchical name in the instance name or pin name format.

LICENSEDEBUG

Usage: `set_option LICENSEDEBUG <yes | no>`

Use this command to print the following license debug information on the screen and in the SpyGlass log file, spyglass.log, in the given order:

- Values of `ATRENTA_LICENSE_FILE` and `LM_LICENSE_FILE`

  The host IP set in these variables is used to search for licenses.

NOTE: `ATRENTA_LICENSE_FILE` *is given preference over* `LM_LICENSE_FILE`*.*

- License checkout start message followed by end message for each license checked out

  Here, the time elapsed during the checkout process is also printed. This information is useful if the license checkout time needs to be benchmarked.

  In addition, the checkout server name (and server host id) is also printed. This information helps in identifying the servers from where various features have been checked out. On reviewing this data, you may want to rearrange the settings in the `ATRENTA_LICENSE_FILE`/ `LM_LICENSE_FILE` variable to get the features checkout from the nearest server(s), if the license servers are geographically distributed, for quick turnaround time.

- Summary for licenses that are checked out when products are being loaded in the initial stages of SpyGlass run. This information is printed in the following order:

  Total number of features successfully checked out

  Total number of denied licensing calls

  Total time between first and last licensing call

  Total time elapsed in successful licensing calls

  Total time elapsed in failed licensing calls

  Total time elapsed in licensing calls

  Here, some rules also checkout few licenses when they run. The summary for such license checkouts will not be printed. However, license checkout start and end messages are printed for these licenses.

All the license debug information starts with ##LICENSEDEBUG: to aid in quick scanning of license related debugging information.

To stop creation of license jobs for the license files/servers, specified with LM_LICENSE_FILE, specify the following option in the .spyglass.setup file.

```
IGNORE_LM_LICENSE_FILE = yes
```

lvpr

Usage:

```
set_option lvpr <rule-name>=<num>
set_option lvpr <num>
set_goal_option lvpr <rule-name>=<num>
set_goal_option lvpr <num>
```

Where:

- <rule-name> is the name of the rule for which you want to limit the number of messages.
- <num> is the maximum number of messages to be reported.

Use this command to specify the maximum number of messages to be reported for a particular rule. This way, you can limit the number of repetitions of individual rule messages, which makes it easier to focus on other messages.

This command is useful when errors are difficult to identify because some rules report multiple messages that are similar in nature. For example, multiple rules may report a name that is frequently used in a design and that name violates the defined naming convention.

Examples of using the lvpr Command

Following are the examples:

- The following command limits the number of messages to 20 for the *SigName* rule of SpyGlass OpenMore solution but keeps an infinite limit for all other rules checked in a Verilog design file:

  ```
  set_goal_option lvpr SigName=20
  ```

- The following command limits the number of messages reported for any rule of the product being used to 30 in a VHDL design file:

  ```
  set_option language_mode vhdl
  set_goal_option lvpr 30
  ```

- The following command limits the number of messages to 30 for the *SigName* rule of SpyGlass OpenMore solution, and limits the number of all the other messages of this product to 20:

  ```
  set_goal_option lvpr 20
  set_goal_option lvpr SigName=30
  ```

NOTE: *The maximum limit per rule (30 for the SigName rule in this case) overrides the overall limit set (20 in this case).*

- The following command limits the number of messages to 10 for the *SigName* rule 20 for the *ClkName* rule:

  ```
  set_goal_option lvpr SigName=10
  set_goal_option lvpr ClkName=20
  ```

- The following command reports all messages of all rules:

  ```
  set_goal_option lvpr -1
  ```

macro_synthesis_off

Usage: `set_option macro_synthesis_off <yes | no>`

Use this command to disable the `SYNTHESIS` macro. By default, SpyGlass includes the `SYNTHESIS` macro.

mapSuffixList

Usage:

```
set_option mapSuffixList <list>
set_goal_option mapSuffixList <list>
```

For details on this command, see *Specify parameter to give the list of suffix strings*.

mapPrefixList

Usage:

```
set_option mapPrefixList <list>
set_goal_option mapPrefixList <list>
```

For details on this command, see *Specify parameter to give the list of prefix strings*.

mapVirtualClkByName

Usage:

```
set_option mapVirtualClkByName <yes | no>
set_goal_option mapVirtualClkByName <yes | no>
```

For details on this command, see *Specify the manner in which virtual-to-real clock mapping to be done*.

mthresh

Usage:

```
set_option mthresh <num>
```

For details on this command, see *Upper Threshold for Compiling Memories*.

net_osc_count_limit

Usage:

```
set_option net_osc_count_limit <num>
```

Use this command to specify the number of oscillations allowed to get a stable value on a particular net within SpyGlass logic evaluator.

By default, the limit for the oscillation count for any net is 100. You can override this default value by using the net_osc_count_limit command, as shown in the following example:

```
set_option net_osc_count_limit 10
```

netlist

Usage: `set_option netlist <on | off>`

Use this command to specify a design as a netlist design to enable various run-time optimizations.

Ensure that this option is specified only for the netlist design, such as HDL source files, if this option is switched on. An HDL source means that the source files, -v, and -y option should be netlist. However, if you specify a non-netlist or RTL design, when this option is switched on, SpyGlass behavior may be undefined.

When the netlist option is set to on:

- Elab_summary report is not generated
- UDPs are not supported

Note that you cannot perform the following tasks using this command:

- Enable module by module deletion by using the enable_mmdelete or higher_capacity command.
- Generate a precompile dump by using the enable_precompile_vlog command or single step precompile feature by using the libhdlf or libhdlfiles commands.

netlist_clock_polarity

Usage: `set_option netlist_clock_polarity <yes | no>`

Use this parameter to improve the calculation of number of inversions inside the library cells.

By default the value of this parameter is set to yes. In this case, the waveform of the generated clocks is shifted appropriately, depending on whether odd number of inversions are found.

Set the value of this parameter to no to ignore the waveform of the generated clocks.

nobb

Usage: `set_option nobb <yes | no>`

For details on this command, see *Exit on Detecting Blackboxes in the Design*.

noelab

Usage: `set_option noelab <yes | no>`

Set this command to `yes` to exit after design analysis without elaborating the design. In this case, only built-in rules are checked on the design.

Set this command to `yes` when you need to compile only Verilog/VHDL files without rule-checking.

NOTE: *Do not set the* `noelab` *command to* `yes` *when the* `top` *command is also specified. If you specify these commands together, SpyGlass exits and reports an error.*

nodefparam

Usage: `set_option nodefparam <yes | no>`

Use this command to ignore explicit parameter re-definition given by a `defparam` Verilog construct.

Verilog parameters can be redefined within a module instance by using `defparam` statements, as shown in the following example:

```
module ram (...);
  parameter WIDTH = 8;
  parameter SIZE = 256;
  ...
endmodule
module my_chip (...);
  ...
  //Explicit parameter redefinition by name
  RAM ram1 (...);
  defparam ram1.SIZE = 1023;
  ...
endmodule
```

If you want to disable these parameter re-definitions and want to use original parameter values, use the `nodefparam` command.

nopreserve

Usage: `set_option nopreserve <yes | no>`

Use this command to remove hanging/unconnected instances and nets.

SpyGlass RTL synthesis engine usually retains hanging nets (open-ended interconnections among logic gates) and hanging/unconnected instances. If you want to remove such instances and nets from your design so that the result matches with that of your main synthesis engine, specify this command.

Preserving all instances and nets makes it easier to relate inferred logic back to your source code.

noreport

Usage:

```
set_option noreport <yes | no>
set_goal_option noreport <yes | no>
```

Use this command to suppress report generation.

Use this command if you want to analyze results separately (for example in the SpyGlass GUI).

If you do not specify a report format, SpyGlass uses the moresimple report, which is the default SpyGlass report.

Sometimes you might not need any extensive report. For example, you might want only a quick status check of your design to make sure no errors remain, or you might prefer to analyze errors using the SpyGlass GUI. In such cases, use this command to suppress all reporting.

Example of using the noreport Command

The following command suppresses report generation for the Verilog file test1.v:

```
set_option language_mode verilog
read_file -type verilog test1.v
set_option noreport yes
```

norules

Usage:

```
set_goal_option norules <yes | no>
```

Use this command to suppress rule-checking during SpyGlass analysis.

When you set this command to `yes`, SpyGlass analyzes a design for syntax errors and warnings only. In addition, SpyGlass reports only RTL description-level built-in messages.

You should set this command to `yes` to check that:

- All files, libraries, include files, macro definition files, etc. are all present and correctly defined before any rules are checked.
- VHDL libraries are precompiled and checked for HDL syntax errors before any rules are checked.

nosavepolicy/nosavepolicies

Usage:

```
set_option nosavepolicy <product-name>
set_option nosavepolicies <product-list>
```

Use these commands to specify a product (`nosavepolicy`) or a list of products (`nosavepolicies`) that should not be saved during design save.

Under Design Save-Restore feature, you specify products during design save and run any subset of these products during design restore. Entire base polices and the four advanced products, that is, SpyGlass Constraints solution, SpyGlass DFT solution, SpyGlass Power Verify solution, and SpyGlass TXV solution, are saved by default. In addition, the design view related to the products being used, which are specified indirectly through goals is also saved. If you do not want to save certain products, specify such products by using the `nosavepolicy/nosavepolicies` command.

Prerequisite

The *nosavepolicies* command works only when the *enable_save_restore* command is set to `yes`.

Exception

The *nosavepolicies* command is ignored during design restore mode.

Keywords supported by nosavepolicy/nosavepolicies

You can specify the following keywords with `nosavepolicy/ nosavepolicies`:

Keyword	Products Covered by the Corresponding Keyword			
basepolicy	timing	starcad-21	starc2005	starc2002
	starc	simulation	openmore	morelint
	miscellaneous	lint	latch	erc
	area	Audits		
advpolicy	txv	lowpower	power_est	dft
	dft_dsm	const_inter n1	constraints	clock-reset
all	Union of basepolicy and advpolicy			

To check for the product names to be used as a keyword for this command, refer to product name directory in your installation directory.

To disable the SpyGlass Constraints product, also specify the const_intern1 and Spyglass Txv product with the nosavepolicy/nosavepolicies option as shown below:

```
set_option nosavepolicy { constraints const_intern1 txv }
```

Similarly, to disable the SpyGlass DFT product, also specify the SpyGlass DFT DSM product with the nosavepolicy/nosavepolicies option as shown below:

```
set_option nosavepolicy { dft dft_dsm }
```

However, if you need to disable the SpyGlass Txv product or SpyGlass DFT DSM product, you need not specify the SpyGlass Constraints or SpyGlass DFT product, respectively, with the nosavepolicy/nosavepolicies.

For example, to discard the design data for the lint product, specify the lint keyword as mentioned in the <your-inst-dir>/SPYGLASS_HOME/policies/lint directory, with the `set_option nosavepolicy` command.

Also, consider the following examples:

```
set_option nosavepolicy dft
```

```
set_option nosavepolicy {basepolicy dft}
```

The first example listed above discards the design data for the SpyGlass DFT product. In the second example, the SpyGlass DFT product and all the products covered under the basepolicy keyword are not saved during design save.

NOTE: *Specify the 'advcdc' keyword along with the nosavepolicy option to prevent the checkout of adv-cdc product license, which otherwise gets checked out implicitly if an Advanced CDC rule is running.*

To save a product during design save, use the *savepolicy/savepolicies* command.

overload

Usage: `set_goal_option overload <named-overload>`

Use this command to runs the specified named overloads for all the specified products.

For details on using named overloads, refer to the *SpyGlass Policy Customization Guide*.

overloadpolicy

Usage: `set_goal_option overloadpolicy <product-name>`

Use this command to run the specified products with overloaded components, if any.

NOTE: *Rule-checking results with overloaded products may be different from results from normal products.*

If you do not use this command, overloaded product components, if any, are ignored and normal products are run.

Need to the overloadpolicy Command

This command is used to overload a product to meet local preferences of a user base. It is typically required when you want to use Atrenta-standard products but with certain customization, such as different severity-labels, pod-cut description, and alias name.

Prerequisites for Using the overloadpolicy Command

To use this command, create a file named <product-name>-policy-overload.pl where *<product-name>* is the name of the original product you want to overload. For example, to overload SpyGlass lint solution, create the file lint-policy-overload.pl file.

NOTE: *For information on creating product overload files, refer to the SpyGlass Policy Customization Guide.*

Product overload files are searched using the same *-I* command-line option mechanism as used for custom products.

This command does not work, if the overloaded product and the directory containing this product have the same name. Use a different name than the name of the overloaded product for the directory containing this product.

Using the overloadpolicy Command

Based on the value specified to this command, SpyGlass performs different actions, as discussed in the following table:

Specification	Action
`set_goal_option overloadpolicy`	Apply product overload on all product being selected to run. This is subject to product overload file being found in the specified search path.
`set_goal_option overloadpolicy none` (Or not specified)	Disables search and loading of overload files

Specification	Action
`set_goal_option overloadpolicy a,b`	Apply product overload to the a and b products only, assuming one or more of these product are selected to run.
`set_goal_option overloadpolicy all`	Apply product overload on all products selected to run. If product overload file is not found for any product, it is considered as an error condition, and (frequently) may be because of incorrect or no *-I* specification.

overloadrules

Usage:

```
set_option overloadrules <rule-name>+<
lang>+severity=<severity-label>+weight=<value>

set_goal_option overloadrules <rule-name>+<
lang>+severity=<severity-label>+weight=<value>
```

Note that all settings and the intervening + characters must be specified without any spaces.

Use this command to overload the severity or weight of a rule for the current SpyGlass run.

NOTE: *The* `overloadrules` *command is used in conjunction with spyOverload specification. However, the* `overloadrules` *command takes precedence over the spyOverload specification.*

NOTE: *The* `overloadrules` *command is not applicable for rule aliases.*

Arguments of this command are discussed below:

- *<rule-name>* is the name of the rule being overloaded.
- *<lang>* is the language for which you want to overload a rule. The allowed values are `VHDL`, `Verilog`, `Verilog+VHDL` (case-insensitive).

  Language specified with the `overloadrules` command should be same as used in the corresponding rule registration. For example, if a rule is registered with `Verilog + VHDL` language, its `overloadrules` specification should also be `Verilog + VHDL`.

  NOTE: *If a rule is registered in a single language (Verilog or VHDL), and there is no* `overloadrules` *specification in the same language (through* `overloadrules` *or through spyOverload), the* `overloadrules` *command with Verilog+VHDL is still applied for backward compatibility purposes. However, this is not a recommended use-model and should not be used.*

  The *<lang>* setting is optional. If you do not specify a language, the overload values are applicable to all language variations of the rule. Therefore, if you are not sure about the language specification of rule registration, you can omit the language option from `overloadrules` specification.

- *<severity-label>* is the overloaded severity label. The severity label can be one of the SpyGlass predefined severity labels (`Fatal`, `Error`, `Warning`, `Info`, and `Data`), one of the severity labels defined in the corresponding product, or a user-specified severity label defined using the `define_severity` command. If the specified severity label is not any of these labels, SpyGlass auto-registers the severity label under the severity class `ERROR`.

NOTE: *As you overload a rule's severity label, the SpyGlass processing of the rules will be governed by the severity class of the overloaded severity label. For example, when you overload a rule with a severity label of severity class FATAL, SpyGlass aborts when a rule-violation of this rule is encountered.*

- *<value>* is the overloaded rule weight value.

You can specify the *<rule-name>* followed by any combination of the other settings.

Example of using the overloadrules Command

Following are some examples:

- The following example indicates that the severity label for the Verilog version of the W402b rule (of SpyGlass lint solution) should be `Warning` (one of the SpyGlass predefined severity labels) for the current SpyGlass run:

```
set_option overloadrules W402b+Verilog+severity=Warning
```

- The following example indicates that the severity label for all language versions of the Clk_Gen03a rule (of SpyGlass Constraints solution) should be `Error` (one of the SpyGlass predefined severity labels) for the current SpyGlass run:

```
set_option overloadrules Clk_Gen03a+severity=Error
```

- The following example indicates that the rule weight for the Verilog version of the InstNameLength rule (of SpyGlass OpenMore solution) should be 10 (the default weight is 5) for the current SpyGlass run:

```
set_option overloadrules InstNameLength+Verilog+weight=10
```

- The following example indicates that the severity label for the VHDL version of the W116 rule (of SpyGlass lint solution) should be myERROR1 (defined using the define_severity command) under the severity class ERROR for the current SpyGlass run:

  ```
  set_goal_option define_severity lint+myERROR1+ERROR
  set_goal_option overloadrules W116+VHDL+severity=myERROR1
  ```

- The following example indicates that the severity label for the TA_01 rule (of SpyGlass DFT solution) should be Info (one of the SpyGlass predefined severity labels) and the rule weight should be 10 (the default weight is 1) for the current SpyGlass run:

  ```
  set_option overloadrules TA_01+severity=Info+weight=10
  ```

Specifying overloadrules Specification for a Particular Label

You can also specify overloadrule specification for a particular label. For example, consider a multi message rule, *dummyRule*, with the following assumptions:

- The rule that has two labels, Label1 and Label2.
- Severity for Label1 is Warning and severity for Label2 is Info.

Now if you want to change the severities of both the labels to Error, use the following overloadrules specification:

```
set_option overloadrules dummyRule+severity=Error
```

To change the severity of a specific label (say Label1), you must also specify the language for which you want to overload, unless label is present for all the personalities of the rule:

```
set_option overloadrules
dummyRule+severity=Error+msgLabel=LABEL1+Verilog
```

Consider the following message labels for the W123 rule for both Verilog and VHDL:

Verilog

- veCheckUsage_Viol_Msg
- SignalUsageReport_Refer
- veCheckArrayOutBound_Info

VHDL

vhLintArray_Viol_Msg

SignalUsageReport_Refer

If you want to overload a specific message label, you must also specify the language for which you want to overload as shown in the following example:

```
set_option overloadrules
W123+severity=Violation+msgLabel=veCheckUsage_Viol_Msg+Veril
og
```

If a message label is present for all the variants of the rule, both Verilog and VHDL, in that case you do not need to mention the language while overloading this label specifically. Following example demonstrates this condition:

```
overloadrules=W123+severity=Info+msgLabel=SignalUsageReport_
Refer
```

overload_rule_category

Usage:	`set_goal_option overload_rule_category <rule_name>+rulecategory=<rule_category>`

Use this command to overload the rule category with a user-specified category.

Overloaded rule categories are included in the moresimple report after run_goal. The overload feature is available only if the `enable_rule_category_in_moresimple option` is enabled for the design. If the `enable_rule_category_in_moresimple` is enabled, all the violations in moresimple report are sorted by severity and then by rule category.

ovl_verilog

Usage:	`set_option ovl_verilog { OVL-file> }`

Use this command to specify an Open Verification Library (OVL) file written in Verilog. This file is used for functional analysis.

ovl_vhdl

Usage:
```
set_option ovl_vhdl { OVL-file> }
```

Use this command to specify an OVL file written in VHDL. This file is used for functional analysis.

param

Usage:
```
set_option param <name>
```

Use this command to specify parameters used during Verilog/VHDL analysis run.

perflog

Usage:
```
set_option perflog <yes | no>
set_goal_option perflog <yes | no>
```

Use this command to generate SpyGlass performance log.

This performance log is written in the spyglass.perflog file in the current working directory. This file contains memory/runtime details of each rule run. In addition, the performance log also contains benchmark data for various stages of SpyGlass run, such as analysis, synthesis, and flattening.

physical

Usage:
```
set_option physical <true | false>
```

Use this command to enable physical library compilation of Synopsys Liberty files (.lib files).

SpyGlass Library Compiler provides the feature to perform physical library compilation of Synopsys Liberty files (.lib files) and other physical related inputs, such as technology LEF files. This is equivalent to the library preparation step for SpyGlass Physical.

This option is meaningful only when automatic compilation of gateslib is enabled and Synopsys Liberty files (.lib files) are provided.

physical_dbdir

Usage: `set_option physical_dbdir <directory_path>`

Use this command to specify the design database directory path for save and restore runs in Physical aware power estimation flow. This switch is used in `physical_power_postfloorplan` and `power_est_average` goals.

By default, if you do not specify a directory path, SpyGlass uses default location for design database save and restore runs.

physical_lib_config_file

Usage: `set_option physical_lib_config_file {<config-file-path>}`

Use this command to specify the path of a configuration file for SpyGlass Physical library preparation through physical library compilation.

This option is meaningful only when you do the following:

- Enable physical library compilation by using the following command:

  set_option *physical* true

- Enable automatic compilation of gateslib.

Using this command enables you to extend SpyGlass Physical library compilation for advanced capabilities.

For information on the semantics of this configuration file, refer to

SpyGlass Physical documentation.

The following example shows the usage of this command:

```
set_option physical_lib_config_file {./
my_libprep_config_file}
```

physical_target

Usage: `set_option physical_target {space-separated-target-list>}`

Use this command to specify a list of library targets for SpyGlass Physical library preparation through physical library compilation.

This option is meaningful only when you do the following:

- Enable physical library compilation by using the following command:

  `set_option` *physical* `true`

- Enable automatic compilation of gateslib.

The following example shows the usage of this command:

```
set_option physical_target {target1 target2 target3}
```

Alternatively, you can specify a list of library targets by specifying the list to the SpyGlass Physical configuration file. You can then specify this file to the *physical_lib_config_file* command.

For information on the semantics of this configuration file, refer to SpyGlass Physical documentation.

portparam

Usage: `set_option portparam { <list-of-space-separated-strings> }`

Use this command to override simple type parameters of top module ports only. The string specified should be in the following format:

```
<module_name>.<port_name>.<param_name>=<param_value>
```

Where:

- <module_name>: Specify only the top module name
- <port_name>: Specify only port name of SV interface type port
- <param_name>: Specify only full name of parameter; do not specify bit select/part select
- <param_value>: Can be an integer or a base number.

Note that there are no spaces between any values in the above format. In addition, a comma separated list of strings is considered as an invalid input.

SpyGlass reports an ELAB_6205 warning with appropriate reason if invalid inputs are specified with the portparam option.

pragma

Usage: `set_option pragma { <list-of-pragmas> }`

For details on this command, see *Interpret Pragma(s)*.

prefer_tech_lib

Usage: `set_option prefer_tech_lib <yes | no>`

Use this command to provide higher preference to technology library definitions present in the .lib/.sglib file over user-specified definitions present in source HDL files, precompiled libraries, and simulation models while searching for the master of an instance.

By default, SpyGlass gives higher priority to user-specified definition.

For example, in the following case, the AN2 definition is picked from .sglib:

```
read_file -type sglib AN2.sglib
read_file -type verilog top.v AN2.v
set_option prefer_tech_lib yes
```

In the following example, the AN2 definition is picked from .lib:

```
read_file -type gateslib AN2.lib
read_file -type vhdl top.vhd AN2.vhd
set_option enable_gateslib_autocompile yes
set_option prefer_tech_lib yes
```

In the following example, the AN2 definition is picked from .sglib even if the library L contains AN2:

```
read_file -type sglib AN2.sglib
read_file -type verilog top.v
set_option lib L P
set_option prefer_tech_lib yes
```

Please note the following points about the `prefer_tech_lib` command:

- Highest priority is given to technology library definitions in the form of the .lib/.sglib file. This priority is above source file specifications and not only HDL libraries models/precompile dump.
- Use the `prefer_tech_lib` command judiciously as there is a time penalty during HDL analysis stage.

  In the absence of this command, whenever there is a duplicate definition in an HDL and technology library, the *IgnoredLibCells* rule reports a violation. In this case, if you want to give higher priority to technology libraries, re-run the design with the `prefer_tech_lib` command.
- Irrespective of whether a functional view exists for a .lib cell definition, SpyGlass gives higher priority to technology library definitions.

  If you want to overwrite or add functional view of .lib cell definition, you can do it in the library compilation stage.
- If you have specified the `ignoredu` command in which the specified design unit is present in both HDL and .sglib/.lib file, the `ignoredu` command is given higher priority. That is, the technology library cell definition is also ignored in this case.
- Presence or absence of the `prefer_tech_lib` command triggers re-synthesis or re-save in the save-restore flow.

- SpyGlass does not report parsing violations on an HDL design unit definition that has been overridden by a technology library definition due to the `prefer_tech_lib` command.
- SpyGlass reports analysis violations, such as *INFO_1006* (Verilog) and *INFO_631* (VHDL) while ignoring a design unit definition because it is also present in technology library input.
- If the `prefer_tech_lib` command is used in an HDL precompile run where the .lib/.sglib files are also passed and duplicate definitions (HDL and .lib/.sglib) exist, SpyGlass does not dump the HDL definitions.

Technology library models are not dumped in the HDL precompile dumps.

preserve_mux

Usage:

```
set_option preserve_mux <yes | no>
set_goal_option preserve_mux <yes | no>
```

Use this command to enable SpyGlass to pick mux cells from a technology library for mapping muxes in a user design.

When you set this command to `yes`, SpyGlass gives priority to the mux cells present in the technology library rather than a mux implementation in terms of more basic cells from the technology library.

If mux cells are not present in the technology library, SpyGlass selects the mux implementation in terms of more basic cells from the technology library.

The `preserve_mux` command can be specified only when the *-est_mode* command is used or EST mode compatible rules are specified.

print_sortorder_only

Usage: `set_option print_sortorder_only <yes | no>`

Use this command to print the list of sorted VHDL files.

NOTE: *The* `print_sortorder_only` *option is used only with the* `sort` *option. Spyglass exits after printing the sorted VHDL files.*

prohibit_waiver

Usage: `set_option prohibit_waiver <rule_names>`

Use this command to prohibit a waiver from waiving the violation messages of specified rules, without editing the waiver commands in the pre-existing waiver files. See *Example 1* for the usage of the *prohibit_waiver* command.

Specify list of rules, whose violation messages you do not want to waive, as an input to this command.

The violation messages for the rule specified in the *prohibit_waiver* command will not be waived, even if you have not specified that rule name in the waive command. See *Example 2* for more information.

Example 1

Consider a waiver file containing the following commands:

```
waive -rule R1
waive -rule R2
```

If you don’t want to waive the violations of the rule R1, then specify the following command in the project file:

```
set_option prohibit_waiver R1
```

Example 2

Consider a waiver file containing the following command:

```
waive -msg M1 -file filename
```

Also consider the following *prohibit_waiver* command specified for the R1 rule:

```
set_option prohibit_waiver R1
```

In the above example, rule name, R1, is not specified in the waiver command, but the violation that is getting waived is for R1. However, this rule is specified in the *prohibit_waiver* constraint command, Therefore, the violation messages for R1 will not get waived.

projectcwd

Usage	`set_option projectcwd <dir-name>`

Use this option to indicate the current working directory when the project file was saved. This option is saved automatically in the project file, when it is saved from SpyGlass Explorer, Atrenta Console, or sg_shell.

This option does not alter tool behavior, but only reports warnings, if any of the files with relative paths are not found.

NOTE: *If the project file contains relative paths, they will always be resolved with respect to the current working directory. These relative paths will not be anchored with respect to the projectcwd option.*

For details on project current working directory, refer to the *Project Current Working Directory* topic in *Atrenta Console User Guide*.

projectwdir

Usage	set_option projectwdir <dir-name>

Use this command to specify a project working directory.

For details on a project working directory, refer to the *Project Working Directory* topic in *Atrenta Console User Guide*.

project_read_only

Usage: `set_option project_read_only <yes | no>`

Use this command to make the project file as read-only and disable overwriting of the project file.

For more details on this command, see *Making Project File Read Only*.

elab_summary_include_localparam

Usage: `set_option elab_summary_include_localparam <yes | no>`

Use this option to avoid including the localparam parameter in the elab_summary report. By default, this option is set to `yes`.

honor_spq_parameter_with_turbo

Usage: `set_option honor_spq_parameter_with_turbo <yes | no>`

Use this option to read parameters from .spq files in turbo mode. By default, this option is set to `no`.

read_file

Usage:

```
read_file -type <file-type> <file-list>
```

Where,

- <file-type> refers to the type of file being read. It can be any of the following:
  verilog, *vhdl*, *def*, *lef*, *hdl*, *gateslib*, *sglib*, *plib*, *sgdc*, *waiver*, *sourcelist*, adc, awl
- <file-list> refers to list of files to be read.

For more information on the Tcl-based usage of the *read_file* command, refer to the *read_file* section of the *SpyGlass Tcl Shell Interface User Guide*.

Use this command to specify the files to be used during SpyGlass analysis.

You can specify any of the following types of files to be read:

def

Specifies Design Exchange Format (DEF) files that are used by SpyGlass Power Verify solution.

These files contain design-specific information of a circuit. For example, these files contain information about instantiated gates in a design. You can specify information about such gates in the Synopsys Liberty™ format library files (.lib files). If you do not specify any information for such gates, SpyGlass considers these gates as black boxes.

You can specify multiple DEF designs in case of hierarchical DEF designs. However, you must:

- Specify these files in a correct compilation sequence.
- Set a top-level design unit by using the *top* command.

NOTE: *You must also specify lef files with this file type.*

NOTE: *Do not mix DEF files with Verilog or VHDL files.*
While analyzing DEF designs, SpyGlass does not accept any other form of RTL, such as VHDL or Verilog design files as input. SpyGlass considers all the specified design files as DEF files. If you specify any Verilog/VHDL file as input, SpyGlass reports an appropriate message.

lef

Specifies Library Exchange Format (LEF) files that are used by SpyGlass Power Verify solution and SpyGlass Power Estimate.

These files provide information about power and ground pins of cells present in library files specified by the *gateslib* or *sglib* commands.

NOTE: *While using SpyGlass Power Verify solution, it is mandatory to specify this file type if you specify the* `set_option enable_pgnetlist` *command. This is required to enable processing of the specified LEF files. However, for SpyGlass Power Estimate, do not specify this file type with the* `set_option enable_pgnetlist` *command.*

vhdl

Specifies VHDL files.

NOTE: *If you specify more than one file, list them in the correct compilation sequence.*

verilog

Specifies Verilog files.

sgdc

Specifies design constraints file (.sgdc). For details on design constraints, refer to the *Working with SpyGlass Design Constraints* topic of *Atrenta Console User Guide*.

waiver

Specifies waiver file containing the `waive` constraints.

The `waive` constraints and SGDC constraints processing is as follows:

File contains	read_file -type waiver <file-name>	read_file -type sgdc <file-name>
Constraints other than `waive` constraint	Other constraints are ignored with an ERROR message	Other constraints are processed normally

File contains	read_file -type waiver <file-name>	read_file -type sgdc <file-name>
Only `waive` constraints	`waive` constraints are processed normally	`waive` constraints are processed with a WARNING message
Both `waive` constraints and other constraints	Other constraints are ignored with an ERROR message. `waive` constraints are processed normally	`waive` constraints are processed with a WARNING message. Other constraints are processed normally.

Files specified by using the `waiver` command may contain `setvar` commands. If you specify more than one waiver file containing `setvar` commands and the `setvar` command in both the waiver files is for the same variable, SpyGlass considers the latter specified waiver file to be used by the RTL pragma (that is, pragma2waiver.swl file).

NOTE: *You can specify the setvar commands to be used in RTL pragmas in waiver files only and not in SGDC files. The setvar commands in SGDC file (specified through sgdc command) will be considered only for the SGDC files and not for RTL pragmas.*

SpyGlass reports a warning message if the `setvar` commands specified in a waiver file are not used in that waiver file or RTL pragmas.

sourcelist

Specifies source files (.spp or .f).

These files are ASCII files that contain the following information in separate lines:

- Path of files to be used in the current SpyGlass run.

NOTE: *SpyGlass expands file names specified with wildcard expressions. For example, file name, such as *.vhd is interpreted at runtime.*

- Moving to the next line

  You can continue a command-line to the next line by using the backslash (\) continuation character as in the following example:

```
...
-y \
./libdir
```

```
+libext+.v
...
```

The above specification is equivalent to the following:

```
...
-y ./libdir
+libext+.v
...
```

- Specifying Comments

  You can also add comments in any of the following formats in source files:

  - ❒ // format
  - ❒ # format
  - ❒ /* */ format

NOTE: *You can specify nested source files.*

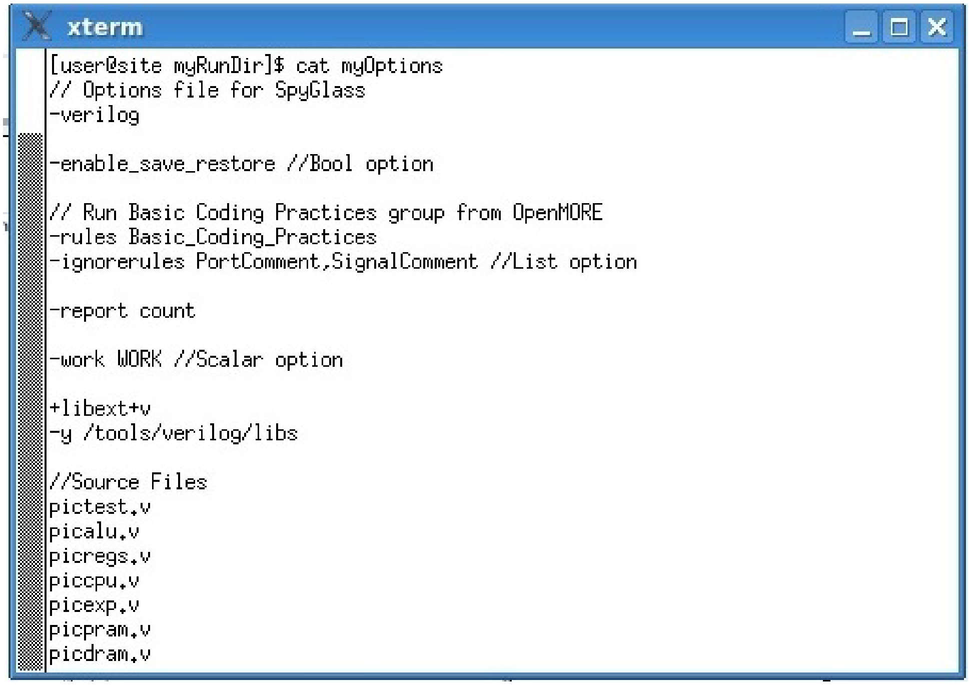
- Type of options

  You can specify the following types of options in a source list file:

TABLE 3 Types of options for Sourcelist File

Option	Label	Syntax
bool	<bool_opt>	''-bool_opt''
scalar/string	<str_opt>	"-str_opt str_opt_val''
list type	<list_opt>	"-list_opt list_opt_val1, list_opt_val2"

Following is an example of a source file:



```
[user@site myRunDir]$ cat myOptions
// Options file for SpyGlass
-verilog

-enable_save_restore //Bool option

// Run Basic Coding Practices group from OpenMORE
-rules Basic_Coding_Practices
-ignorerules PortComment,SignalComment //List option

-report count

-work WORK //Scalar option

+libext+v
-y /tools/verilog/libs

//Source Files
pictest.v
picalu.v
picregs.v
piccpu.v
picexp.v
picpram.v
picdram.v
```

FIGURE 1. Sample Source File

sglib

Specifies SpyGlass-compatible format files (`.sglib` files).

hdl

Specifies HDL files, such as Verilog, VHDL, and DEF files.

gateslib

Specifies library files containing functionality information of gates (cells) instantiated in a design.

plib

Specifies the plib format files.

While specifying these files, you must also specify .lib files and .sglib files using the gateslib or sglib options of the `read_file` command.

In addition, to enable processing of plib files, the `set_option enable_pgnetlist` command.

NOTE: *The* `plib` *option is used by the SpyGlass Power Verify solution only.*

read_protected_envelope

Usage: `set_option read_protected_envelope <yes | no>`

This command enables the parsing and decryption of Protected Envelope in both VHDL and Verilog.

For VHDL envelopes, this command has lower priority than the `set_option sort yes` command. Although, the parsing and decryption in the same design would work for Verilog files without any issues.

reference_design_projectfile

Usage: `set_goal_option reference_design_projectfile <file-name>`

Specifies a project file containing details, such as design files, design options, and SGDC for the reference design while running a DDR (Dual Design Read) goal.

You can use this command instead of the *reference_design_sgdc* and *reference_design_sources* commands. In this case, the project file specified by this command should contain details that you wanted to specify by using the *reference_design_sgdc* and *reference_design_sources* commands.

reference_design_sgdc

Usage: `set_goal_option reference_design_sgdc <file-name>`

Use this command to specify an SGDC file that contains constraints for the reference design in the Dual Design Read (DDR) flow.

reference_design_sources

Usage: `set_goal_option reference_design_sources <file-name>`

Use this command to specify a source file (.f file) that specifies design files and options for a reference design used in the DDR flow.

This command is used while running a DDR goal where both reference and implement designs need to be loaded together.

relax_hdl_parsing

Usage: `set_option relax_hdl_parsing <yes | no>`

Set this command to yes to enable SpyGlass perform VHDL semantic checking in the following manner:

1. By automatically inferring missing 'library' clauses for user-defined libraries, provided correct library mapping is specified.

   Consider the following example given in two steps:

   Step 1:

   Compile the following code into a user-library, `userlib1`:

   ```
   entity an2 is
     port (
       A   : in  bit;
       B   : in  bit;
       Y   : out bit);
   end an2;

   architecture behav of an2 is
   begin
     Y <= A and B;
   end behav;
   ```

   Step 2:

Compile the following code using the user-defined library, `userlib1`, and set the `relax_hdl_parsing` command to `yes`:

```
--library userlib1; -- if -relax_hdl_parsing switch --is
used this declaration of library is not needed.

entity top is
  port (
    topA : in  bit;
    topB : in  bit;
    topZ : out bit);
end top;

architecture struct of top is
  component an2
  port (
    A   : in  bit;
    B   : in  bit;
    Y   : out bit);
  end component;

for I1 : an2 use entity userlib1.an2(behav);
  begin
  I1 : an2
    port map (topA,
      topB,
      topZ);
end struct;
```

If you do not set the `relax_hdl_parsing` command to `yes` in the above case, SpyGlass reports the STX_VH_11 error.

2. Relaxes STX_VH_455 (OTHERS must be the only choice in aggregate of non-locally static size)

   For example:

```
entity E is
end;
architecture A of E is
  function ff(a: bit_vector) return integer is
    type mytype is array(a'range) of bit;
```

```
        variable j: mytype;
        begin
          j := (4=>'1', 5=>'0', others=>'1');
          return a'length;
        end;
      begin
    end A;
```

remove_file

Usage:

```
remove_file -type <file-type>
remove_file -type sgdc [<file-list>]
```

For more information on the Tcl-based usage of the *remove_file* command, refer to the *remove_file* section of the *SpyGlass Tcl Shell Interface User Guide*.

Use this command to remove the files that have been added using the *read_file* command. Except for the SGDC type of file, the remove_file command removes all the files of a given type from the project, and it is not context-sensitive.

Following table lists the file types, which you can specify to remove:

Type	Description
hdl	Removes all HDL files (Verilog, VHDL, or DEF). **NOTE**: You cannot remove individual HDL file types, such as Verilog or VHDL files. The complete HDL file set is removed.
gateslib	Removes gateslib files
sglib	Removes sglib files
lef	Removes LEF files
plib	Removes plib files
sgdc	Removes all or given SGDC files
waiver	Removes waiver files

remove_work

Usage:

```
set_option remove_work <yes | no>
```

Use this command to delete contents of the WORK directory (based on the `work` command) and recompiles all design units in the design.

By default, SpyGlass creates the WORK directory if it does not exist. If the specified directory already exists, SpyGlass recompiles the required design units (all or some) in this directory.

When you set the `remove_work` command to `yes`, SpyGlass removes all contents of the WORK directory and recompiles all design units even if some of the existing precompiled design units did not need re-compilation.

When you save a design view by using the `enable_save_restore` command and set the `remove_work` command to `yes`, you would need to set the `remove_work` command to `yes` during design restore also. However, this command is ignored during the full restore mode (all rules are of Rule Type 1 as described in the *Design Save and Restore Feature* section).

report

Usage:

```
set_option report <report-name>
set_goal_option report <report-name>
```

For details on this command, see *Reports Name*.

report_adjustment_waiver

Usage: `set_option report_adjustment_waiver <yes | no>`

Set this command to `yes` to print waived messages in the Waiver report even when the `-ignore` argument of the `waive` constraint is specified.

By default, only the waived message count is printed in the Waiver report when the `-ignore` argument of the `waive` constraint is specified.

These waived messages are printed in the *Adjustments Waiver Report* section of the Waiver report.

reportfile

Usage:

```
set_option reportfile <name>
set_goal_option reportfile <name>
```

For details on this command, see *Report File*.

report_incr_messages

Usage: `set_option report_incr_messages <yes | no>`

Use this command to enable reporting of incremental messages so that you can compare results of a previously run goal with the current goal.

report_inst_backref

Usage: `set_option report_inst_backref <yes | no>`

Use this command to print back-reference information, such as file name and line number containing definition for a design and its instances in the elab_summary.rpt report.

report_ip_waiver

Usage:

```
set_option report_ip_waiver <yes | no>
set_goal_option report_ip_waiver <yes | no>
```

Use this command to print the waived messages in the waiver report when the -ip argument of the waive constraint is specified. By default, only the count of waived messages is reported in the waiver report.

These waived messages are printed in the *IP/Legacy Waiver Report* section of the waiver report.

report_max_inst

Usage:

```
set_option report_max_inst <value>
```

Use this command to specify a maximum number of instances to be displayed for each design unit in the elab_summary.rpt report.

By default, a maximum of five instances per design unit are displayed in this report. Specify a positive integer value to display the required number of maximum instances per design unit.

For example, if you set report_max_inst to 2, the elab_summary.rpt report shows two instances for each design unit.

If you want to view all instances for each design unit in this report, set report_max_inst to -1.

To specify a negative value, enclose it inside the curly braces as shown in the following example:

```
set_option report_max_inst { -4 }
```

If you specify a negative value other than -1, SpyGlass displays a maximum of five instances per design unit. In addition, a warning message appears in the report to indicate that an incorrect value is specified for this command.

report_max_size

Usage:	`set_option report_max_size <value>`
	`set_goal_option report_max_size <value>`

For details on this command, see *Reports Max Count Size*.

report_per_policy

Usage:	`set_option report_per_policy <yes | no>`

Use this command to split the report specified by the *report* command based on products. In this case, a separate report file (<report-name>_<product-name>.rpt) is generated for each product used in the current run.

The following SpyGlass reports can generate individual report for individual products:

simple	moresimple	inline	waiver	count	summary

Example of using the report_per_policy Command

Consider a scenario in which you are using the SpyGlass lint solution, SpyGlass STARC solution, and a custom product.

Now consider that you specify the following project file commands:

```
set_option report { "simple" }
set_option report_per_policy yes
```

In this case, the following report files are generated:

simple_lint.rpt	simple_starc.rpt	simple_custom.rpt	simple_spyglass.rpt

Here, the first three files are the reports specific to SpyGlass lint solution, SpyGlass STARC solution, and custom product. The last file is the report for built-in messages.

You can use the *reportfile* command to change the name of the generated report files. For example, consider the following project file commands:

```
set_option report { "simple" }
set_option report_per_policy yes
set_option reportfile mysimple
```

In this case, the following report files are generated:

mysimple_lint.rpt	mysimple_starc.rpt	mysimple_custom.rpt
mysimple_spyglass.rpt		

NOTE: *The* `report_per_policy` *command has no impact on product-specific reports.*

Special Case: Inline Report

By default, the inline report generates the following files and directories in the current working directory:

- The inline.rpt file, which is a concatenation of all design files with messages embedded between HDL lines.
- The inline directory that contains individual inline reports (.rpt files) for each design file.

  For example, if you have specified the test1.v and test2.v source design files, the inline directory contains the test1.v.rpt file (that has test1.v contents with embedded messages) and the test2.v.rpt file (that has test2.v contents with embedded messages).

Consider the following project file commands when you are using the SpyGlass lint solution and SpyGlass STARC solution:

```
set_option report { "inline" }
set_option report_per_policy yes
```

In this case, the following files and directories are generated in the current working directory:

File/Directory	Description
inline_lint.rpt	Contains a concatenation of all design files with messages of SpyGlass lint solution embedded between HDL lines
inline_lint directory	Contains individual inline reports (.rpt files) for each design file with messages of SpyGlass lint solution
inline_starc.rpt	Contains a concatenation of all design files with messages of SpyGlass STARC solution embedded between HDL lines
inline_starc directory	Contains individual inline reports (.rpt files) for each design file with messages of SpyGlass STARC solution
inline_spyglass.rpt	Contains a concatenation of all design files with SpyGlass built-in messages embedded between HDL lines

report_style

Usage:

```
set_option report_max_size <value>
set_goal_option report_max_size <value>
```

For details on this command, see *Report Style*.

report_unreachable_default_case

Usage:

```
set_option report_unreachable_default_case <true|false>
```

The default value of this option is true.

Use this option to suppress the SYNTH_5039 messages reported by SpyGlass Synthesis for scenarios in which case-conditions are complete and case-default statement is unreachable.

resetall

Usage:

```
set_option resetall <yes | no>
```

Use this command to reset the Verilog compiler directive `default_nettype` to language default, which is `wire`.

Currently, other Verilog compiler directives are not reset by this option.

It is useful while analyzing multiple design files where the user does not want to specify this default in each of these files.

NOTE: *This option is used in Verilog and Mixed mode and is ignored when used in VHDL mode.*

rules

Usage: `set_goal_option rules <rule-names>`

Use this command to specify a space-separated list of rules or rule groups to run during SpyGlass analysis.

NOTE: *Rule names are case sensitive.*

NOTE: *If you try to specify a rule that is not included in the product you are using, SpyGlass reports the 'rule not registered' error at runtime.*

NOTE: *If you specify both the* `ignorerules` *or* `rules` *commands, the precedence of a command is controlled by the* `RULE_SELECTION_ON_CL_POSITION` *variable. For details, see Effect of the RULE_SELECTION_ON_CL_POSITION Command.*

The `rules` command option overrides all product rule parameter that enable or disable specific rules. For example, the W546 rule of SpyGlass lint solution is controlled by the `verilint_compat` rule parameter. However, the W546 rule will run when it is specified with the `rules` command (directly or indirectly) irrespective of the `verilint_compat` rule parameter (whether specified or not). Such rule parameters are effective only when you run a complete product without using the `rules` command.

NOTE: *Use of rule parameters may affect the functional behavior of some rules and may determine if those rules should be run or not run. Refer to the corresponding product documentation for details on the rule parameters and the effect of the parameters on some rules.*

Examples of using the rules Command

Following are the examples:

- The following example specifies the R1 rule to run during SpyGlass analysis:

  ```
  set_goal_option rules R1
  ```

- The following example specifies the R1 and R2 rules to run during SpyGlass analysis:

  ```
  set_goal_option rules {"R1 R2"}
  ```

- The following example specifies the grp1 rule group to run all rules of this group during SpyGlass analysis:

  ```
  set_goal_option rules grp1
  ```

sca_on_net

Usage: `set_option sca_on_net <0|1>`

Use this option to apply set_case_analysis constraints on a net when the port/pin name matches the net name. This impacts the logic cone inside the hierarchy, which is driven by the net connected to pin/port.

By default, the value of this command is set to 0. In this case, the set_case_analysis constraint is applied on port/pin.

To apply the set_case_analysis constraint on a net, set the value of the command to 1.

For example, consider the following design:

```
module Bottom(input x, output c);
….
endmodule
module top (input a,b, output c);
Wire x; // net name: x
Bottom I1 (.x(x), .y (c)); // pin name: x
……
```

```
endmodule
```

In the above example, when you specify set_case_analysis constraint on `I1.x`, it is applied on the `I1.x` pin.

To apply set_case_analysis on net x, use the following command:

```
set_option sca_on_net 1
```

savepolicy/savepolicies

Usage:	`set_option savepolicy <product-name>`
	`set_option savepolicies <products-list>`

Use these commands to specify a product (`savepolicy`) or a list of products (`savepolicies`) that are not run during design save but can be run during design restore.

Under Design Save-Restore feature, you specify the products during the design save and then run any subset of these products during design restore. Entire base polices and the four advanced products, that is, SpyGlass Constraints solution, SpyGlass DFT solution, SpyGlass Power Verify solution, and SpyGlass TXV solution, are saved by default.

In addition, the design view related to the products being used, which are specified indirectly through goals is also saved. You can further specify additional products to be saved during the design save by using the `savepolicy/savepolicies` command so that these products are not run during design save but can be run during design restore.

If save/restore is enabled then the synthesis mode for the run is determined by the combination of products specified in the *<goal>.spq* file and the products specified using the set_option savepolicy <policy_list>. If the synthesis mode is conflicting based on this product set, and the rules enabled in the current run, then spyglass FATALs out. For more information, see *Goal run Synthesis Flavor* section of the *Atrenta Console Reference Guide*.

For more details on Design Save-Restore feature, see *Design Save and Restore Feature*.

NOTE: *The* `savepolicies` *command works only when the* `enable_save_restore` *command is also specified.*

NOTE: *The* `savepolicies` *command is ignored during the design restore mode.*

Keywords supported by savepolicy/savepolicies

You can specify the following keywords with `savepolicy/savepolicies`:

Keyword	Products Covered by the Corresponding Keyword			
basepolicy	timing	starcad-21	starc2005	starc2002
	starc	simulation	openmore	morelint
	miscellaneous	lint	latch	erc
	area	Audits		
advpolicy	txv	lowpower	power_est	dft
	dft_dsm	const_intern1	constraints	clock-reset
all	Union of basepolicy and advpolicy			

To check for the product names to be used as a keyword for this command, refer to product name directory in your installation directory.

For example, to run the lint product during design restore, specify the lint keyword as mentioned in the <your-inst-dir>/SPYGLASS_HOME/policy/lint directory, with the `set_option savepolicy` command.

Also, consider the following examples:

```
set_option savepolicy dft
```

```
set_option savepolicy {basepolicy dft}
```

The first example listed above runs the SpyGlass DFT product during design restore. In the second example, the SpyGlass DFT product and all the products covered under the basepolicy keyword are run during design restore.

If you do not want to save a product during design save, use the *nosavepolicy/nosavepolicies* command.

sdc2sgdc

Usage:

```
set_option sdc2sgdc <yes | no>
set_goal_option sdc2sgdc <yes | no>
```

The default value of the `sdc2sgdc` option is `no`. However, for `sg_shell` runs, SpyGlass automatically sets this option to yes, when at least one SDC file is specified using the `sdc_data` constraint.

For details on this command, see *Enable SDC-to-SGDC translation*.

sdc2sgdcfile

Usage:

```
set_option sdc2sgdcfile <file-name>
set_goal_option sdc2sgdcfile <file-name>
```

For details on this command, see *Specify the file to save output of SDC-to-SGDC translation*.

sdc2sgdc_mode

Usage:

```
set_option sdc2sgdc_mode <mode-name>
set_goal_option sdc2sgdc_mode <mode-name>
```

For details on this command, see *Specify the mode of the SDC file to be translated to SGDC*.

sfcu

Usage: `set_option sfcu <yes | no>`

Use this command to enable each file to be compiled as a separate compilation unit.

By default, SpyGlass compiles all the specified files in a single compilation unit. Set the `sfcu` command to yes to compile each file as a separate compilation unit.

NOTE: *The* `sfcu` *command can be used only when the* `enableSV` *command is set to* `yes`*.*

sgdc_check_severity

Usage:

```
set_option sgdc_check_severity <COMPAT | DEFAULT | ERROR |
FATAL | WARNING | IGNORE>
```

Use this command to modify the severity or to provide severity control for the *checkSGDC_existence* rule (built-in) and other non-built in rules. Using this command, the specified value is applied on all design constraints globally.

Input Value Description

The following table describes the permissible values for this command and the respective functionality that is triggered for each of the values:

Value	Functionality
COMPAT	This is the default value for the sgdc_check_severity command. When this option is specified, the related rule retains the existing severity. You can perform severity overload for non-built in rules, using this option.
FATAL	Generates violation message with the FATAL severity for the first SGDC command that fails the sanity check.

Value	Functionality
ERROR	Generates violation message with the ERROR severity for any SGDC command that fails the sanity check. The software then ignores or deletes the SGDC command.
WARNING	Generates violation message with the WARNING severity for any SGDC command that fails the sanity check. The software then ignores or deletes the SGDC command.
IGNORE	Ignores and deletes any SGDC command that fails the sanity check.
DEFAULT	Uses the registered severity for the generated violation message. You can not perform severity overload for non-built in rules, using this option

Examples

The following lists some of the examples of using the *sgdc_check_severity* command.

```
sg_shell> set_option sgdc_check_severity {"error"}
sg_shell> set_option sgdc_check_severity {"warning"}
sg_shell> set_option sgdc_check_severity {"compat"}
```

Pragmas to define sgdc severity class

You can also specify individual severity class for one or more constraints, thereby allowing more granularity and block-level control.

The following describes the syntax for specifying sgdc severity class on a block-level:

```
define_sgdc_severity_class -value <option-value>
    constraint <constraint_name>
end_sgdc_severity_class
```

Here, define_sgdc_severity_class -value <option-value> is the pragma for starting a sgdc severity class. Also, end_sgdc_severity_class denotes the pragma for ending an sgdc severity class.

Consider the following sample SGDC file, test.sgdc:

```
sg_shell> cat test.sgdc
current_design test
```

```
constraint 1
define_sgdc_severity_class -value error
    constraint 2
    constraint 3
end_sgdc_severity_class
constraint 4
```

In the above example, constraint 2 and 3 have SGDC existence checks failure reported with severity error. While constraint 1 and constraint 4 either follow the global option value, if specified, or they use the default severity registered with Spyglass.

sgdc_validate

Usage:	`set_option sgdc_validate <true	false>`

Use this command to enable the SpyGlass CDC validation flow.

sgsyn_clock_gating

Usage:	`set_option sgsyn_clock_gating <1	0>` `set_goal_option sgsyn_clock_gating <1	0>`

Specifies whether clock gating should be enabled or disabled.

By default, value of the *sgsyn_clock_gating* command is set to 0, and clock gating is disabled.

Set the value to 1 to enable clock gating. When you set its value to 1, inference of clock gating for flip-flops with explicit enables or feedback loops, is enabled.

After you enable clock gating, use the *sgsyn_clock_gating_threshold* command to set the threshold for the insertion of clock gate logic.

sgsyn_clock_gating_threshold

Usage:

```
set_option sgsyn_clock_gating_threshold <num>
set_goal_option sgsyn_clock_gating_threshold <num>
```

Specifies the threshold of the clock gating logic.

When you set this command, the *sgsyn_clock_gating* command is automatically set to yes.

By default, the *sgsyn_clock_gating_threshold* command is not set and insertion of clock gating logic is disabled. Set the value of the *sgsyn_clock_gating_threshold* option to zero or a positive integer value. This enables clock gating for registers with width greater than or equal to the specified value.

sgsyn_enable_latch_removal

Usage:

```
set_option sgsyn_enable_latch_removal <yes | no>
```

Use this command for better latch detection and removal of redundant latches.

When you set this command to `yes`, SpyGlass checks if the enable pin of a latch is driven by VCC. If yes, SpyGlass replaces such a latch with a buffer.

NOTE: *This optimization results in an increased runtime for synthesis.*

sgsyn_loop_limit

Usage:

```
set_option sgsyn_loop_limit <value>
```

Use this command to specify a loop-rolling limit during design synthesis.

By default, the loop-unrolling limit is set to 2048. When a loop is not completely rolled in the specified number of iterations, SpyGlass marks the module as un-synthesizable.

show_all_sdc_violations

Usage: `set_parameter show_all_sdc_violations <yes | no>`

Use this parameter to show all the sdc_violations during the SDC parsing.

By default, the value of the show_all_sdc_violations parameter is set to no. During SDC parsing, when the parameter is set to no, SpyGlass reports violations for only those products, which you have selected to run.

During sdc2sgdc translation, the sgdc constraints for only those commands are generated, which are converted used sdc2sgdc translation. Syntax error for other commands is not reported.

Use the show_all_sdc_violations parameter to view violation messages for such the sdc commands.

Set the value of this parameter to yes to view all the SDC violations during the SDC parsing, irrespective of the products selected to run.

show_lib

Usage: `set_option show_lib <yes | no>`

Use this command to generate messages for each library module (from libraries specified using the `v` and `y` commands) as it is loaded.

By default, these messages are not generated.

show_sdc_progress

Usage: `set_parameter show_sdc_progress <yes | no>`

Set this command to yes to show the SDC parsing progress bar on standard output during SDC parsing.

NOTE: *The show_sdc_progress parameter will be deprecated in a future release.*

skip_rules_for_fast_restore

Usage: `set_option skip_rules_for_fast_restore <yes | no>`

Use this command to skip checking of rules that require design re-parsing and/or re-synthesis during design restore. For details, see *Design Save and Restore Feature*.

Depending on the selected characteristics of rules, SpyGlass may not be able to work with saved design view and therefore, unchanged design re-parsing and/or re-synthesis may be required. You can skip such rules by setting the `skip_rules_for_fast_restore` command to `yes`.

Example of using the skip_rules_for_fast_restore Command

Consider an example in which you run the *Initial_RTL/ Ensure_RTL_Block_is_simulation_ready/Connectivity* goal, and save the design by setting the `enable_save_restore` command to `yes`.

Now, in the second SpyGlass run, if you specify the same commands as in the first run, SpyGlass reports the following message:

```
WARNING [237]    Following rules cannot run on restored
design database. Hence, HDL being re-read:
W110
STARC05-2.1.3.1
```

In this example, SpyGlass re-reads the design, runs the `W110` and `STARC-2.1.3.1` rules, and restores the netlist from the disk.

In the third SpyGlass run, if you specify the same commands as in the previous run and set the `skip_rules_for_fast_restore` command to `yes`, SpyGlass reports the following message:

```
WARNING [236]    Following rules not being run (design
database restored & 'skip_rules_for_fast_restore' is set):
W110 (need HDL re-read)
STARC05-2.1.3.1 (need HDL re-read)
```

In this case, SpyGlass disables the `W110` and `STARC-2.1.3.1` rules, skips the design parsing, and loads the netlist straight from the disk.

NOTE: *The* `skip_rules_for_fast_restore` *command works only when the enable_save_restore command is set to* `yes`*.*

skip_sanity_on_blocks_nonhier_nets

Usage: `set_option skip_sanity_on_blocks_nonhier_nets <0 | 1>`

Use this command to skip sanity checks on the constraints specified for the non hierarchical nets of an abstract block.

After block abstraction, such nets become non existent. So when you set this command to 1, SpyGlass marks the constraints for such nets as deleted.

sort

Usage:

```
set_option sort <yes | no>
set_option sortmethod <du | lexical>
```

For details on this command, see *Automatically Sort VHDL File(s)*.

sortrule

Usage:

```
set_option sortrule <language>+<rule-name>+<sortorder>
set_goal_option sortrule <language>+<rule-name>+<sortorder>
```

Use this command to specify a sort order for messages in SpyGlass reports.

For details on using this command, see *Customizing the Sorting Order*.

stop

Usage:

```
set_option stop <module-name>
```

For details on this command, see *Stop Design Unit(s)*.

stopdir

Usage:

```
set_option stopdir <directory-name>
```

Use this command to specify a directory so that SpyGlass skips rule-checking on all design units present in source files present in this directory. Such design units are considered as grey boxes during SpyGlass analysis.

For details, refer to the *Managing the Design Hierarchy* topic of *Atrenta Console User Guide*.

When this command is run, Atrenta Console generates the *stop_summary report*.

NOTE: *If your directory name includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your directory name is "abc*d", you need to refer to it as {{abc*d}}.*
*However, if you want to refer to two directories, for example "abc1d" and "abc2d", you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see Pattern Matching Across Features.*

NOTE: *You can specify the `stopdir` command with the stop and the stopfile commands.*

NOTE: *The `stopdir` command works recursively in the specified directory. Therefore, the following example stops all design units in all source files in the directory, `mydir`, and all files in all sub-directories (all levels) under the `mydir` directory:*

```
set_option stopdir mydir
```

Wildcard Support

You can specify the directory/file names with the `stopdir` command by using expressions, as shown in the examples in the following table:

`set_option stopdir "dir1/*"`	Stop all files in directory `dir1` and all files in sub-directories (all levels) under directory `dir1` recursively (as in UNIX Shell expansion)
`set_option stopdir "dir1/*/"`	Stop all files in sub-directories (all levels) under directory `dir1` recursively (as in UNIX Shell expansion)

For more information, see *Pattern Matching Across Features*.

stopfile

Usage: `set_option stopfile <file-name>`

Use this command to specify a file name so that SpyGlass skips rule-checking on all design units specified in that file. Such design units are considered as grey boxes during SpyGlass analysis.

For details, refer to the *Managing the Design Hierarchy* topic of *Atrenta Console User Guide*.

When this command is run, Atrenta Console generates the *stop_summary report*.

NOTE: *If your filename includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your filename is "abc*d", you need to refer to it as {{abc*d}}.*
*However, if you want to refer to two files, for example "abc1d" and "abc2d", you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see Pattern Matching Across Features.*

NOTE: *You can specify the* `stopfile` *command with the stop and the stopdir commands.*

Wildcard Support

You can specify the filenames with the stopfile option using SpyGlass Pattern Matching Support, as shown in the examples in the following table:

`set_option stopfile “a*”`	Stop all files in the current directory whose names match the wildcard expression `a*` (for example, a1, aa1 , abc, etc.)
`set_option stopfile “dir1/*”`	Stop all files in directory dir1
`set_option stopfile “dir?/*”`	Stop all files in directories that match the wildcard expression `dir?` (for example, dir1, dir2 etc.)

For more information, see *Pattern Matching Across Features*.

support_sdc_style_escaped_name

Usage:

```
set_option support_sdc_style_escaped_name <yes | no>
set_goal_option support_sdc_style_escaped_name <yes |
no>
```

Set this command to `yes` to enable SpyGlass recognize Synopsys-style escaped names in SGDC files.

By default, SpyGlass recognizes names that start with a backslash character and end with a space (Verilog) or a backslash character (VHDL) as escaped names as per respective HDL conventions. Therefore, you need to specify such escaped names in the set format in SGDC also.

When you set this command to `yes`, you can specify object names without escaped delimiters in SGDC files. For example, you can specify `'ab%c'` instead of `'\ab%c '` or `'\ab%c\'`.

You can specify Synopsys-style escaped names at all levels in a hierarchical name. For example, you can specify `'top.e.&f%g'` instead of `'\top .\e.&f%g '` or `'\top\.\e.&f%g\'`.

You can use both the default format escaped names and the Synopsys-style escaped names in the same SGDC file.

target

Usage:

```
set_option target <lib-name-list>
set_goal_option target <lib-name-list>
```

Use this command to specify libraries to be used for technology-mapping out of the specified .sglib libraries.

By default, all the specified .sglib libraries are used for technology-mapping.

top

Usage: `set_option top <module-name>`

For details on this command, see *Top Level Design Unit*.

treat_priority_pin_as_obs

Usage: `set_option treat_priority_pin_as_obs <yes | no>`

Use this command to consider the highest priority asynchronous pin of sequential block as observable/unblocked.

By default, the value of this command is set to no. In this case, under certain conditions, the highest priority asynchronous pin may be marked as un-observable/blocked.

For example, in a flip-flop having highest priority `-clear` pin, when `-data` pin is constrained to value 0, Currently the `-clear` pin is returned as un-observable/blocked in default flow.

To treat highest priority asynchronous pin as observable/un-blocked unconditionally, set the value of the treat_priority_pin_as_obs command to yes.

treat_rtl_macro_as_lib_cell

Usage: `set_option treat_rtl_macro_as_lib_cell <yes | no>`

Use this command to treat all the macros in the design as lib cells.

The default value of this command is no.

Usually, there are size limitations while evaluating a macro, for example, max data size = 1024 (2^10). Therefore, evaluation does not take place completely for a large macro used in the design because it exceeds the size limits specified.

However, when you set the value of the *treat_rtl_macro_as_lib_cell* command to yes, all the macros in the design are treated as lib cells. Therefore, evaluation takes place completely for those large macros.

unify_sdc2sgdc

Usage:

```
set_option unify_sdc2sgdc <1 | 0>
set_goal_option unify_sdc2sgdc <1 | 0>
```

Use this command to enable unification of mutually exclusive information from different sources, that is, SDC and SGDC.

NOTE: *In 5.2, unification is done only for the clock sgdc constraint.*

In this flow, SpyGlass first reads the user-specified SGDC constraints and then reads the sdc2sgdc constraints. When you specify the unify_sdc2sgdc option, SpyGlass creates a single unified constraint by combining the information (options) derived from user-specified SGDC constraints and the SGDC constraints that are generated during translation of SDC to SGDC using the *sdc2sgdc* command.

use_block_interface

Usage: `set_option use_block_interface <1 | 0 | yes | no>`

Use this option to trigger the software flow wherein the block interface information in the abstract model specified using the *include_block_interface* is processed. The abstract model can be either be encrypted or in the form of an SGDC.

If both encrypted and interface information is available through SGDC, Spyglass picks the encrypted part and ignores the abstract interface SGDCs.

use_du_sch_hier

Usage:

```
set_option use_du_sch_hier <yes | no>
set_goal_option use_du_sch_hier <yes | no>
```

Set this command to `yes` to:

- Enable use of hierarchical information while applying IP waivers. Consider a design where a module is instantiated in multiple IPs, but `waive -ip` command is not specified for all the IPs. In this case, if you use the `use_du_sch_hier` command, the violation messages for the specified modules are reported.
- Enable use of schematic information to waive violations on a design unit, using the `waive -du` command. By default, only back-reference information of a design unit is considered while waiving a violation using the `waive -du` command.

By default, SpyGlass waives violations on only those module instances that are present in the IPs specified using the `waive -ip` specification.

use_generate_index_style

Usage: `set_option use_generate_index_style <>`

Use this command to specify the format to generate the names of nets/instances inside for-generate statements. You can specify a string or multiple strings in the following format, as an input to this command:

```
set_option use_generate_index_style "X%sY%dZ"
```

The following are the arguments for the above command:

- **X**: Represents the start delimiter. This is an optional argument.
- **Y**: Represents the generate index separator. This is an optional argument.
- **Z**: End delimiter. This is an optional argument.
- **%s**: Mapped to for-generate block label name. The input string must have 0 or 1 occurrence of `%s`. This means that the for-generate block label is not used in the name.
- **%d**: Mapped to for-generate bit index. The input string must have exactly 1 occurrence of `%d`.

Default value of this command is `%s[%d]`. This means that the if nothing as specified as an input to this command, then this format is used to generate names.

NOTE: *The use_generate_index_style command works only when the allow_non_standard_sdc command is specified.*

This command doesn't affect instances or nets inside the `if-generate` and `case-generate` statement. It is only used for naming of `for-generate` objects.

use_generate_separator

Usage: `set_option use_generate_separator <separator string>`

Use this command to specify the separator string to be used between block labels to generate the names of instance/net inside the *generate* statement. The specified string acts as a separator between block and nested blocks and block and nested instance/net.

For example, consider the following command:

```
set_option use_generate_seperator "@^$#&"
```

The default value of this command is '.'.This means, if no value is specified for the command, then '.' is used as generate separator. You can specify single character or a string of multiple characters as an input to this command.

NOTE: *The use_generate_separator command works only when the allow_non_standard_sdc command is specified.*

use_goal_rule_sort

Usage:	`set_option use_goal_rule_sort <yes	no>` `set_goal_option use_goal_rule_sort <yes	no>`

Use this command to sort violation messages in the moresimple report based on the order of rules specified in a goal file.

use_scan_flops

Usage:

```
set_option use_scan_flops <yes | no>
set_goal_option use_scan_flops <yes | no>
```

Use this command to enable SpyGlass to pick scan flip-flops from a technology library for mapping flops in a design.

When you set this command to `yes`, SpyGlass gives priority to scan flip-flops rather than normal flip-flops in a technology library for mapping. If scan cells are not present in the technology library, SpyGlass selects normal flip-flops for mapping in the design.

The `use_scan_flops` command can be specified only when the *-est_mode* command is used or EST mode compatible rules are specified.

use_hier_sep_slash

Usage:

```
set_parameter use_hier_sep_slash <yes | no>
```

Use this parameter to change the default hierarchy separator of the SGDC constraints.

By default, the hierarchy separator used in SGDC constraint’s name field is '.'

For details on this command, see *Changing the Default Hierarchy Separator of the SDC2SGDC Constraints*.

v

Usage:

```
set_option v <file-names>
```

For details on this command, see *Specify the library files in the source design*.

validate_hiersgdc

Usage:

```
set_option validate_hiersgdc <yes | no>
```

Use this command to enable validation of hierarchically migrated block-level SGDC commands to the chip-level. For details, refer to the *Validating Hierarchical SGDC File* topic of *Atrenta Console User Guide*.

verbosity

Usage:

```
set_option verbosity <0 | 1 | 2 | 3>
```

Use this command to enable you to control the verbosity of the log file thereby ensuring that the spyglass.log file is easily readable and content is controllable.

For more information on this command, see *Define Verbosity Level for Log File*.

w

Usage:

```
set_option w <yes | no>
set_goal_option w <yes | no>
```

Use this command to enable generation of warnings for Perl-level compilation and the SpyGlass checker activities.

While running SpyGlass, if you find problems, such as a rule failing, you can get more information about these problems by using the `w` command and running SpyGlass again.

This command enables SpyGlass to display more details while running, identifying each rule as it is checked. Using this command, you can easily identify the failing rule, since it will be the last rule listed before the error message.

Once you have identified the rule causing the problem, you can exclude it by using the `ignorerules` command. This enables you to continue analyzing your design while the specific rule problem is being fixed.

NOTE: *If for some reason, there is a problem in Atrenta Console, it is sometimes difficult to reproduce the problem outside of your company without the design files. These may however be so confidential that it is impossible to make them available, even with a non-disclosure agreement. In order to provide as much debug information as possible without the design files, SpyGlass supports a* `DEBUG` *option. This saves information about what processes SpyGlass was running at the time of the problem, allowing Atrenta development to gain additional insight into the problem. This information is stored by default in a* spyglass.log *file in the working directory, although this can be changed using the* `logfile` *option. You should send this file to Atrenta Customer support when reporting the problem.*

waivers_translate_generate_name

Usage: `set_option waivers_translate_generate_name <yes | no>`

Use this command to enable use of a non-escaped `generate block` name or an `instance array` name in the -`msg` field of the waive command. See *Waiving Messages by Using the waive Constraint* section in the SpyGlass Console User Guide.

work

Usage: `set_option work <work-dir-name>`

For details on this command, see *Logical Working Directory*.

write_sdc

Usage: `set_parameter write_sdc <yes | no>`

Set this command to yes to generate error free constraints in the TCwritesdcInfo file under the <wdir>/spyglass_spysch/spyglass_sdc/ directory.

SpyGlass generates constraints in the TCwritesdcInfo file in the following format:

```
Input SDC command :
set_ideal_network  {A1/in1 in2}
```

```
Output SDC command:
#ideal.sdc@@28@@
set_ideal_network  [list [get_pins {A1/in1}] [get_ports
in2]]
```

These constraints are generated in the following manner:

- All the SDC commands are generated as un-commented and all the non-SDC commands are generated with the `sg_` prefix and are commented.
- Erroneous commands are not translated.
- Basic Tcl commands, such as `puts`, `if`, and `else` are not translated.
- For all objects that are a part of commands, the corresponding `object_access` commands are used with them, such as `get_ports {p1}`.

By default, this command is set to `no`, and the TCwritesdcInfo file is not created.

Please note the following points:

- Although the aim is to draft a legal SDC file that runs error-free in all the tools, SpyGlass is currently not able to stop translation of some erroneous commands, such as SDC_209, SDC_288, etc.
- The set `bus_naming_style/sdc_version` and `setenv` are the only basic Tcl commands that are translated.
- SpyGlass is not able to insert `object_access` commands, `get_clocks`, `get_libs`, `get_lib_pins`, and `get_lib_cells`. They are decompiled as names only.

write_vlog_config_report

Enables reporting of instances, which are bound using the configuration rules.

For more information on this report, see *Support for Verilog Configuration*.

Usage: `set_option write_vlog_config_report <yes|no>`

strict_vlog_config

Use this command to black box and assign the ELAB error to the instances, which are unable to bind, using the verilog configuration rules.

When the value of this command is set to no, the instances that are unable to bind using the verilog configuration flow, are assigned the ELAB warning. However, their binding is done using default spyglass binding flow.

For more information on verilog configuration, see *Support for Verilog Configuration*.

Usage: `set_option strict_vlog_config <yes | no>`

y

Usage: `set_option y { space-separated path-names of directories }`

For details on this command, see *Specify the library directories containing libraries*.

vlog2001_generate_name

Usage: `set_option vlog2001_generate_name <yes | no>`

Set this command to `yes` to unroll all Verilog generate statements in the Verilog 2001 syntax. Unrolling occurs even if a design is a SystemVerilog design for which the SystemVerilog mode is enabled by setting the *enableSV* command to `yes`.

vlog2005_lrm_naming

Usage: `set_option vlog2005_lrm_naming <yes | no>`

Set this command to yes to unroll all Verilog generate statements in the Verilog 2005 syntax. Unrolling occurs even if a design is a SystemVerilog design for which the SystemVerilog mode is enabled by setting the *enableSV* command to yes.

Options Not Recommended

Following commands are not recommended to be used in a project-based flow:

Option	Description
I	Used to specify the directories that contain additional files.

I

Usage: `set_option I {space separated list of directory-name}`

Use this command to specify directories that contain additional files, such as customized rule-deck files and customized SpyGlass-compatible library files (.spyso files).

SpyGlass first locates the additional files in the directories specified by this command, before searching in default locations, such as, <your-inst-dir>/SPYGLASS_HOME/lib and <your-inst-dir>/SPYGLASS_HOME/policies.

You can specify multiple directories, which are searched in the same order as they are specified.

This command also adds the specified directories to LD_LIBRARY_PATH environment variable that defines the location from where the shared objects are loaded.

NOTE: ***The I command is not recommended to be used in the project file. For details, see*** *Options Not Recommended*. ***However, you should specify the directory paths using the -I option on the command-line itself while invoking console or sg_shell. You can view the custom reports by selecting the Report > Default menu option.***

Handling Wildcard Expressions

If your directory name includes wildcard characters (*, or ?), the name should be enclosed in curly braces, and the wildcard characters should be preceded by a backslash (\) to treat them as literal. For example, if your directory name is "abc*d", you need to refer to it as {{abc*d}}. However, if you want to refer to two directories, for example "abc1d" and "abc2d", you can specify them using SpyGlass pattern matching support, that is, you can specify {{abc*d}} in this case. For details on pattern matching support, see *Pattern Matching Across Features*.

Search Mechanism used by the I Command

When you start SpyGlass, it searches the default directory (<your-inst-dir>/SPYGLASS_HOME/policies) to find the available policies, that is, all files with the filename *-policy.pl. Using the I command, you can redefine the path to the policy directory, and have SpyGlass start looking for its files in your custom path first.

Whatever is defined using the I command is prefixed to the already defined default search path. SpyGlass searches in your directory first to find any subsequently referenced files not qualified by full paths. Similarly, it uses your directory as a starting point for relative path references.

Changing the Default File Search Path

To change the path to the directory where SpyGlass begins file searches, enter the I command followed by the path to your directory.

For example, to tell SpyGlass to start the search in the `myperlfiles` directory for a custom policy called, `mypolicy`, and its subsequent file references, specify the following command:

```
set_option I /mypath/myperlfiles
```

You can specify multiple directories with multiple arguments in the I command. Each argument is prefixed to the existing search path. For example, to search mydir1 followed by mydir2 before searching the default search path, specify the following command:

```
set_option I /mydir1 /mydir2
```

List of Topics