Using Waivers to Skip Irrelevant Violations

Table of Contents

Introduction

A Design Rule Checking (DRC) solution like Aldec ALINT-PRO is often used as a signoff tool at the end of an important project's milestone. In the context of a safety-critical process, like DO-254, it would be important to reach 100% conformance between the code of the design and the agreed policy. Although it is generally recommended to resolve all the reported rule violations, sometimes part of the reports might not be that critical to block the milestone finalization. To close the verification loop in such circumstances, ALINT-PRO provides a flexible mechanism of user-defined waivers to annotate the reports the designer would like to treat as irrelevant. The violations thus get classified either as relevant or waived. Waived violations can be hidden from the reports and the GUI debugging instruments, although they can be brought back, if needed. A waived violation is displayed using a gray color scheme to emphasize that it does not have impact on the progress of the linting session. A waiver may also have a comment explaining the adjustment, and those comments could be printed in the reports for a detailed review, if required.

Concept of Waivers

ALINT-PRO verifies designs according to the configured linting policy. Using a flexible policy configuration mechanism, you can include/exclude rules from the process as well as adjust rule's parameters to the actual design's needs.
If any violation is detected during a linting session, it is marked as relevant by default. In most cases, this means that there are certain potential defects or deviations from the agreed coding guidelines within the design, and a deeper analysis and correction might be necessary.
In some scenarios, the detected violation may be considered as irrelevant or low priority. The user then needs to mark those violations as waived manually to express the intent. The waived violations should not be counted as issues that break the verification scenario. A part of the design might be generated by an external tool, i.e. by an IP generator, and the output HDL code for such block could be formatted very differently compared to custom written RTL. A rule violation may be valid, but the issue could have a very low priority for the user. It could also be the case that the violated rule itself is producing overpessimistic results, i.e. the rule could be flagging a wide signal synchronized with multi-bit 2DFF circuit, while implicitly the issue of data coherence between the individual bits is irrelevant in a concrete situation.
The violation may only be waived as a whole - it is not possible to waive a detailed message separately from the main message.
Violation analysis is an iterative process, repeated until all the violations detected by ALINT-PRO are fixed or marked as waived. It is important to remember that waivers may become obsolete as the design progresses. For instance, if a signal or a design unit is renamed between two revisions of the same design, and the waiver could still refer to an old name. As a sanity check, ALINT-PRO warns the user about the outdated waivers, which could not match a design unit or a single violation. Reviewing these sanity check warnings will help to keep the waiving settings up-to-date with the latest revision of the design.

Types of Waivers

The rule violations reported during linting can be distinguished by their corresponding "origin" objects:

- File - code formatting violations (indentation, alignment, comments, etc.);
- Design unit - coding style violations (e.g., described constructs or naming conventions);
- Elaboration unit - violations that take generic values into account (i.e., bit-width rules) and violations that require a netlist for a single unit (i.e., flip-flop inference rules);
- Instance - chip level violations (control signals, design partitioning, CDC, DFT).

In a similar way, there are four origins to add waivers for:

- Directory - grouping origin; affects violations reported on all the files from the directory and its nested elements (useful for 3rd-party IPs or legacy blocks);
- File - file, design units from this file, their elaboration units, instances, and sub-instances with reported violations to be waived;
- Design unit - a waiver relates to the violations reported on a design unit, its elaboration units, instances, and sub-instances;
- Instance - involves violations detected for an instance and its sub-instances.

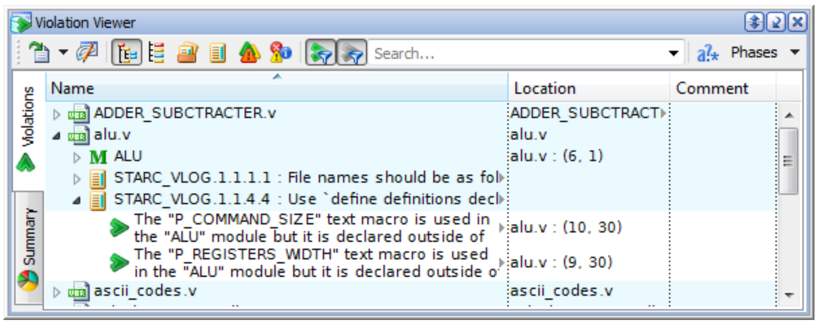
Per each origin, it is possible to:

- Waive all rules for the selected origin object;
- Waive a single rule for the origin object;
- Waive an undesired violation individually - a regular expression waiver.

ALINT-PRO offers several options for waivers management distributed between different windows. Violation Viewer, Waivers Editor (integrated into the project Properties window), Library Viewer, Elaboration Viewer, Project Manager, and File Browser - altogether provide a full-featured environment for addition/removing waivers of all kinds and for all origins.

Waivers in Violation Viewer

All the violations detected during each linting phase can be observed in the Violation Viewer window (Alt + 7). Different views of the window - Hierarchical, Flat, Rulesets, and Rules - regroup the violation list. The user should review every case and fix the problem or introduce a waiver if he/she is sure that it is safe. As it was said above, from the very beginning all the violations are considered relevant by default.

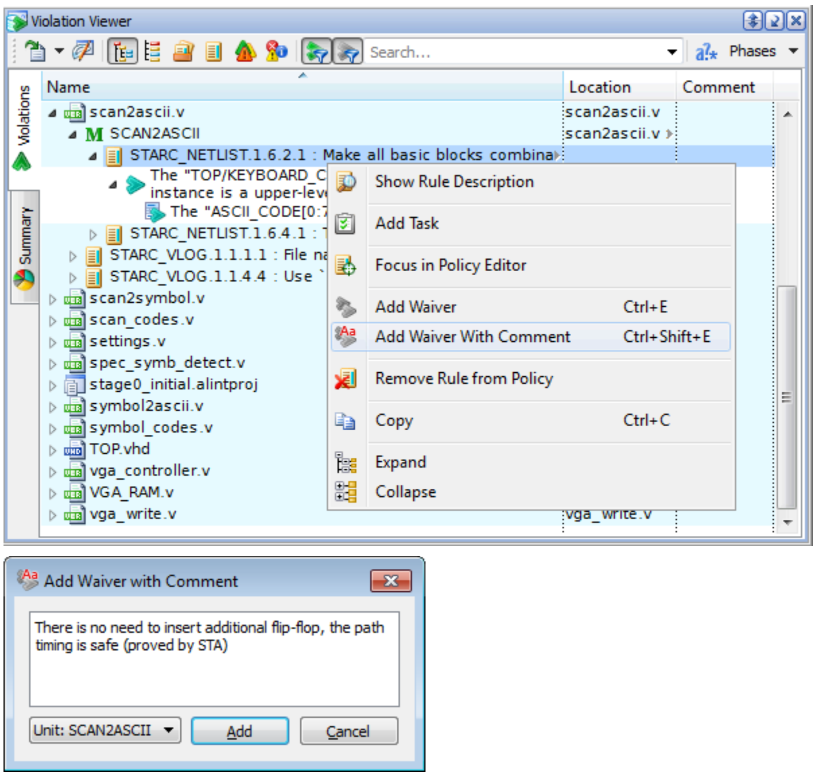


Name	Location	Comment
ADDER_SUBTRACTER.v	ADDER_SUBTRACT	
alu.v	alu.v	
ALU	alu.v : (6, 1)	
STARC_VLOG.1.1.1.1 : File names should be as fol		
STARC_VLOG.1.1.4.4 : Use `define definitions decl		
The "P_COMMAND_SIZE" text macro is used in the "ALU" module but it is declared outside of	alu.v : (10, 30)	
The "P_REGISTERS_WIDTH" text macro is used in the "ALU" module but it is declared outside o	alu.v : (9, 30)	
ascii_codes.v	ascii_codes.v	

Figure 1. Violation Viewer

Let us focus on the violation of rule STARC_NETLIST.1.6.2.1 "Make all basic blocks combinational circuit input and FF output" detected for a design unit. To examine it, we can invoke the Violation Viewer pop-up menu and use the corresponding options to locate the violation in the source code or in the RTL schematic. The role of the reported block might not be described as a "basic block", so there could be a relaxed requirement about the timing boundaries at the output. The user decides to waive the issue. If we click the Add Waiver pop-up menu option for the considered rule node, the rule waiver will be added. It means that from this point on the violation has changed its relevance: it is now waived.

There could be many reasons to ignore certain rules checking results in a certain context. That is why it is highly advisable to add comments explaining the waiving details by using the Add Waiver With Comment pop-up menu option. For our example, we can comment waiving: "There is no need to insert additional flip-flop, the path timing is safe (proved by STA)".

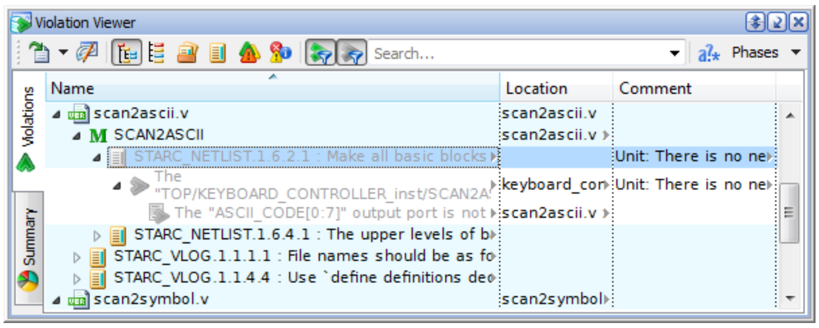


Name	Location	Comment
scan2ascii.v	scan2ascii.v	
SCAN2ASCII	scan2ascii.v	
STARC_NETLIST.1.6.2.1 : Make all basic blocks combina		
The "TOP/KEYBOARD_C instance is a upper-lev		
The "ASCII_CODE[0:7		
STARC_NETLIST.1.6.4.1 :		
STARC_VLOG.1.1.1.1 : File na		
STARC_VLOG.1.1.4.4 : Use `		
scan2symbol.v		
scan_codes.v		
settings.v		
spec_symb_detect.v		
stage0_initial.alintproj		
symbol2ascii.v		
symbol_codes.v		
TOP.vhd		
vga_controller.v		
VGA_RAM.v		
vga_write.v	vga_write.v	

There is no need to insert additional flip-flop, the path timing is safe (proved by STA)

Figure 2. Adding a Waiver with Comments

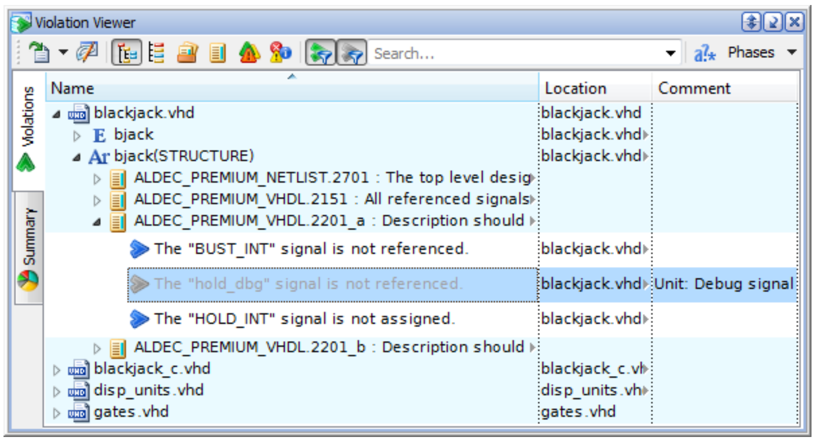
Entered text will be displayed in the Comment column of the Violations tab. Further, it can be edited by clicking on the comment field of the Waiver Editor window.



Name	Location	Comment
scan2ascii.v	scan2ascii.v	
SCAN2ASCII	scan2ascii.v	
STARC_NETLIST.1.6.2.1 : Make all basic blocks		Unit: There is no ne
The "TOP/KEYBOARD_CONTROLLER_inst/SCAN2A	keyboard_con	Unit: There is no ne
The "ASCII_CODE[0:7]" output port is not	scan2ascii.v	
STARC_NETLIST.1.6.4.1 : The upper levels of b		
STARC_VLOG.1.1.1.1 : File names should be as fo		
STARC_VLOG.1.1.4.4 : Use `define definitions dec		
scan2symbol.v	scan2symbol	

Figure 3. Commented Waiver

In some circumstances, you may decide that it is reasonable to waive a single violation of a rule. Violated rule ALDEC_PREMIUM_VHDL.2201_a "Description should rely on each declared signal" may serve as an example. If you right-click the rule node and select Add Waiver (or Add Waiver With Comment), the rule waiver will be set up. Alternatively, you can press either Ctrl+E or Ctrl+Shift+E.



Name	Location	Comment
blackjack.vhd	blackjack.vhd	
bjack	blackjack.vhd	
bjack(STRUCTURE)	blackjack.vhd	
ALDEC_PREMIUM_NETLIST.2701 : The top level desig		
ALDEC_PREMIUM_VHDL.2151 : All referenced signals		
ALDEC_PREMIUM_VHDL.2201_a : Description should		
The "BUST_INT" signal is not referenced.	blackjack.vhd	
The "hold_dbg" signal is not referenced.	blackjack.vhd	Unit: Debug signal
The "HOLD_INT" signal is not assigned.	blackjack.vhd	
ALDEC_PREMIUM_VHDL.2201_b : Description should		
blackjack_c.vhd	blackjack_c.vh	
disp_units.vhd	disp_units.vh	
gates.vhd	gates.vhd	

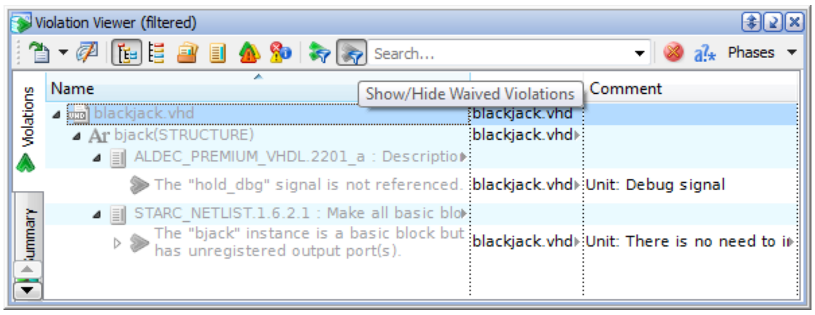
Figure 4. Individual Violation Waiver Example

Internally, the individual violation waivers are based on matching the regular expression against a violation's parameters (such as signal names, values, hierarchical paths). When adding such waivers through the Violation Viewer window, the regular expression is built automatically for your convenience, based on the concrete parameter values available in the violation message. As for our example, it is extracted as "hold_dbg.*signal". All the violations that match this regular expression change their relevance and become waived.

The advanced users may find the waivers based on regular expressions very efficient to select a large group of violations using a short expressive pattern. These advanced waivers will need to be entered explicitly either in the Waivers Editor window or using equivalent TCL commands.

The Violation Viewer provides pop-up menu options to waive all rules for the origin. Select an origin node (it can be a file, a design unit or an instance you need) and click Add Waiver (or Add Waiver With Comment), an all-rules waiver will be set up for the selected origin - both options are context-sensitive.

To keep waivers always in focus and to facilitate the analysis of the violations listed in Violation Viewer, they can be filtered according to their type: either relevant (click Show/Hide Relevant Violations on the toolbar) or waived (click Show/Hide Waived Violations).

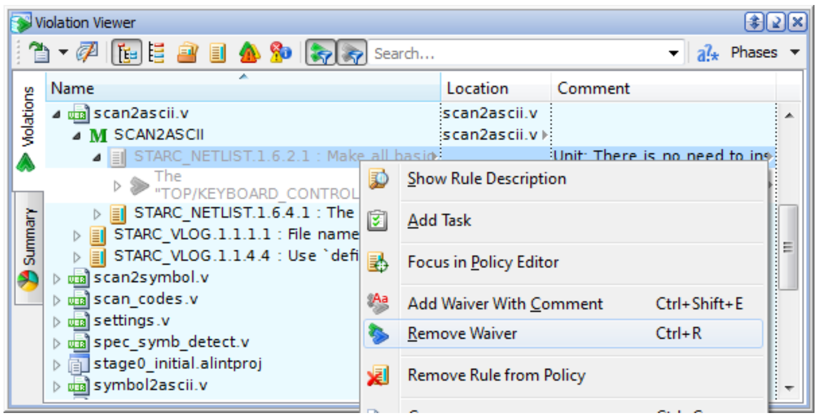


Name	Location	Comment
blackjack.vhd	blackjack.vhd	
bjack(STRUCTURE)	blackjack.vhd	
ALDEC_PREMIUM_VHDL.2201_a : Descriptio		
The "hold_dbg" signal is not referenced.	blackjack.vhd	Unit: Debug signal
STARC_NETLIST.1.6.2.1 : Make all basic blo		
The "bjack" instance is a basic block but has unregistered output port(s).	blackjack.vhd	Unit: There is no need to i

Figure 5. Results of the Show/Hide Waived Violations Filter

Each modification of waivers is applied immediately, there is no need to rerun the linting to see the updated status of violations.

Waivers of any origin and type can be easily removed: use the Remove Waiver pop-up menu option or press CTRL+R against a corresponding waiver node.



Name	Location	Comment
scan2ascii.v	scan2ascii.v	
SCAN2ASCII	scan2ascii.v	
STARC_NETLIST.1.6.2.1 : Make all basic		Unit: There is no need to ins
The "TOP/KEYBOARD_CONTROL		
STARC_NETLIST.1.6.4.1 : The		
STARC_VLOG.1.1.1.1 : File name		
STARC_VLOG.1.1.4.4 : Use `defi		
scan2symbol.v		
scan_codes.v		
settings.v		
spec_symb_detect.v		
stage0_initial.alintproj		
symbol2ascii.v		

Figure 6. A Waiver Removal

Waivers Editor

The Waivers Editor window is a special tool for waivers management. It can be opened via the project Properties | Linting | Waivers page. Here you can set a waiver for any origin using the following buttons on the toolbar or corresponding pop-up menu options:

- Add directory element (specify the name and location of a directory you need to waive);
- Add file element (select a file you need to waive);
- Add design unit element (to waive, select a design unit listed in the drop-down menu);
- Add instance element (locate an element in the drop-down menu).

After adding an element, you should specify what exactly should be waived.

Let us consider design unit bjack(structure). To set up an all-rules waiver for it, you need only to add the following element.

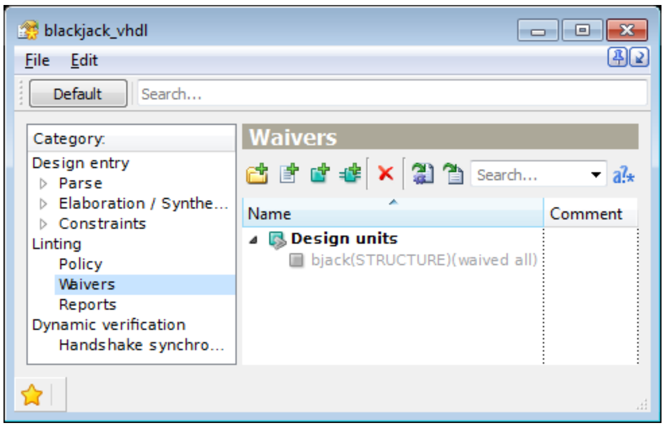


Figure 7. All Rule Waiver Setup

If you need to waive a single rule for this origin, invoke the Add Rule pop-up menu option and select the required rule.

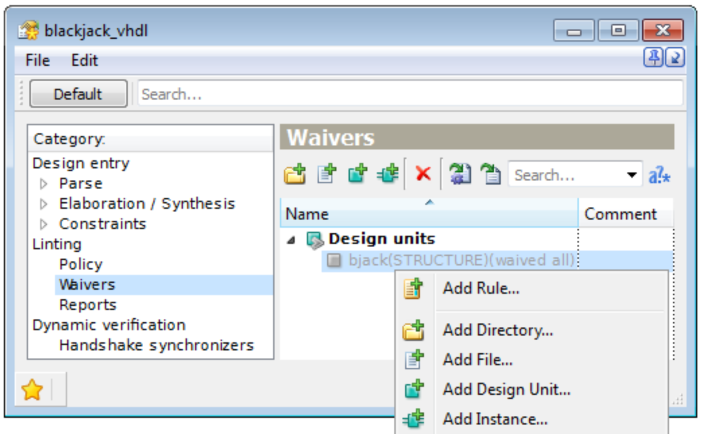


Figure 8. Setup a Rule Waiver

An individual rule violation will be waived if you define a regular expression for the added rule using the corresponding pop-up menu option. The rule violations reported for this origin will be validated and only the violations having fragments that match the regular expression will be waived. Further, you can edit the regular expression (just double-click it and enter a new one).

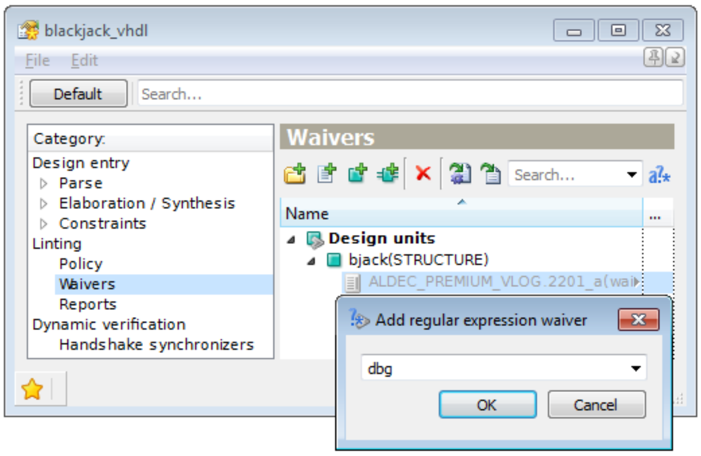


Figure 9. Setup a Waiver with a Regular Expression

It is essential to add comments explaining the details of the waiver. To do this, first select the node in the Name column of Waiver Editor, then click in the adjacent cell (or press F2), and enter the comment. To edit or delete the comment do the same.

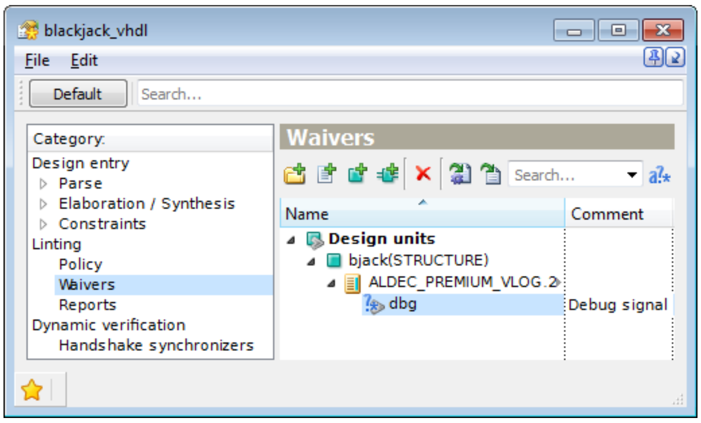


Figure 10. Waiver Comments Editing on Waiver Editor

If you find a necessity to delete or rename the element included in the list of waivers, select such element and use a corresponding option from the pop-up menu. You can clear all waivers in one stroke: use the Clear All Waivers option of the same pop-up menu.
Note that changes of waivers settings are applied immediately after saving (the Save icon or Ctrl+S): there is no need to rerun linting.

Adding Waivers from Other Views

ALINT-PRO allows waiving the entire origin objects in Library Viewer, Elaboration Viewer, Project Manager and File Browser. It is very convenient if the design idea does not require analysis of this or that instance, design unit or file. Set before linting, all rule waivers anticipate unnecessary violations and thus result in less debugging iterations.
For example, to set an all-rules waiver for a design unit, select it in Library Viewer, open the pop-up menu and click the Add Waiver option. At the same time, Waivers Editor includes this waiver into the hierarchical structure.

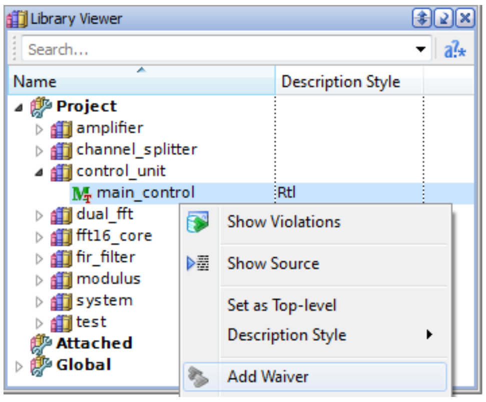


Figure 11. All-Rule Waiver Setup for a Design Unit
All-rules waiver setup for an instance has quite a similar workflow. Select an instance in Elaboration Viewer, open the pop-up menu for it and click the Add Waiver option. Waivers Editor displays the instances hierarchical structure with the grayed-out node. The corresponding node also becomes grayed-out in Elaboration Viewer.
To set up an all-rules waiver for a file, make the Project Manager window active, select the file you need, and then click the Add Waiver pop-up menu option. The result can be observed in Project Manager and Waivers Editor.

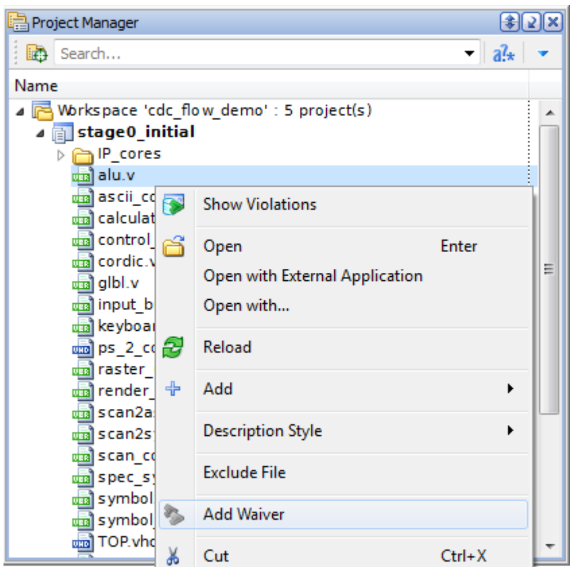
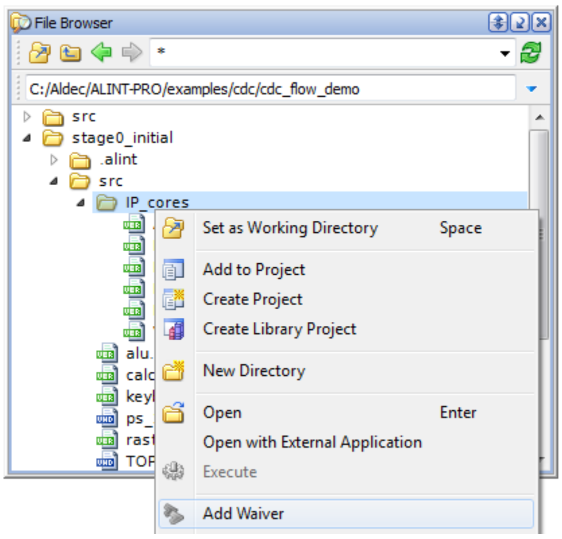


Figure 12. All Rule Waiver Setup for a File
The Add Waiver/Remove Waiver option of the pop-up menu is available in the File Browser against the file(s) or the directory(ies) name. The entire source directory of an IP core or a large functional block that was already verified can be waived all at once.



If the whole directory is waived, any violations reported for a file that belongs to this directory becomes waived automatically.

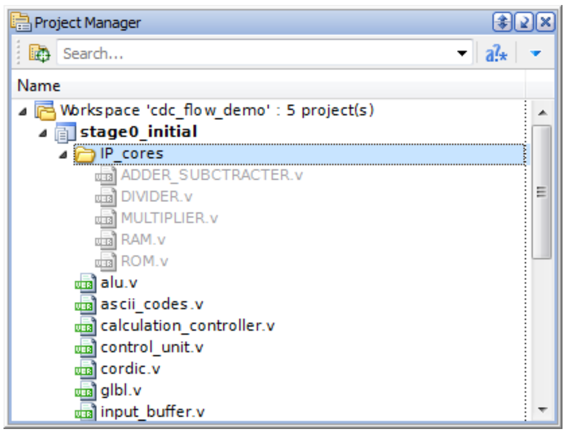


Figure 13. A Waiver Setup from File Browser

Waivers Export to TCL

Waivers settings can be exported to preserve and re-use the configuration. Further, they can be used to revert to an older version or copied to another project sharing some parts with the other. To export waivers, click the Export Waivers to File button on the toolbar of Waiver Editor. Before the export, ALINT-PRO will prompt you to save changes in waivers (if any) and then exports waivers as a macro (*.do) file.
For the example considered at the beginning of this application note, the file with the exported waivers reads:

```
project.waiver.clear

project.waiver.add -rule ALDEC_PREMIUM_VHDL.2201_a -regexp hold_dbg.*signal -unit bjack(STRUCTURE) -comment {Debug signal}
```

Reporting

ALINT-PRO's reporting options summarize verification activities and debugging results. Violations either relevant or waived can be exported to files named after the project (by default). Though reports differ in formats, any of them generally includes:

- Execution information (ALINT-PRO version, project or workspace name, report generation date);
- Summary part: violated rules structured by the projects or by certain criteria (e.g., violations origin, violated rule level, etc.) specified in Summary tab of Violation Viewer;
- List of violations.

For more information on ALINT-PRO reports, see 'Getting Started with ALINT-PRO' application note.
Two windows - Violation Viewer and Flow Manager - enable report generation. The Violation Viewer window drop-down menu lists the formats for linting results export. Click the most suitable for your particular purpose (manual, automatic analysis, design verification process documenting, etc.) and then locate the report file.

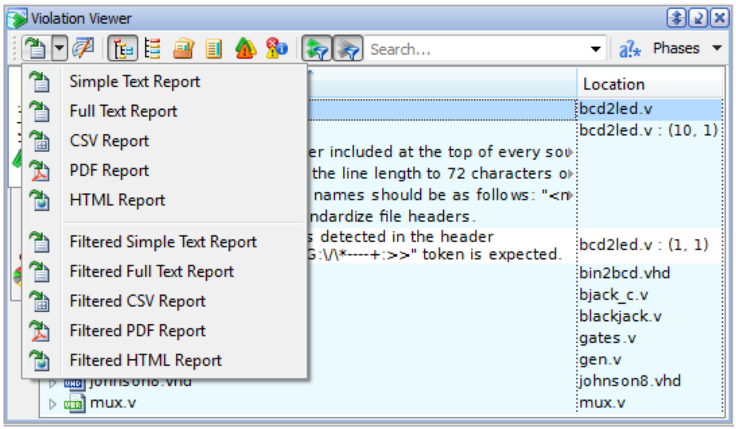
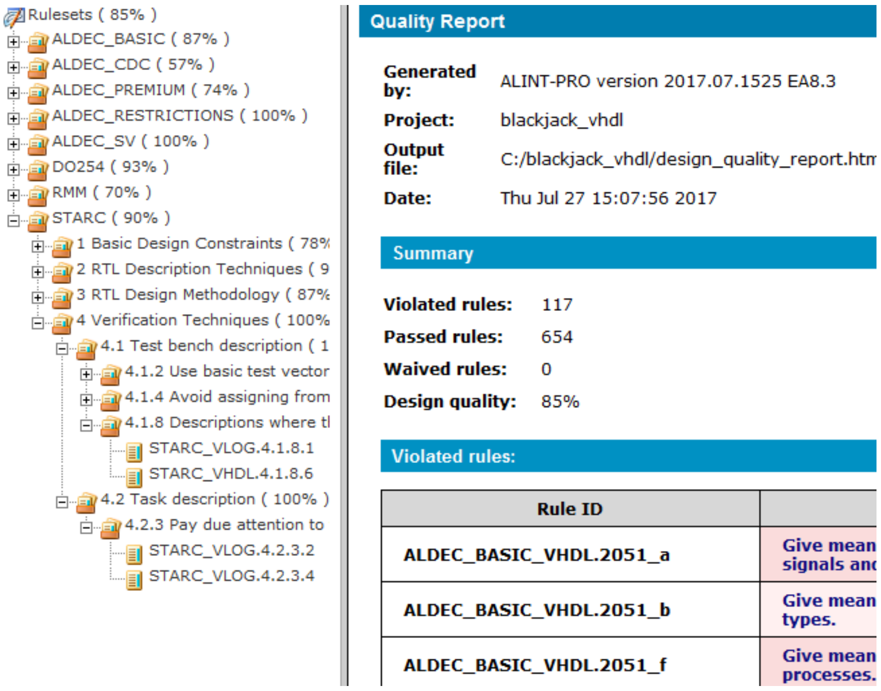


Figure 14. List of Available Report Formats

Report content can be predefined by checking options in the global Preferences window at the Violation Viewer page ("Include summary part in violation reports", "Show empty nodes", "Include source", "Omit origin names") and by entering the max number of violations: in total and per a rule. A report created from Violation Viewer will follow such predefined settings, but you can filter the content in Violation Viewer". Use the toolbar buttons to show/hide relevant or waived violations, then invoke any button for the filtered report - Filtered Simple Text Report, Filtered Full Text Report, Filtered CSV Report, Filtered PDF Report or Filtered HTML Report. This will generate a report with only displayed violations. You can apply regular expressions or user-defined masks to filter the linting results, and again, only displayed violations will be listed upon clicking any button for the filtered report.
Reporting options are also available in the Flow Manager window: double-clicking a certain node within View Results opens the active project report. The report content can be customized if you click a toolbar button or invoke the Properties pop-up menu option upon the node. A configuration panel offers such settings as report location, the max number of violations (total and per a rule), etc. If you need a filtered report, here you can define how to filter the violations (only relevant, only waived or both) as to severity, rule level, rule, etc. Be sure to save changes.
The Custom report node in Flow Manager allows configuring the report in the most optimal way including filtering violations by relevance.
Yet one report type can be generated using the Violation Viewer toolbar button or the Quality Report node in the Flow Manager window - HTML quality report. ALINT-PRO introduces a concept of quality level based on the ratio of the violated and non-violated rules weights. It is expressed in quality percentage and should not be lower than 70%, but it is configurable. Design quality level helps to estimate how close the design is to its performance goals and can be read from the quality report. The report settings are at the global Preferences | Violation Viewer | Quality report page; for the active project report settings click the Properties option on the Flow Manager pop-up menu. The settings allow structuring the navigable rule list, including severity level and waivers comments, demonstrating only violated rules. It is

important that waived rules are considered as non-violated, and thus have no effect on quality percentage. This means that if a rule has several violations one of which is waived, the rule is still considered to be violated and its name is not listed in the "Waived rules" section of the quality report.



Rulesets (85%)
- ALDEC_BASIC (87%)
- ALDEC_CDC (57%)
- ALDEC_PREMIUM (74%)
- ALDEC_RESTRICTIONS (100%)
- ALDEC_SV (100%)
- DO254 (93%)
- RMM (70%)
- STARC (90%)
  - 1 Basic Design Constraints (78%
  - 2 RTL Description Techniques (9
  - 3 RTL Design Methodology (87%
  - 4 Verification Techniques (100%
    - 4.1 Test bench description (1
      - 4.1.2 Use basic test vector
      - 4.1.4 Avoid assigning from
      - 4.1.8 Descriptions where tl
        - STARC_VLOG.4.1.8.1
        - STARC_VHDL.4.1.8.6
    - 4.2 Task description (100%)
      - 4.2.3 Pay due attention to
        - STARC_VLOG.4.2.3.2
        - STARC_VLOG.4.2.3.4

Quality Report

Generated by:	ALINT-PRO version 2017.07.1525 EA8.3
Project:	blackjack_vhdl
Output file:	C:/blackjack_vhdl/design_quality_report.htm
Date:	Thu Jul 27 15:07:56 2017

Summary

Violated rules:	117
Passed rules:	654
Waived rules:	0
Design quality:	85%

Violated rules:

Rule ID	
ALDEC_BASIC_VHDL.2051_a	Give mean signals and
ALDEC_BASIC_VHDL.2051_b	Give mean types.
ALDEC_BASIC_VHDL.2051_f	Give mean processes.

Figure 15. Quality Report Example

Generated reports make it possible to analyze the results outside the tool.

Command Line

For users who prefer TCL scripts and batch mode flows, ALINT-PRO provides a set of commands to control management of waivers, which is equivalent to GUI functionality.

With the project.waiver.add command you can set a waiver for any origin and of any type. This command accepts several arguments. First of all, you need to specify the project name when it comes to a non-active project. Then indicate a rule to be waived (the -rule <rule_name> argument). For all rules waiver use the all keyword. If there is a need to waive a single rule violation, the -regexp <regexp> argument should be added. The next argument should indicate an origin to add waiver for: a directory (-directory <directory_name>), a whole file (-file <file_name>) or its part (-range <range>), a unit (-unit <unit_name>), or an instance (-instance <instance_name>). There are some peculiarities in the -unit argument format: the design unit name specification differs for VHDL and Verilog.

Let us consider the command adding the ALDEC_BASIC_VHDL.2306_b rule waiver for the bjack_c design unit of the blackjack_vhdl project:

```
project.waiver.add -project blackjack_vhdl -rule ALDEC_BASIC_VHDL.2306_b -unit bjack_c
```

Please note that the waiver is applied to the bjack_c entity only, but this rule's violations detected on its architecture are still relevant. Let us make some changes:

```
project.waiver.add -project blackjack_vhdl -rule ALDEC_BASIC_VHDL.2306_b -unit bjack_c -unit bjack_c(bjack_c_arch)
```

Unlike the previous command, this rule waiver is applied to both the bjack_c entity and its architecture bjack_c(bjack_c_arch).

A waiver setup for a single rule violation can be illustrated in the following way:

```
project.waiver.add -rule ALDEC_PREMIUM_VHDL.2201_a -regexp dbg  -unit bjack(STRUCTURE) -comment "Debug signal"
```

The above command waives all the violations that correspond to this regular expression.

Sometimes it is convenient to waive rule violations by their location in the HDL code: specify the line range with the -range argument of the project.waiver.add and all the violations detected for the line range (from Line 23 to Line 25 in the example below) becomes waived.

```
project.waiver.add -project blackjack_vhdl -rule ALDEC_BASIC_VHDL.2306_b -file ./src/blackjack_c.vhd -range (23:25) -comment "ignored rule"
```

Another important argument of the project.waiver.add command is -comment. You are highly recommended to explain the reason of waiving. It facilitates linting results analysis at a later stage. You can comment a waiver while adding it:

```
project.waiver.add -project blackjack_mixed -rule STARC_NETLIST.1.6.2.1 -unit MIXED_BJACK \
                   -comment "There is no need to insert additional flip-flop, the path timing is safe (proved by STA)"
```

or with a separate command later:

```
project.waiver.comment -project blackjack_mixed -rule STARC_NETLIST.1.6.2.1 -unit MIXED_BJACK \
                       -comment "There is no need to insert additional flip-flop, the path timing is safe (proved by STA)"
```

Due to the iterative nature of violations debugging, you should pay close attention to waivers as the project moves forward. With every iteration, they can become obsolete, and it may be reasonable to remove them. ALINT-PRO alerts the user by listing obsolete waivers in the Console window and provides a waiver removal with the project.waiver.remove command. It has the equivalent arguments as project.waiver.add except -comment.

```
project.waiver.remove -rule STARC_NETLIST.1.6.2.1 -unit MIXED_BJACK
```

To remove all the waivers in one stroke, use the project.waiver.clear command. If you have a multi-project workspace, specify the project name you want to clear waivers from.

To check the current state of the violation list, you can use ALINT-PRO reporting options which are also available with the command line. Such commands as project.report.violations and project.report.quality (both with a set of arguments) display printing violations from the project violations database.

To match your particular purpose, you can specify the format of the report as the -format arguments of the project.report.violations command. It will be printed to the Console (simple text report only) or to a file if the -report <output_file_name> argument is specified. Also, you can filter the violations by relevance and print only relevant, only waived, or both.

For example, the following command produces a comma-separated value report with only waived violations found in the blackjack_vhdl project to the report.csv file:

```
project.report.violations -project blackjack_vhdl -report report.csv -format csv -relevance waived -csv_separator ";"
```

where -csv_separator sets a symbol to be used as a separator in the CSV report. It accepts either one-character value or a string. If omitted, then it uses a separator defined by the global Preferences.

The summary of all waivers that exist in a project can be obtained with the project.waiver.list command as well.

Filtering options of project.report.violations also include origin (a source file, an elaboration unit, an instance name and/or unit name should be specified). Each of these four arguments is processed as a regular expression. You can see all the violations of a particular rule(s) if you apply the -rule or the -tag arguments where you specify a rule name or a rule tag.

If there is a need to include violations that have a given severity or a rule level, use the -severity or the -rule_level argument correspondingly. You can also limit the number of the reported violations, or group the violations according to the specified criteria. The Console window will display only relevant violations (by default) that answer to the filter.

The project.report.quality command generates a design quality report. You can specify not only the file name but how to classify violations (according to rules, rule levels or rulesets). By default, the quality report shows violated, non-violated, and waived rules (with comments if -waiver_comment is applied), but there is a possibility to limit this number with only violated rules (-violated_only).

```
project.report.quality -project blackjack_mixed -report qreport.html -classification rule_levels -waiver_comment
```

This command generates the project quality report in the qreport.html file where rule levels are used as a grouping criterion.

As it was said above, the existing project waivers can be exported to a TCL macro:

```
project.export.waivers -project blackjack_mixed -name my_newdesign_waivers.do
```

where -name specifies a name of an exported file but, if omitted, it is named as <project_name>_waivers.do. The .do extension is added to the exported file name if no extension is specified for it. When the macro is run, the exported waiver settings are applied to the active project.

Refer to ALINT-PRO documentation for details on the possible commands and their arguments (see Help | Command Reference).

Conclusion

ALINT-PRO provides instrumentation to handle the results obtained during an HDL design's verification. The analysis of the detected violations may lead to the code debugging, linting policy reconfiguration (see 'Customizing Linting Policy' Application Note), and setting a certain number of waivers, which seem to be irrelevant. Waivers can be applied to the violations of different origins and be configured to achieve the acceptable of linting results (preferably, having 0 relevant violations).

It's important, however, not to overuse the waivers to mask real bugs. When placing a waiver, it is highly recommended to specify an explaining comment. This will help other members of the team to understand the motivation of the waiver. It will also serve as a future reminder for the next milestone. It is important to keep the waiver settings up-to-date, and that's when the sanity check warnings become useful.

Aldec, Inc.
Corporate Headquarters
2260 Corporate Circle
Henderson, NV 89074 USA

Tel: +1 702 990 4400
Fax: +1 702 990 4414
https://www.aldec.com