



# OpenHW Group

Proven Processor IP

# Linting CORE-V-VERIF

Mike Thompson  
[mike@openhwgroup.org](mailto:mike@openhwgroup.org)



**OPENHW** GROUP™  
— PROVEN PROCESSOR IP —

# Objectives

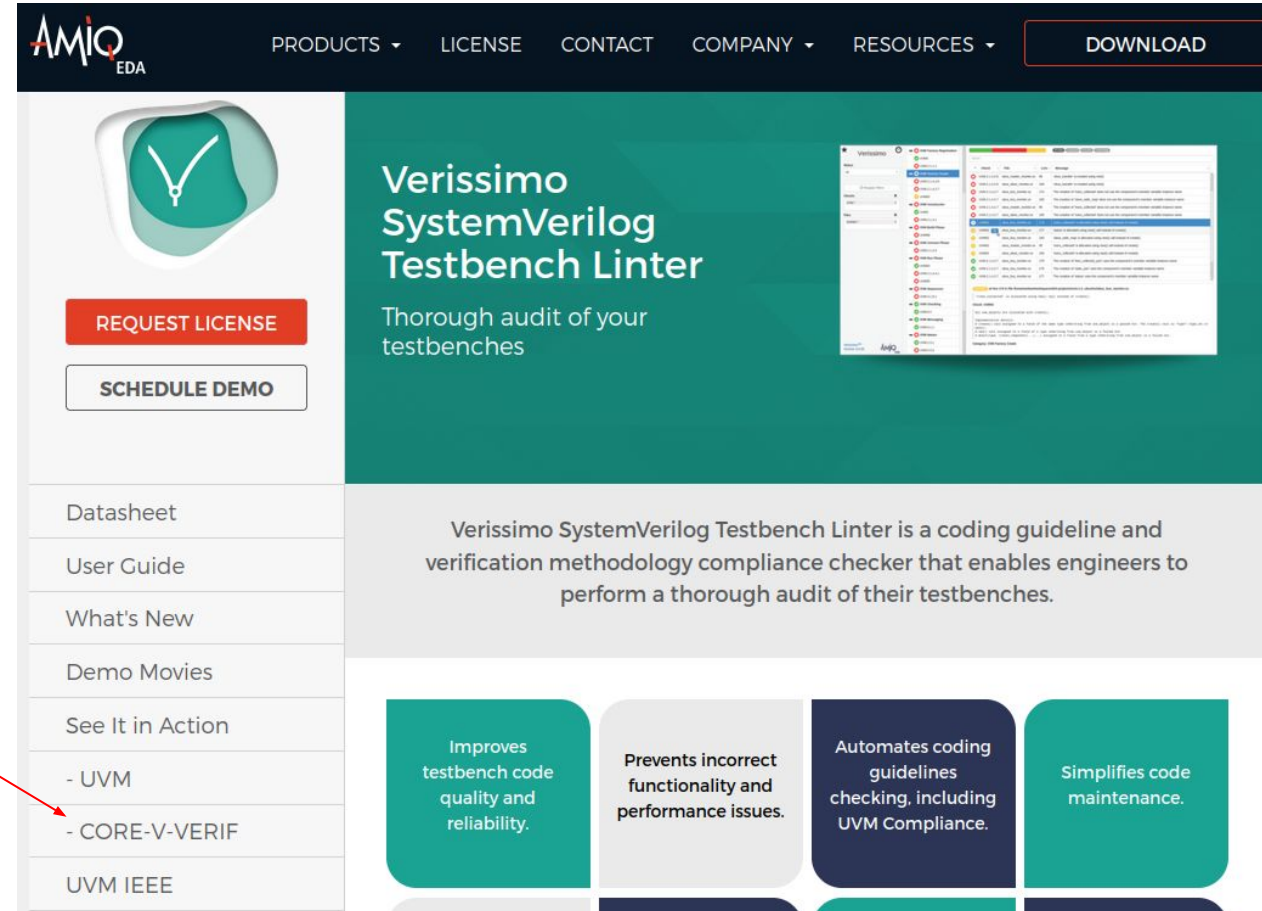
- What-and-Why of Linting for DV code.
- Introduce the AMIQ EDA Verissimo Linter.
- Communicate Linting strategy CORE-V-VERIF.
- Get starting cleaning up our code.

# What-and-Why of Linting

- A Linter is a static code analysis tool used to identify stylistic errors and suspicious constructs in source code.
- Linters are not widely used in DV code for many reasons:
  - Viewed as “extra work”.
  - Limited value in a closed-shop using a single simulator.
  - Low signal-to-noise ratio.
  - Most HDL linters target RTL code, not DV code.
- Why should OpenHW use a linter?
  - Our code base is open-source, so we want to set a good example.
  - Need to work with any 1800-2017 compliant SV simulator.
  - Automate checking of good coding standards.

# Verissimo SystemVerilog Testbench Linter

- Commercial product of AMIQ EDA.
- AMIQ is regressing CORE-V-VERIF using Verissimo:
  - Check out <https://dvteclipse.com/products/verissimo-linter> and click CORE-V-VERIF
- Verissimo automatically runs every six (6) hours on the [master](#), [cv32e40p/release](#) and [cv32e40p/dev](#) branches, plus a set of recent PRs.

A screenshot of the Verissimo SystemVerilog Testbench Linter website. The header features the AMIQ EDA logo and navigation links for PRODUCTS, LICENSE, CONTACT, COMPANY, and RESOURCES, along with a prominent DOWNLOAD button. The main content area has a green background with the product name and a tagline "Thorough audit of your testbenches". A sidebar on the left contains links to a Datasheet, User Guide, What's New, Demo Movies, and See It in Action, as well as a list of supported standards including UVM, CORE-V-VERIF, and UVM IEEE. A red arrow points from the "CORE-V-VERIF" link in the sidebar to the third bullet point in the text on the left. The bottom section highlights four key benefits: improving testbench code quality, preventing functionality issues, automating coding guidelines, and simplifying code maintenance.

# Quick Demo...

(1 of 3)



- Click CORE-V-VERIF and scroll down to get here.
- Select **OPEN RESULTS** on the **master** branch

Branch	Commit	Result
<b>master</b>	<b>Branch tip (current)</b> <a href="#">TREE</a> <a href="#">DIFF</a> <b>1f430b7</b> / Mike Thompson / 4 days ago Merge pull request #909 from MikeOpenHWGroup/master	<b>OPEN COMPARE</b> cv32e40p <b>Removed Rules:</b> 19 errors <b>Common:</b> 231 errors, 101 disabled
	<b>Previous commit (baseline)</b> <a href="#">TREE</a> <a href="#">DIFF</a> <b>471aede</b> / MikeOpenHWGroup / 5 days ago Disable Check: SVTB.12.1.2	<b>OPEN RESULTS</b> 231 errors, 101 disabled
		<b>OPEN RESULTS</b> 250 errors, 105 disabled



# Quick Demo...

(2 of 3)



Click “Reapply Filters”

Select your name

Status

All

Reapply Filters

Checks

Files

Author

Mike Thompson

MikeOpenHWGroup

Alfredo

Alfredo Herrera

Andreas Kurth

Andreas Traber

André Sintzoff

Antonio Mastrandrea

Arjan Bink

bluew

All Failures

Compilation

SYNTACTIC\_PROBLEM

SEMANTIC\_PROBLEM

NON\_STANDARD

Literal Values

SVTB.4.1.7

Singular Data Types

SVTB.5.2.1.1

Aggregate Data Types

SVTB.6.1.2.1

Classes

SVTB.7.1.2

SVTB.7.13

SVTB.7.15

SVTB.7.20

SVTB.7.26

Operators

SVTB.5.11.2.1

SVTB.10.7.3

Methods and Method Calls

SVTB.12.2.6.1

Randomization

Summary

Verissimo generated the report using: ruleset.xml and waivers.xml.

Checks (55) | Autocorrectable (10)

Passed

60.00% (33)

Failures (231) | Autocorrectable (12)

Errors

100.00% (231)

Top Failed Checks

Check	Failures
NON_STANDARD	51
SVTB.32.2.0	33
SVTB.29.1.3.1	33
SVTB.29.1.7	23
SVTB.33.1.0	16
SVTB.28.1	12

Top Failed Files

File	Failures
monitor.sv	25
uvma_obi_memory_drv.sv	19
uvme_rv32isa_covg.sv	17
uvmt_cv32e40p_firmware_test.sv	14
typedefs.sv	9
uvme_cv32e40p_env.sv	8

# Quick Demo...

(3 of 3)



Select a specific “Failure”

The screenshot displays the Verissimo web interface. On the left, there are filters for Status (set to 'All'), Checks, Files, and Author (listing 'Mike Thompson' and 'MikeOpenHWGroup'). A red arrow points from the text 'Select a specific “Failure”' to the 'SVTB.15.4.1.1' failure entry in the central list. The central list also includes categories like 'Randomization', 'Duplicate Code', 'XVM', 'UVM28', 'ARDI', and 'ARSI'. On the right, a table shows the details of the selected failure:

Check	File	Line	Message
SVTB.15.4.1.1	riscv_random_interrupt_generator.sv	128	value.randomize() with {...} is not checked!

Below the table, the 'Author' is listed as 'Mike Thompson'. The error message is repeated: 'ERROR at line 128 in file cv32e40p/tb/core/tb\_riscv/riscv\_random\_interrupt\_generator.sv: value.randomize() with {...} is not checked!'. A check summary states: 'Check: SVTB.15.4.1.1 - Check randomize calls'. The explanation notes: 'The LRM does not specify a simulator runtime error in the event of failed randomization. You must check randomization failures using if or assert.' Examples of code are provided:

```
Examples:  
if(!myclass.randomize())  
    uvm_fatal( get_type_name(), "myclass randomize failed!" )
```

At the bottom left, the 'Verissimo™ Version 21.1.41' logo is visible, and at the bottom right, the 'AMIO EDA' logo is present.



# Goals for CORE-V-VERIF Linting

- In the short to medium term: a “Clean” SystemVerilog/UVM code base that is:
  - 100% IEEE-1800-2017 (SystemVerilog) and IEEE-1800.2-2017 (UVM) Compliant.
  - Works with all known IEEE-1800 (2017) capable simulators.
  - Free of static bugs.
- In the long term, linting should contribute to the definition CORE-V-VERIF DV coding style guidelines.



# CORE-V-VERIF Linting Strategy

- Examine issues as reported by Verissimo
  - Mike will generate GitHub issues for existing issues.
- Possible outcomes of lint issues:
  - Fix the code.
  - Waive the instance.
  - Recommend project wide waiver:
    - Current project-wide waivers are in [vendor\\_lib/verissimo/waivers.xml](#)

# Getting Started



- We already are...
- Mike will be issuing GitHub issues to each Contributor that authored an issue flagged by Verissimo.

# Thank You