OpenHW Group

Proven Processor IP

Linting CORE-V-VERIF

Mike Thompson

mike@openhwgroup.org

October 5, 2021

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

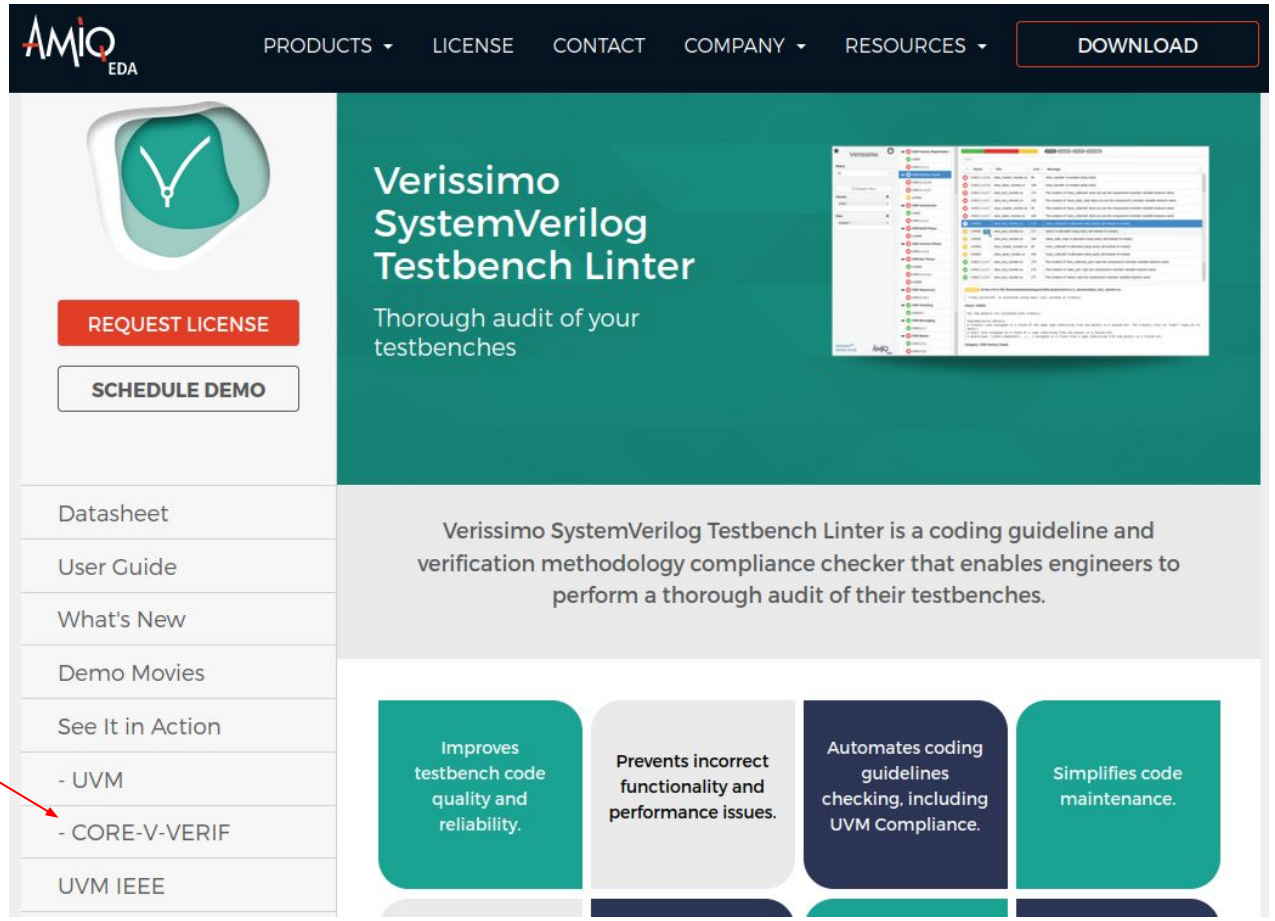
- A Linter is a static code analysis tool used to identify stylistic errors and suspicious constructs in source code.

- 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.



Quick Demo…

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

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

Quick Demo...

(2 of 3)

Click “Reapply Filters”

Select your name

File Edit View History Bookmarks Tools Help

https://dvteclipse.com/core5verif-verissimo/1/reports/run/1f430b7d805e2459f01c46e6ba58ef1dcbe59419/index.html

Verissimo

Status

All

Reapply Filters

Checks

Files

Author

Mike Thompson

MikeOpenHWGroup

Alfredo

Alfredo Herrera

Andreas Kurth

Andreas Traber

André Sintzoff

Antonio Mastrandrea

Arjan Bink

bluew

Verissimo™
Version 21.1.41

AMIQ EDA

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

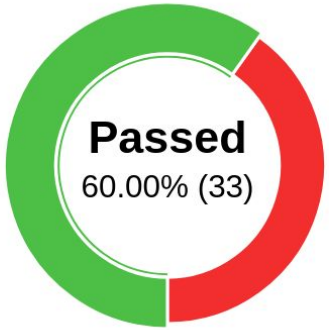
SVTB.12.2.6.1

Randomization

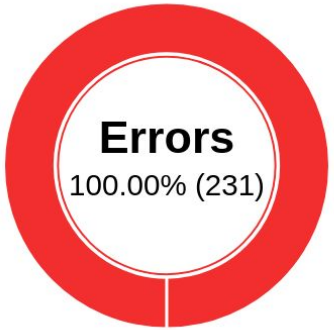
Summary

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

Checks (55) | Autocorrectable (10)



Failures (231) | Autocorrectable (12)



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

GROUP OPENHW™ PROVEN PROCESSOR IP

CORE-V™

Quick Demo...

(3 of 3)

Select a specific “Failure”

https://dvteclipse.com/core5verif-verissimo/1/reports/run/1f430b7d805e2459f01c46e6ba58ef1dcbe59419/index.html

Verissimo

Status

All

Reapply Filters

Checks

Files

Author

Mike Thompson

MikeOpenHWGroup

Verissimo™
Version 21.1.41

AMIQ EDA

All Failures

Randomization

SVTB.15.4.1.1

Duplicate Code

SVTB.33.1.0

SVTB.33.2.0

XVM

UVM28

ARDI

ARSI

1 error

Search...

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

Author Mike Thompson

ERROR at line 128 in file cv32e40p/tb/core/tb_riscv/riscv_random_interrupt_generator.sv

```
value.randomize() with {...} is not checked!
```

Check: SVTB.15.4.1.1 - Check randomize calls

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

CORE-V™

OPENHW GROUP™
PROVEN PROCESSOR IP

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 will 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