Improve the Quality of the Testbenches using specialized PySlint solutions

- Open-source based, customizable TestBench analysis

Balram Meghavath (Balram.Meghavath@Broadcom.com)

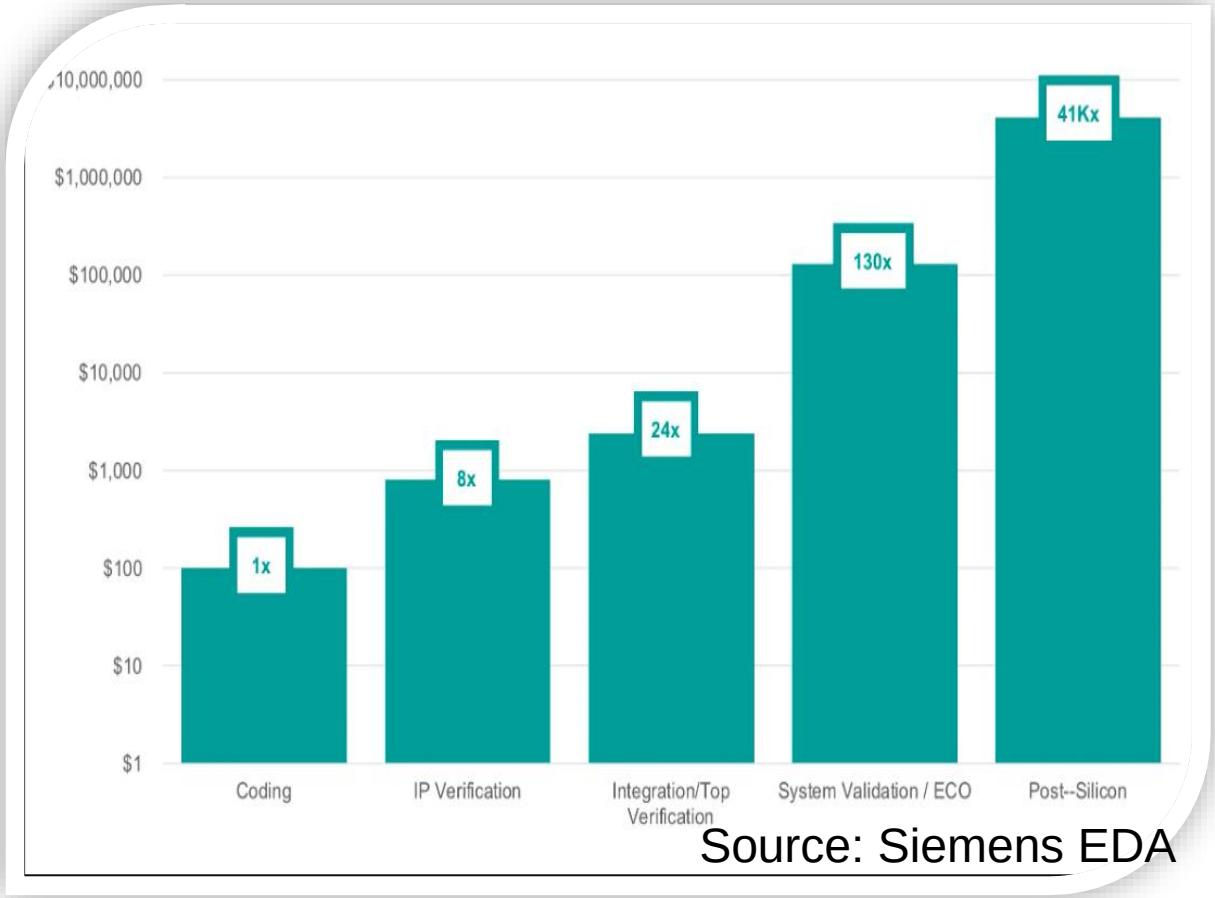
Srinivasan Venkataramanan (svenkat@asfigo.com)

As Figo open-source driven

BROADCOM

Lint and Code Analysis

- Find bugs early
- RTL Lint well proven
- Verification
  - SystemVerilog TB
  - UVM Code base
  - Assertions
  - Acceleration
  - Emulation
  - Formal Verification



Source: Siemens EDA

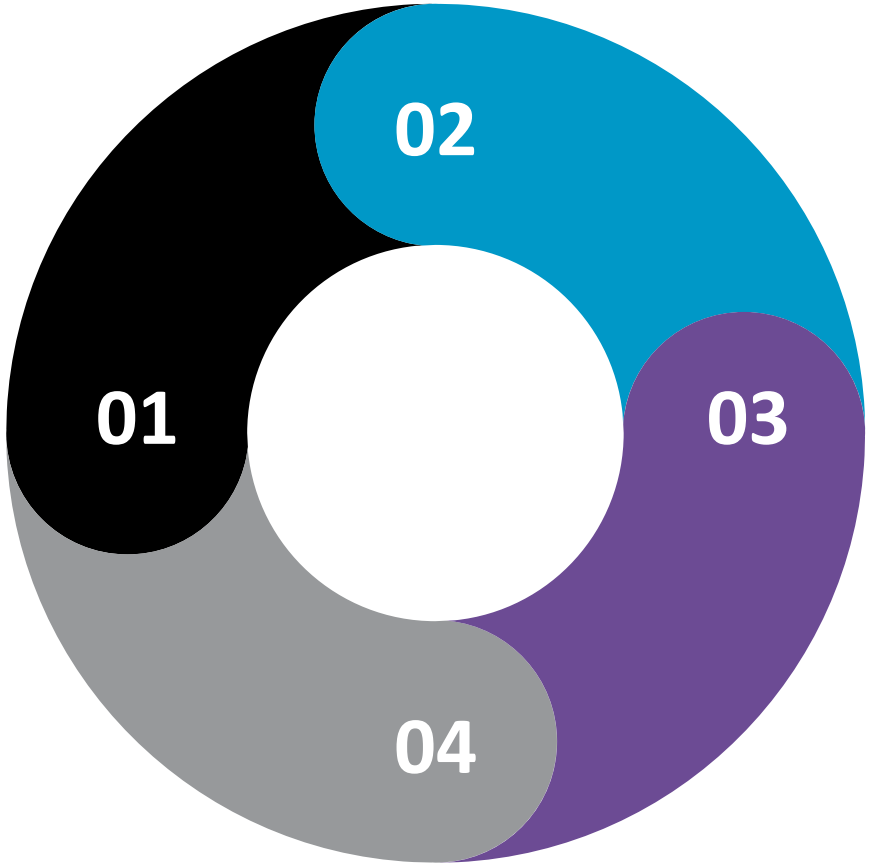
As Figo open-source driven

BROADCOM

RTL Lint is Very Popular

RTL linting is widely used in the field of digital design

It helps to identify and fix issues in RTL code



Ensures adherence to naming guidelines and coding style checks

Improves overall code quality and reliability

As Figo
open-source driven

BROADCOM

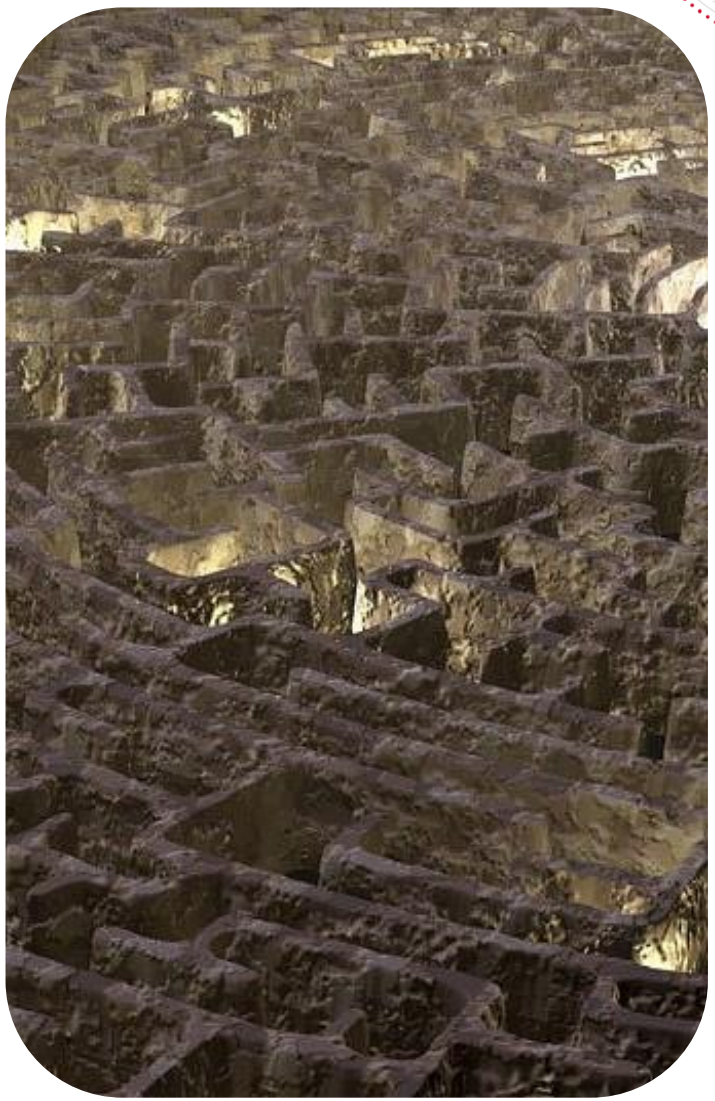
Open-source RTL Lint - options

- Widely used flow – RTL Lint
  - Naming guidelines
  - Coding style checks
  - Synthesis checks
  - DFT checks
- Many teams even integrate this to CI/CD flows
- Good opensource alternates becoming available

BROADCOM

TB Lint - Challenges and Alternatives

- New Paradigm: Code analysis approaches changing
- Lack of good parsers/tools/API for newer languages
- Slang Verible, Svlint, and PySlint are good opensource alternatives
- Provide flexibility and customization options



As Figo open-source driven

BROADCOM

Testbench Linting

- Need solid parser front-end
- Verible – C++ based rules https://github.com/chipsalliance/verible
- Svlint – Rust based, custom plugin
  https://github.com/dalance/svlint
- UVMLint – Python based, custom plugin for UVM TB
  - https://github.com/AsFigo/UVMLint
- PySlint – Python based, works on top of slang/pyslang
  - https://github.com/AsFigo/pyslint

AsFigo open-source driven

BROADCOM

Sample rules in PySlint – Class, SVA, UVMLint

PySlint Rule ID	Description
NAME_CL_PREFIX NAME_CG_PREFIX NAME_ASM_PREFIX NAME_COV_PREFIX	Naming/style checks
SVA_MISSING_LABEL SVA_MISSING_ENDLABEL SVA_NO_PASS_AB SVA_MISSING_FAIL_AB	Assertion specific. Performance, Debug categories
CL_METHOD_NOT_EXTERN CL_MISSING_ENDLABEL FUNC_CNST_MISSING_CAST	Class specific – style, functionality

UVMLint Rule ID	Description
UVM_DRV_MISSING_PARAMS	Driver should have REQ parameter overridden
UVM_AGENT_IS_ACT_REUSE	Agent should check is_active before constructing drivers, sequencer for reuse
UVM_AGENT_IS_ACT_HIDE	Do not re-declare is_active field in uvm_agent
UVM_MON_MISSING_VIF	Monitor should have a handle to virtual interface

Case study: SystemVerilog constraint dist issue

```
class data;
 rand bit [15:0] field1;
 constraint c_f1 { field1
   dist {[0:31] := 1, [32:65535] := 1}
```

:=

vs.

:/

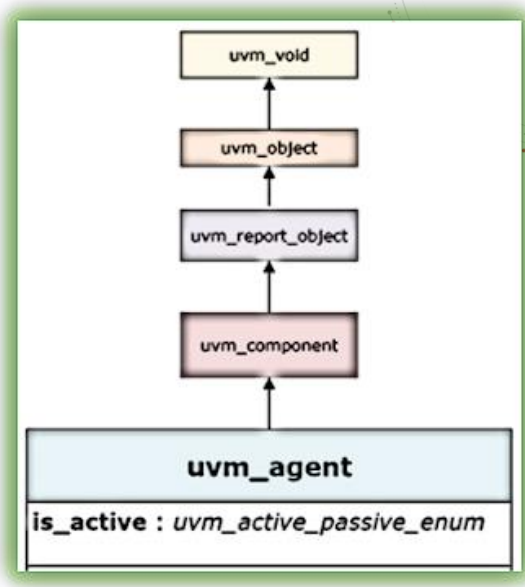
The := operator assigns the specified weight to the item or, if the item is a range, to every value in the range.

The :/ operator assigns the specified weight to the item or, if the item is a range, to the range as a whole. If there are n values in the range, the weight of each value is range_weight / n. For example:

```
PySlint: Violation: [FUNC_CNST_DIST_COL_EQ]: Potentially incorrect constraint
expression! An expression involving dist ColonEquals if found And the range us
ed with ColonEquals is large This is likely to skew the random generation and
prevent other values in the dist expression to be generated less-frequently th
an the large range values Review to check if you intended to use ColonSlash in
stead of ColonEquals field1 dist {[0:31] := 1, [32:65535] := 1}
```

Case study: UVM agent - is_active redeclared

```
16
17 class hbus_master_agent extends uvm_agent;
18
19   // This field determines whether an agent is
20   protected uvm_active_passive_enum is_active
21   // Master's id
```



- The value of *is_active* as declared above would be set to be *UVM_PASSIVE* - this is against the UVM standard implementation that makes it *UVM_ACTIVE* as the default.
- Protected would prevent an env class unable to access this field - a common code style other UVCs (that do not override base class's *is_active* unlike above).
- What's the point of using *uvm_agent* here as base class and not *uvm_component*? - make no mistake; we are not suggesting you do that, rather questioning why the re-declaration in user agent code.

As Figo open-source driven

BROADCOM

Case study – improper use of array reduction methods in constraints

Avidan Efody • 1st
HW/SW Verification Expert
2d • Edited •

All except Dave Rich: Where's the bug in this code?

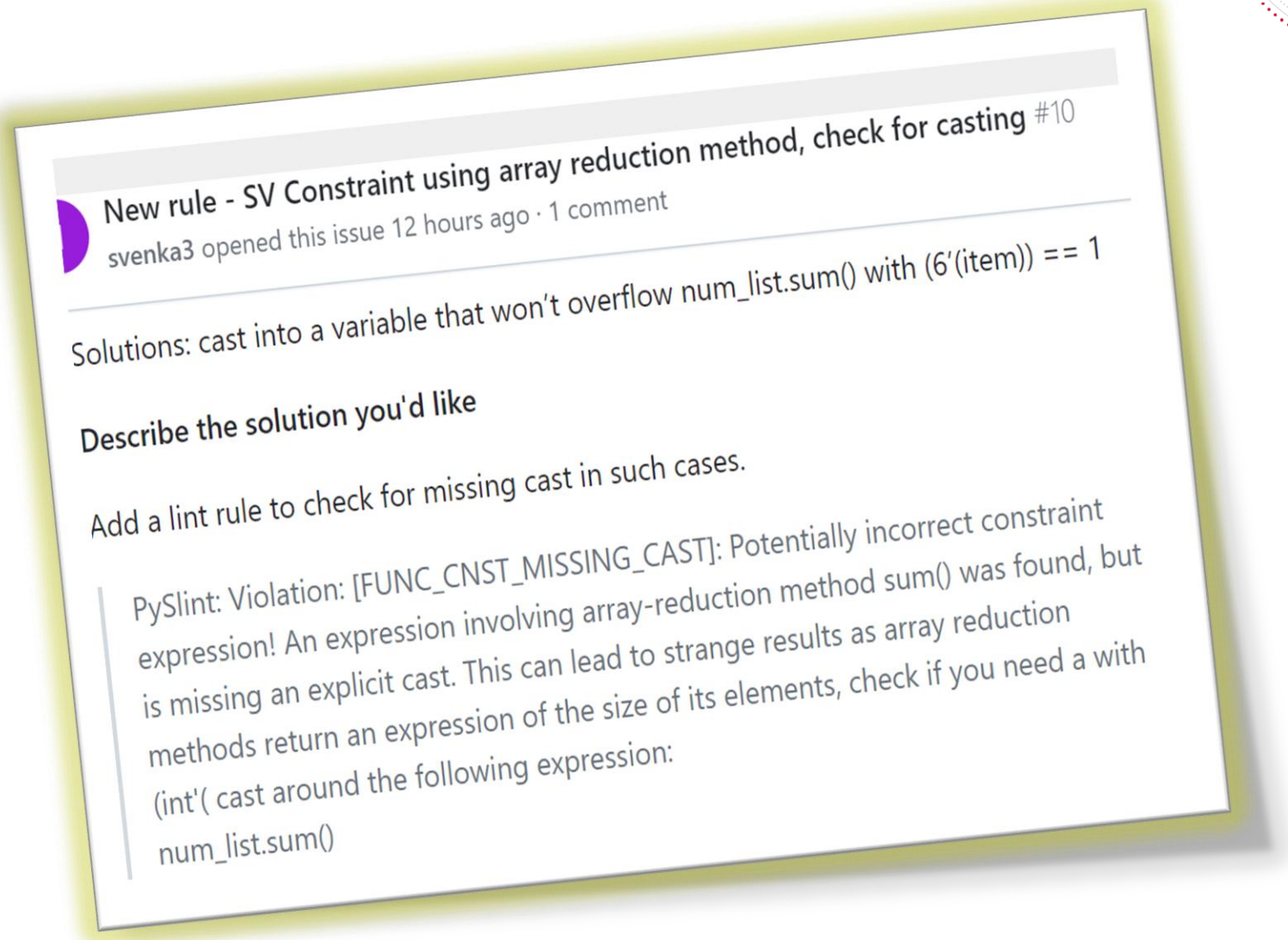
(a bug worth mentioning because it is a very common and a good one to keep in mind when reviewing code)

Answer in first comment.

#constraint_solver_tips #servingTheNextBug #apple_dv #applesilicon #verification

```
class bug_c;
  rand bit num_list[32];

  constraint c_sum {
    num_list.sum() == 1;
  }
endclass
```



New rule - SV Constraint using array reduction method, check for casting #10

svenka3 opened this issue 12 hours ago · 1 comment

Solutions: cast into a variable that won't overflow num_list.sum() with (6'(item)) == 1

Describe the solution you'd like

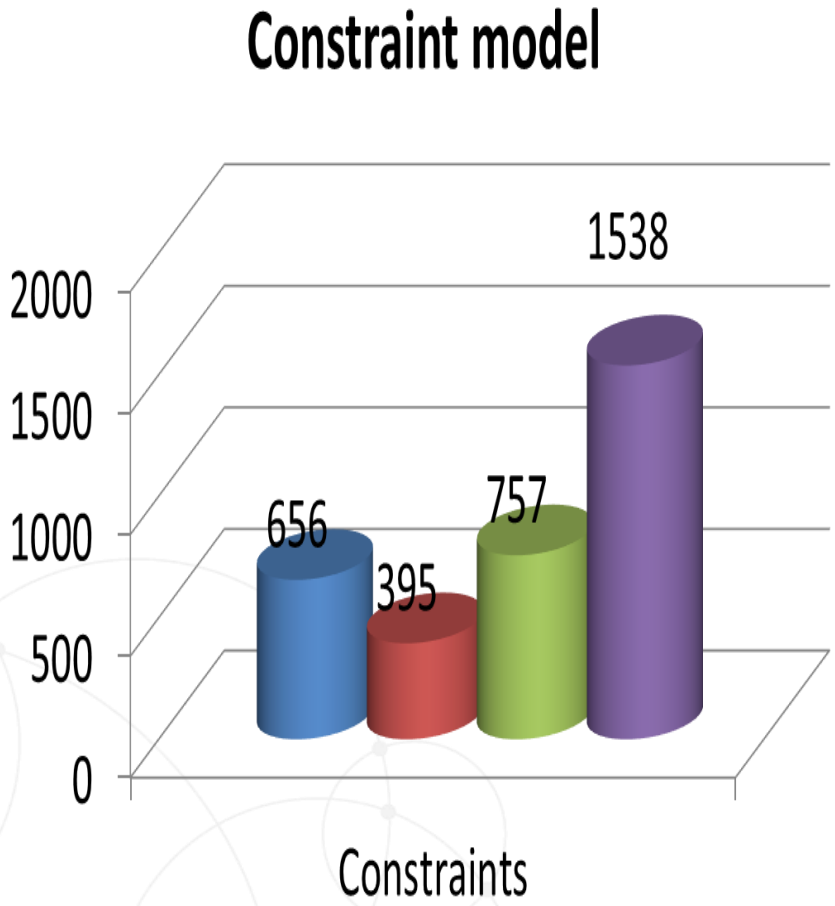
Add a lint rule to check for missing cast in such cases.

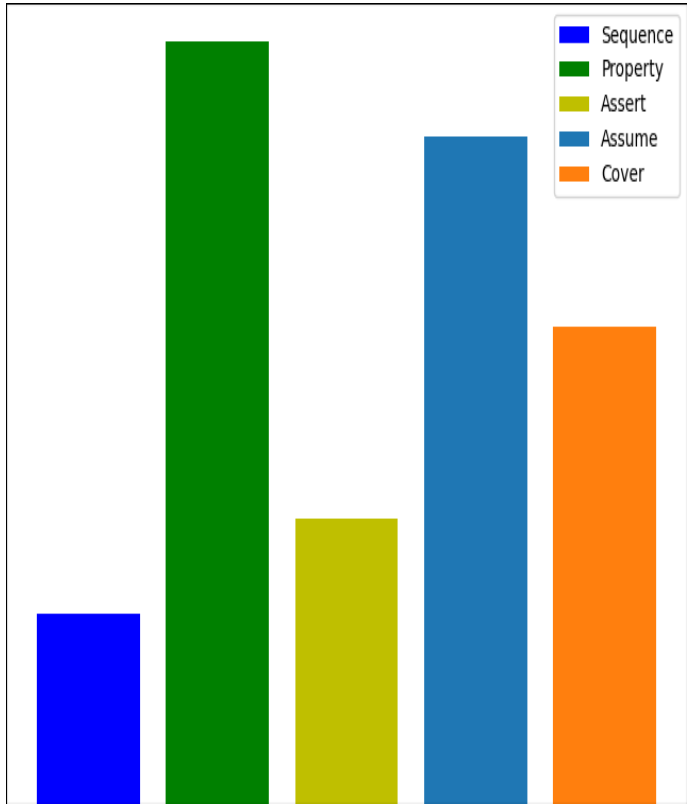
> PySlint: Violation: [FUNC_CNST_MISSING_CAST]: Potentially incorrect constraint expression! An expression involving array-reduction method sum() was found, but is missing an explicit cast. This can lead to strange results as array reduction methods return an expression of the size of its elements, check if you need a with (int'(cast around the following expression:
> num_list.sum()

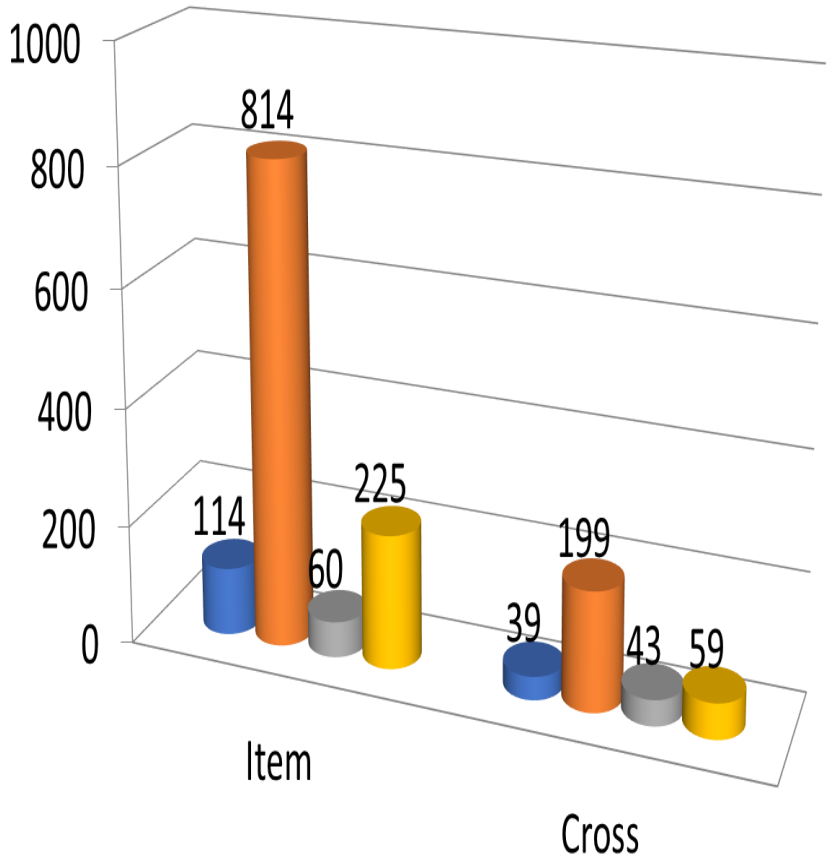
As Figo open-source driven

BROADCOM

Quantitative analysis – SV TestBenches







As Figo
open-source driven

BROADCOM

Testbench Linting - summary

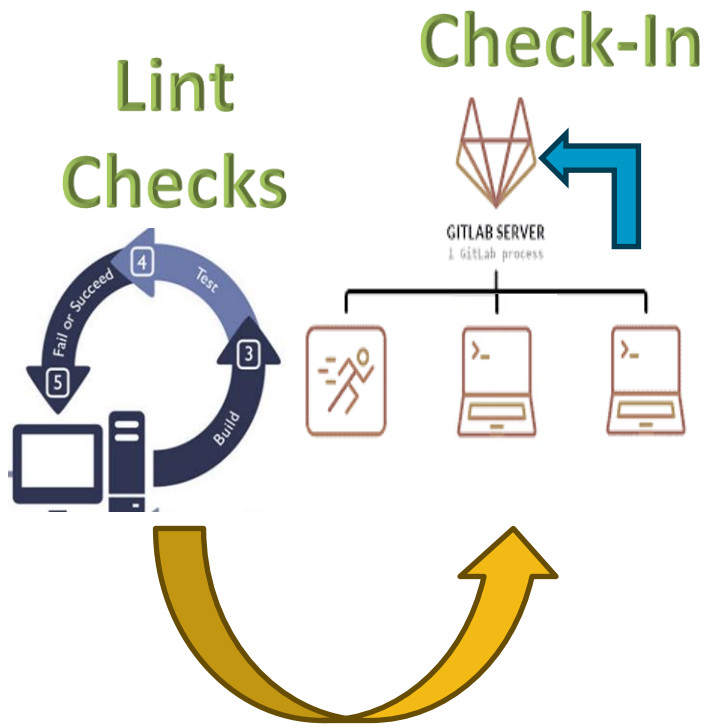
- Exciting new frontiers
- Opensource can now lead the way for commercial tools to follow
  - Than just playing a catch-up game
  - Faster turn-around time
  - Potentially vast set of developers!
- Invite-to-innovate:
  - What analysis would you build on top?

As Figo open-source driven

BROADCOM

Integrate Lint to CI/CD pipeline flow

- Subset of Lint rules
- Customize as per project stage/phase
- Stricter as it gets mature
- Applies to RTL & TB
- Ideally use open-source tools
- Free of license restrictions
- Easy to customize
- Commercial license models available (MIT)



As Figo
open-source driven

BROADCOM

Pyslint approach

- Python based SystemVerilog Linter built based on slang/pyslang(SV parser)
- Help customers build on top with:
  - Own rules (Customers own the rules, proprietary if need-be)
  - Customize existing rules
  - Newer analysis tools/apps
- Not intended as a traditional EDA product!
  - Rather open room for innovation

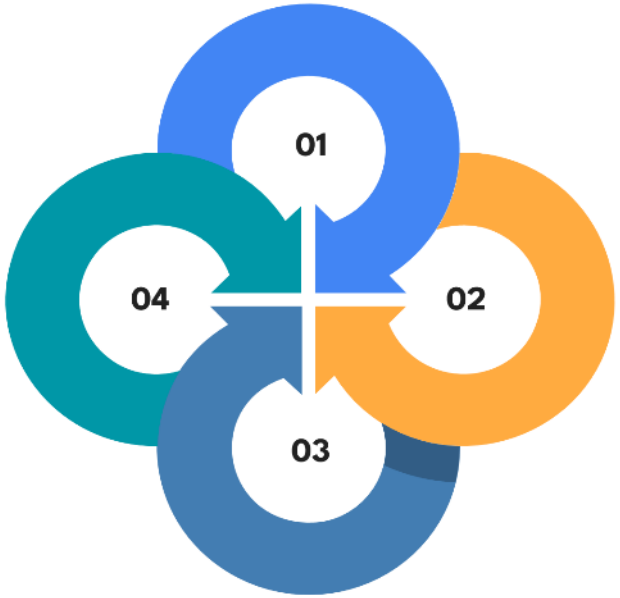
- Qualitative & Quantitative metrics
  - Unique in Lint/static analysis
- Quick Setup & Less dependencies
  - Pyslang (open-source)
  - make
  - Python 3.xx
- Many style check rules now available
  - SystemVerilog TestBench rules
- Introducing UVMLint (UVM Linter)
  - Linter for UVM code
  - UVM rules check now available via GitHub!

BROADCOM

Conclusion

Integration with CI/CD flows helps in continuous improvement

RTL & TB linting and code analysis are essential for quality digital design



Adopting new approaches and alternatives is essential for evolving languages and paradigms

Using popular tools and alternatives improve code reliability

As Figo
open-source driven

BROADCOM