What is the Quality of Your DV Environment?

Wayne Yun
Advanced Micro Devices Inc.

October 8, 2014
SNUG Canada

Agenda

Impact of DV Environment Quality

Finding a Metric

Answer from Synopsys Certitude

Experience on a Block-level TB

Conclusions

Impact of DV Environment Quality

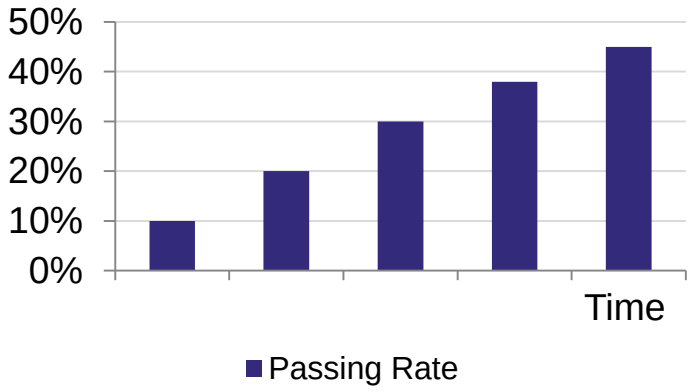
Impact of DV Environment Quality

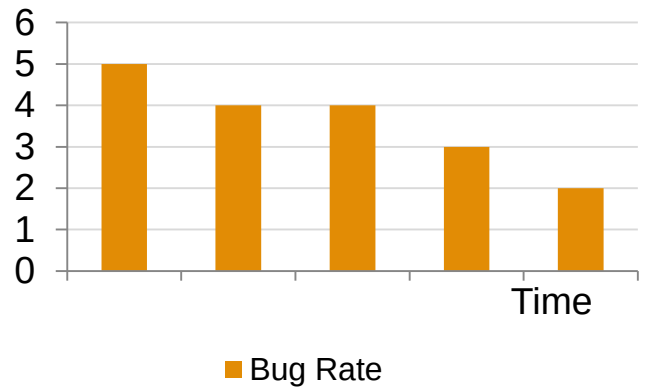
- DV is to improve quality of design
- Silicon bugs do show up though DV metrics were excellent
- Post-mortem often finds weakness in DV environment
  - A checker or assertion was disabled
  - A scenario was missed in test plan, etc.
- A metric or indicator is needed to
  - Measure quality of a DV environment
  - Improve DV and design

Finding a Metric

Metrics of Test Passing Rate and Design Bug Rate

- Test passing rate
  - number_of_passing_tests/ number_of_ total_tests
  - NOT indicating missing scenario
  - NOT reflecting checkers working or not
- Design bug rate
  - Number of design bugs found per week or other interval
  - NOT indicating missing scenario
  - NOT reflecting checkers working or not



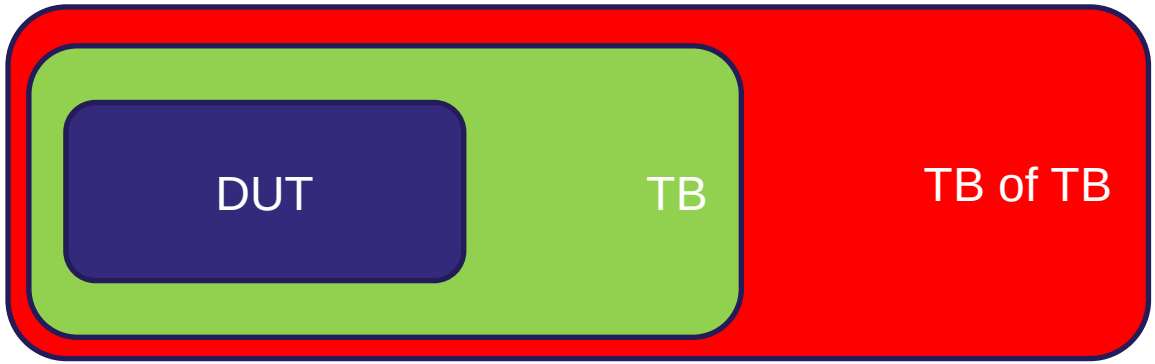


Metrics of Functional Coverage and Code Coverage

- Functional Coverage
  - Records the fact that a scenario defined in Verification Requirements occurred
  - Anything missing in Verification Requirements is also missing
  - NOT reflecting checkers working or not
  - Scenarios not checked become false positive
- Code coverage
  - Records execution of source lines, branches, conditions, toggles , etc.
  - In real projects, code coverage is often not driven to 100%
  - Even if 100% is achieved, there still could be missing design feature and incorrect DV checking

Number of Silicon Bugs and Finding a Metric

- Number of silicon bugs
  - Reflect DV quality
  - It is too late
  - Number of them is too small
  - Too expensive
  - More suitable for conformational metric
- It is not surprising that DV metrics are not indicative of DV quality
  - They are engineered to reflect design quality
  - Design is not purposely structured to find DV environment bugs



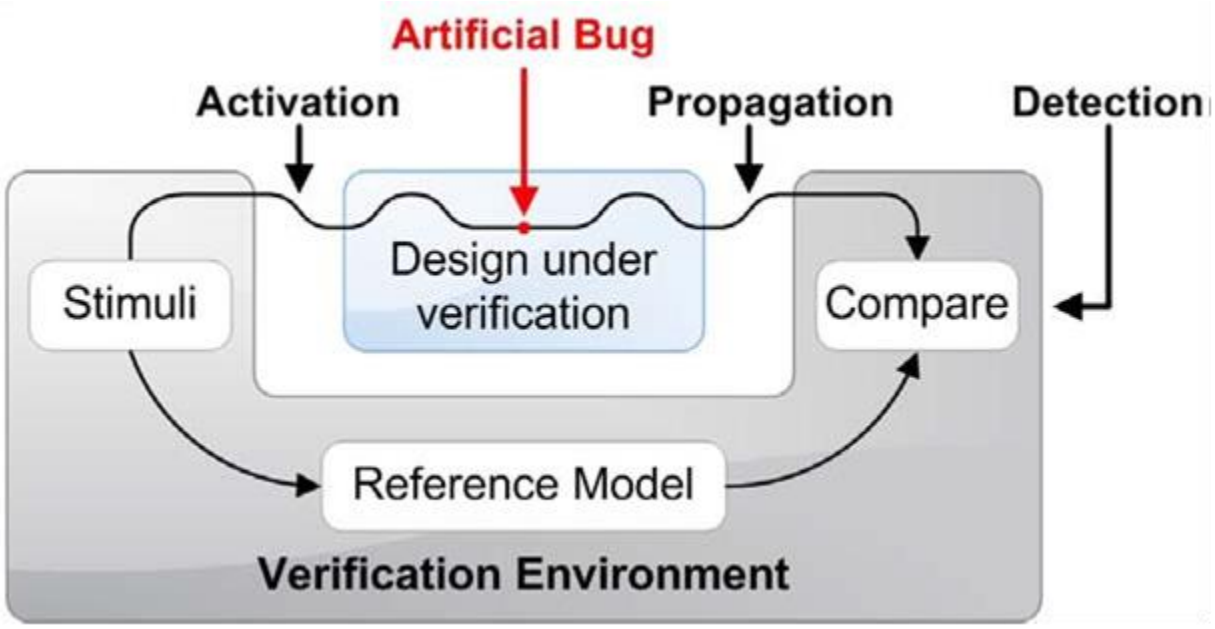
Calibration Weights and Measuring DV Quality

- Precision of scale is checked by calibration weights
  - Scale can measure, but weights cannot

- Quality of DV environment can be checked by “Calibration Designs”
- Made by injecting faults/bugs into good design, and tests are expected to fail with them
- Many “calibration designs” are needed to be statistically meaningful
- Certitude automatically generates faults and calculates quality metrics

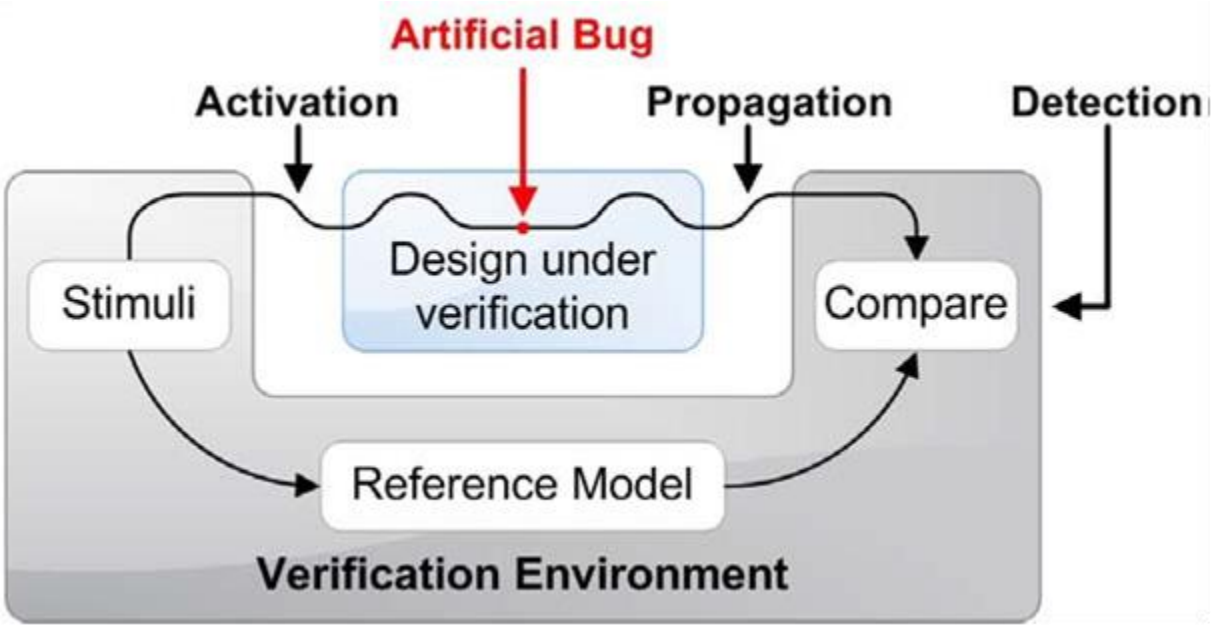
Answer from Synopsys Certitude

Certitude Bug Finding Model



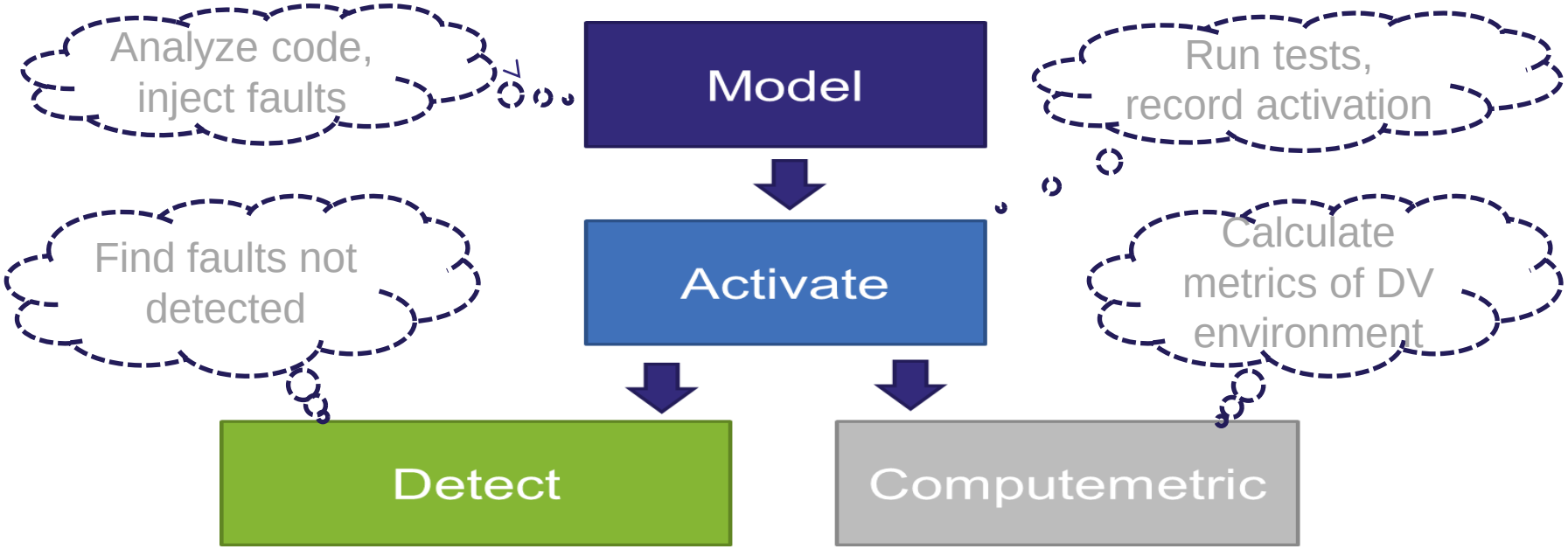
- 3 steps to find a bug
  - Activation – stimulus has to reach the bug
  - Propagation – effect of bug has to propagate to output ports
  - Detection – change of behavior at ports has to be detected and reported by DV environment

Types of Faults Certitude Finds



- 3 types of faults named by the failing step
  - Non-activated fault – stimulus did not reach the bug
  - Non-propagated fault – activated but effect unobservable
  - Non-detected fault – activated and propagated but not caught by verification environment

Certitude Working Model



- Every test is run without fault at Activate
- One fault is injected a time at Detect and Computemetric, only tests activating the fault are chosen to run
- Metrics calculated at Computemetric indicate quality of DV environment

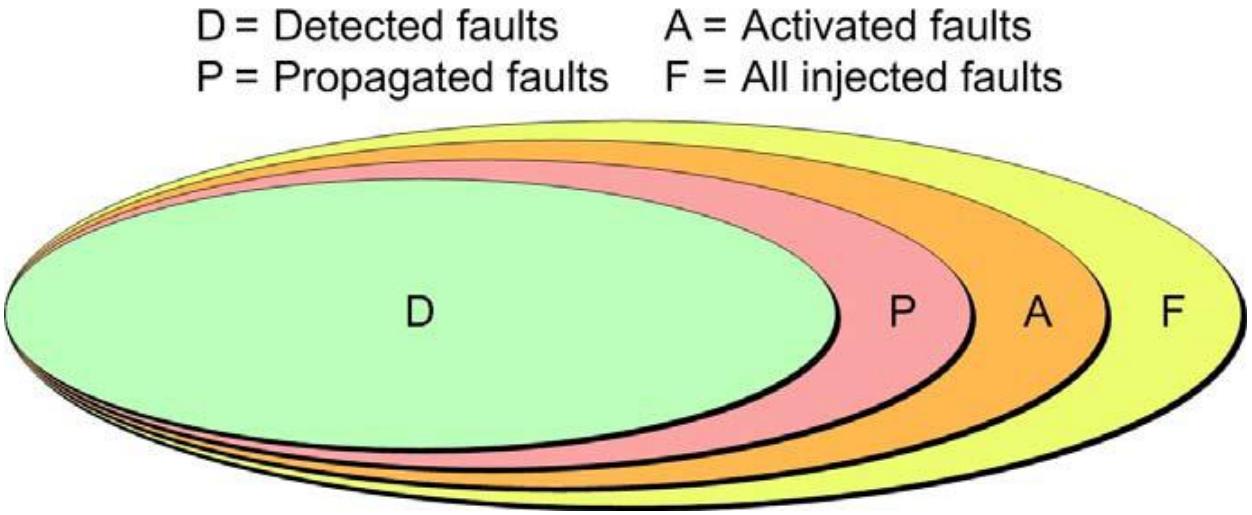
Certitude Report

Synopsys Users Group

Class Name	Faults In Design	Faults In List	Non-Activated	Non-Propagated	Detected	Non-Detected	Disabled By Certitude	Disabled By User	Dropped	Not Yet Qualified
TopOutputsConnectivity	6	6	0	0	1	2	0	0	3	0
ResetConditionTrue	1	1	0	0	0	0	0	0	1	0
InternalConnectivity	0	0	0	0	0	0	0	0	0	0
SynchronousControlFlow	3	3	0	0	0	0	0	0	3	0
SynchronousDeadAssign	2	2	0	0	0	0	0	0	2	0
ComboLogicControlFlow	0	0	0	0	0	0	0	0	0	0
SynchronousLogic	1	1	0	0	0	0	0	0	1	0
ComboLogic	2	2	0	0	0	0	0	0	2	0
OtherFaults	7	0	0	0	0	0	0	0	0	0
All Fault Classes (9)	**22**	**15**	**0**	**0**	**1**	**2**	**0**	**0**	**12**	**0**

- Above is summary table from Certitude report
- Different faults are color coded on top of source code
- Source code of fault is available
- List of tests run for specific fault

Certitude Metrics



- Global Metric = D/F in percentage
- Activation Score = A/F, low indicates insufficient stimulus
- Propagation Score = P/A, more DUT setups to improve
- Detection Score = D/P, mostly related to checkers
- Global Metric = Activation * Propagation * Detection

Experience on a Block-level Testbench

Certitude Setup (1/2)

- DUT is about 25,000 flip-flops, regression runs 1,200 tests
- Certitude needs 5 setup files
  - Certitude Configuration, file certitude_config.cer

```
# set simulator to VCS
setconfig -Simulator=vcs
# set top module name of design
setconfig -TopName=blockA
```

  - HDL File List, file certitude_hdl_files.cer

```
# Verilog HDL is compiled as SystemVerilog
# Faults should be injected to this file
addsystemverilog -file=/pathto/blocka.v –qualify
```

  - Test Cases, file certitude_testcase.cer

```
addtestcase -testcase=testname1
…
```

Certitude Setup (2/2)

- Certitude needs 5 setup files (Cont'd)
  - Compile Script, file certitude_compile
  - Execution Script, file certitude_execute

```
# Command to run test
simv +testname=test1 …
# Pass test result to Certitude
if (testfailed) then
  echo "Test $1 failed"
  $CER_SIMULATION_RESULT -result=Fail
else
  echo "Test $1 passed"
  $CER_SIMULATION_RESULT -result=Pass
Endif
```

Certitude Metrics

- “computemetric” ran about 28,000 tests, sampled 300 faults, and gave metrics

```
**** Scores ****

All percentages and margins of error are valid 95 times out of 100
(confidence level).

Global metric       (D/F) : 57.76 % +- 4.43 %
Activation score    (A/F) : 85.41 %
Propagation score   (P/A) : 71.87 % +- 4.89 %
Detection score     (D/P) : 94.08 % +- 3.21 %
```

- Note: Create new DUT setups, add more probes

Non-detected Faults

- First 5 non-detected faults took about 7 wall-clock hours
- Wall-clock time for each phase

Activation	Metrics	Detect
4.6 hrs	61.6 hrs	192.0 hrs

- “Detect” phase was pushed to find as many non-detected faults as possible, it ran for about 3,200 CPU hours
- Among 45,000 injected faults, found 53 non-detected faults
- While investigating non-detected faults, **seven RTL bugs** were found, **one DV bug** was found

Conclusion

Value Statement

- Certitude can calculate quality metrics of DV environment
  - Not replaced by other widely used DV metrics
  - These metrics indicate areas of potential improvement
- Certitude can also find faults not detected by DV environment
  - By fixing weakness in DV environment, more RTL bugs are found and results in higher quality design and DV
- Best time to run Certitude is when DV environment is stable, and before RTL freeze
- Certitude most likely will find issues when all other metrics are saturated

New Feature Suggestion

- A minor design change most likely happens after a bug is found
- Is it possible to shorten turnaround time? Like,
  - Running a reduced regular regression suite
  - Reusing previously generated faults/results at “model”
  - Only activating affected tests
  - Keeping previous test results of faults, if they are still relevant, such “detect” and “computemetric” can be quicker
  - Merging activation and reference signature runs

AMD

snug
Synopsys Users Group

Thank You