Beyond SystemVerilog Coverage

Thinh Ngo, Ph.D. Can Tho University, Vietnam

***Abstract*—SystemVerilog coverage enables detailed measurements of constrained randomization, including interested scenarios (i.e., binning). Coverage-driven verification (CDV) has been the methodology of choice for design verification. However, coverage distribution is often not considered an essential factor; coverage fulfillment is reactive, not proactive, and coverage results are available late at post-regression. This research investigates and proposes enhancement extensions for SystemVerilog coverage and CDV. Extensions of SystemVerilog coverage are proposed as followings:**

1) **Extend SystemVerilog coverage capability with probability density function (pdf) reporting.**
2) **Leverage and include coverage as constraints or coverage-driven constraints**
3) **Provide probability-based and pre-regression coverage reporting capability, including covergroups, cover assertions, and code coverage. Additionally, coverage predictions of covergroups can be provided pre-DUT.**

With the above coverage extensions, CDV should apply the principle of coverage first, Coverage-First Verification (CFV), not during or near the end of the verification process. Coverage of covergroups and cover assertions should be implemented, collected, and computed early during or after the verification planning before DUT availability and time-consuming checkers (i.e., SVAs, and scoreboards). This shift allows the discovery and remedy of coverage holes early. This research also explores potential implementations of the above extensions.

I. Coverage PDF

Distributions of coverage should be a concern. Typically, one would expect uniform distributions among coverage specifications unless constrained otherwise. However, the actual distributions can be skewed far from the expected distributions, driven by conditional randomization constraints (e.g., implication and if). Under-coverage with a single or a few occurrences may be insufficient for verification intents. On the other hand, over-coverage may be inefficient, wasting verification resources. Therefore, the ability of simulation tools to report coverage frequency and distribution is recommended.

II. Coverage-Driven Constraints

SystemVerilog constraints do not include global distribution capability. The provided distribution construct is local and not guaranteed. As a result, coverage results are available post-regression, and coverage fulfillment is reactive and late, negatively impacting the verification closure. However, constraints can utilize coverage. Coverage-driven constraints would include pdf-specified coverage as constraints that steer constrained randomization to meet coverage with specified pdf. This capability would expedite coverage closure and increase resource utilization.

III. Probability-based Coverage Reporting

Actual coverage computation is based on regressions, whose individual tests or transactions are randomized, with constraints, using uniform distribution functions. The same coverage computation can be determined probabilistically, a Monte-Carlo approach, without running regressions. This ability to provide coverage reporting pre-regression is precious for early coverage closure. Additionally, coverage of covergroups can be predicted without DUT.

IV. Coverage-First Verification (CFV)

As a result of the above enhancements, coverage first verification (CFV) is recommended to supplement CDV. Testbench incorporates coverage implementation (i.e., covergroup, cover assertion), after verification planning, before checkers (e.g., SVAs and scoreboards), and DUT availability (i.e., for covergroups). Utilizing the Monte-Carlo coverage predictions, the coverage implementation can be executed and debugged to meet the verification plan coverage requirements pre-regressions, including covergroup, cover assertion, and code coverage.

V. Implementation

The best approach to implement extensions is updating the SystemVerilog language. Alternatively, it can be done as extension packages similar to UVM, Python's NumPy, or C++'s Vector. Workarounds have been reported to count coverage of cover assertions; the same can be used to count coverage of covergroups by creating cover assertions for DUT inputs of covergroups. Another way is to encapsulate covergroups in classes with coverage sample counting functions. Coverage-driven constraints can be implemented by encapsulating in classes with functions to steer constrained randomization to meet required pdf-specified coverage dynamically. Finally, coverage can be computed deterministically or via Monte Carlo experiments. The Monte-Carlo coverage predictions can be independently and expeditiously performed using a simple testbench containing random variables, constraints and coverage, without drivers, monitors, or TLM ports, shown at [1]

VI. Conclusion

This research presents proposed enhancements to the SystemVerilog coverage and the coverage-driven methodology. The coverage extensions and the resultant Coverage-First Verification (CFV) would contribute to DV effectiveness and efficiency. This is another effort to shift and release engineering resources from routine or tool-limited workaround coding tasks to engage in higher abstraction and creativity activities. Other recent research papers by the same author are [2], [3], [4], [5], [6], [7].

References

[1] https://github.com/thinhngo11/Coverage-First-Verification
[2] https://www.academia.edu/105351354/Beyond_SystemVerilog_Constraint_and_Randomization, 8/2023
[3] https://www.academia.edu/105351677/Polymorphic_Transaction_Level_Constraint, 8/2023
[4] https://www.academia.edu/105351236/Dynamic_Testbench_using_Polymorphism 7/2023
[5] https://www.academia.edu/88327789/Beyond_UVM, 6/2022
[6] https://www.academia.edu/90144515/On_the_Automation_of_Coverage_Closure, 6/2022
[7] https://www.academia.edu/87381334/Applying_DevOps_Continuous_Integration_CI_in_Design_Verification, 6/2022