Saving and Restoring Simulation Methodology using UVM Factory Overriding to Reduce Simulation Turnaround Time

Ahhyung Shin, Yungi Um, Youngsik Kim, Seonil Brian Choi
Samsung Electronics Co., Ltd., Seoul, Korea

Motivation

Background

- Turn-around Time (TAT) increases according to design complexity
- Not enough time to cover all tests
- Solution in previous studies
  - Enhance simulation performance
    Performance enhancement (10% / year) < Design size/complexity increase
  - Emulation based verification : High cost
- Proposed "Save and Restore" technology
  - Reduction of run-time for each test scenario
  - Reduction of repetitive instructions in test scenario

Analysis test scenarios

- To find basic and commonly used sequences in every test scenario
- System boot-up and module initiation sequences
  - Consume long verification time before actual HW testing
  - Run time of common sequences > that of actual test sequence
  - Operated in every test scenarios regardless of users

Advanced Save and Restore Methodology

- The proposed Save and Restore Methodology
  - Reduction of simulation TAT
  - Independent of test sequence modification
  - Easiest way to reduce verification TAT in UVM environment with very low cost

Typical Save and Restore Methodology

Saving and Restoring Simulation Based on Simulation Time

- Time based SnR : most basic and simple solution
- Saving the state of DUT and test bench at specific time using simulator command and re-load it
- Difficulties in alignment of a simulation time and instruction sequences

Saving and Restoring Simulation with DPI-C and UVM Factory Override Mechanism

- Using DPI-C and Tcl command to perform UVM factory overriding
- Re-loading saved files / Tcl command overrides new UVM sequences those will be executed after the saved point
- New UVM sequences cannot be modified after file saving

Suggested Save and Restore Simulation

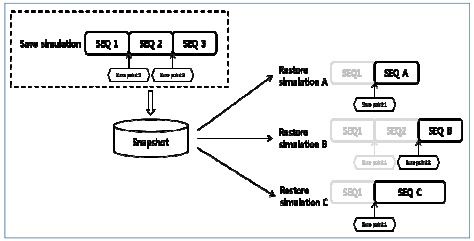
Save Simulation

- Executing the sequences that would be saved
- All states of DUT and test bench are written at a file as a snapshot when sequence reaches the saving point
- The file is called as a "snapshot"

Restore Simulation

- Restore simulation with the snapshot written in a file
- UVM factory overriding is performed during the restoring simulation
- Restoring order
  1. New input file lists are compiled and elaborated
  2. A Tcl command of simulator performs a UVM factory overriding
  3. Start restoring simulation from the saved point in the test scenario
- Modification of a new UVM sequences
  - Input files are parsed, compiled and elaborated at starting point of restoring

  → Dynamic modification of a new UVM sequence should be supported

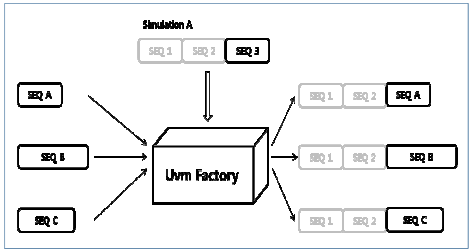


[Dynamic SnR Methodology]

Implementation

UVM Factory Override Mechanism

- UVM Factory
  - Flexible verification environment construction
  - Provided by UVM Class
- Using Built-in UVM Factory
  - Reduces the effort of creating an advanced factory
  - Reduces the effort of Implementing factory methods in class definitions
- Condition for successful UVM sequence factory overriding implementation
  1. Both test sequences to override and to be overridden must be registered to the UVM Factory
  2. Type of sequences must be same because proposed overriding is based on type overriding
  3. UVM sequences which are performed after the save point must be existed in the snapshot

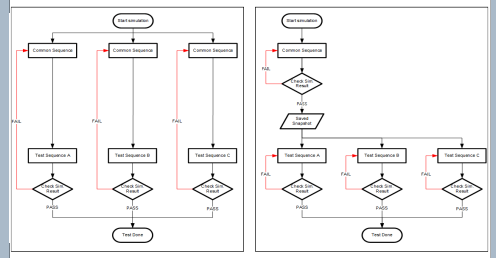


[Basic Concept of UVM Factory Override Mechanism]

SystemVerilog Package

- One of items those compilation-unit scope of SystemVerilog can contain
- The Dynamic SnR method is possible when test bench is described in the SystemVerilog Package
- To make modifiable test sequence, the restoring simulation test sequence should be packaged with SystemVerilog Package
- Restoring Simulation Progress
  1. Test sequence to override must be located inside of a SystemVerilog Package
  2. Restoring simulation receives a file includes the SystemVerilog Package of the test sequence to override as an input
  3. Simulator compiles and elaborated the input file and starts the restoring simulation

Practical SnR Methodology Workflow



(a) Normal verification workflow w/o SnR (b) Proposed verification workflow w/ SnR

[Proposed Workflows with SnR Methodology]

Result

Experiment Configuration & Result

- Two functional blocks are used for comparison
- Test scenarios of each block includes SOC boot-up sequences and PHY initialization sequences commonly
- Total number of test cases: 30 per each block
- Saving points called as "check_point"
  - For Block-A, "check_point" is set at end of boot-up sequences
  - For Block-B, "check_point" is set at end of PHY initialization

[Table 1. TAT Reduction Result of the proposed SnR Methodology]

	***Block-A (Booting seq)			***Block-B (Booting seq + PHY init)		
	Normal Sim.	Restore Sim.	TAT Reduction	Normal Sim.	Restore Sim.	TAT Reduction
Simulation Time (min)	2507	303	88%	16161	537	97%

- **Block-A: TAT is reduced by 88%**
- **Block-B: TAT is reduced by 97%**

Conclusion

All test scenarios have common sequences

Almost common sequences can be skipped by the propose SnR Methodology

Dramatic TAT reduction was confirmed with the propose methodology

Suggested application by the proposed methodology

- Power aware Simulation
- Gate-Level Simulation
- Central Regression Test

Contact information

Email:ah0403.shin@Samsung.com
Samsung Electronics Co., Ltd., Seoul, Korea