

Mastering SystemVerilog UVM (Universal Verification Methodology)

Overview

The Accellera Universal Verification Methodology (UVM) standard defines a methodology for using SystemVerilog for the verification of complex designs. UVM enables engineers to write thorough and reusable test environments. UVM is a robust methodology with many advanced features. In this *Mastering UVM* workshop, engineers will learn to apply the UVM for transaction level verification, constrained random test generation, coverage, and scoreboarding. Topics include UVM test phases, UVM class libraries, UVM utilities, UVM factories, UVM sequencers, UVM drivers, UVM Monitors, UVM scoreboards, UVM registers, and configuring UVM tests. Several labs reinforce the concepts presented during the course. NOTE: To benefit from this workshop, engineers must already have a good understanding of the SystemVerilog language (such as from the SystemVerilog Verification Foundations workshop)

Course length: 3-days on-site. 4-days *eTutored™ live*. 3 to 60 days *eTutored™ self-paced*.

Intended Audience and Objectives

This workshop is for design and/or verification engineers who will be creating test environments for the verification of digital designs. At the conclusion of this workshop, engineers will understand how to take full advantage of the verification capabilities in the SystemVerilog language in order to develop object-oriented testbenches that utilize constrained-random verification methodologies, functional coverage, mailboxes and scoreboarding.

Prerequisites (essential)

A working knowledge of SystemVerilog is essential in order to fully benefit from this workshop. In order to fully understand and utilize the concepts presented in this course, students should have first completed the Sutherland HDL *SystemVerilog Verification Foundations* course or equivalent.

Included Materials

- Comprehensive binder with all PowerPoint slides (printed in color), lab instructions, and supplemental white papers. (*eTutored™ self-paced* courses use an eBook and other online materials instead of a training binder.)
- A handy “*Verilog HDL Quick Reference Guide*” (\$15 value).
- All lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

The Cadence *Incisive™*, Synopsys *VCS™* or the Mentor Graphics *Questa™* simulator will be used for labs.

Workshop Locations

This workshop can be presented on-site at your facilities or as an *eTutored™ live* online class. We also offer several public *eTutored™ live* workshops throughout the year. For more information, please visit www.sutherland-hdl.com, or call us at +1-503-692-0898.

Syllabus — Mastering UVM (Universal Verification Methodology)

Day One

UVM Overview

- The purpose of UVM
- Transaction based verification
- UVM testbenches
- UVM test phases
- UVM objects and components
- Lab: Examining a UVM testbench

UVM First Look

- uvm_object class
- uvm_component class
- UVM class library
- UVM utilities
- UVM configuration
- Lab: Defining a simple UVM testbench

Review of SystemVerilog Verification Constructs

- SystemVerilog's class data type
- Class methods
- Class inheritance
- Virtual methods
- Virtual classes
- Constrained random value generation
- Functional coverage
- Lab: Creating a simple OO testbench

UVM Transactions and Sequences

- Transaction Level Modeling (TLM)
- TLM ports, exports, and analysis ports
- Sequences of transactions
- UVM sequencers
- Connecting sequences to drivers
- Lab: Defining a sequence of transactions

Day Two

UVM Drivers, Monitors and Agents

- UVM drivers
- Virtual interfaces
- UVM monitors
- UVM agents
- Active and passive modes
- Lab: Defining and configuring a UVM agent

UVM Environments, Predictors and Scoreboards

- Scoreboard fundamentals
- Predicting expected results
- Comparing expected and actual results
- Encapsulation in a test environment
- Lab: Verifying outputs of a (faulty) DUT

UVM Tests and Top-level Testbenches

- Putting everything together in a UVM test
- Top-level modules
- Running multiple tests
- Using SystemVerilog program blocks with UVM
- Reuse and scalability considerations
- Lab: Define and run a series of UVM tests

UVM Factories

- Understanding the UVM factory
- Registering verification components
- Using the UVM Configuration database
- UVM messages and reports
- Lab: Working with factories

Day Three

UVM Functional Coverage

- Types of coverage
- Where coverage can be specified
- Enabling coverage
- Lab: Defining and examining coverage

UVM Compound Sequences

- Sequences of sequences
- Layered sequencers
- Virtual sequencers
- Configuration of compound sequences
- Lab: Working with layered sequencers

UVM Register Layer

- Register packages
- Registers and register files
- Memories
- FIFOs
- Register stimulus generation
- Backdoors
- Lab: working with registers