SUTHERLAND HDL

22805 SW 92nd Place., Tualatin, Oregon 97062 • Phone: +1-503-692-0898 • URL: www.sutherland-hdl.com

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 SystemVerilog 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. The UVM 1.1 and 1.2 standards are presented in the class discussions. 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 and object-oriented programming (such as from the Sutherland HDL *SystemVerilog Object-Oriented Verification* training workshop).

Course Length:

- As a stand-alone workshop for engineers who are already familiar with SystemVerilog object-oriented programming:
  3-days on-site, 4-days *eTutored*™ *live*, 2 to 30 days *eTutored*™ *self-paced*.
- Combined with Sutherland HDL's *"SystemVerilog Object Oriented Verification"* workshop:
  5-days on-site (not offered as an online *eTutored*™ *live* workshop).

Intended Audience and Objectives

This workshop is for verification engineers who will be using UVM to code complex testbenches and stimulus for digital designs. After completing this workshop, engineers will know the types of verification components that make up a UVM testbench, transaction-level modeling, the proper ways to use UVM phasing and objections, and how to ensure a UVM testbench will work correctly with the UVM 1.1 and 1.2 standards.

Prerequisite Knowledge (essential)

Attendees ***must*** be familiar with SystemVerilog programming, including object oriented programming techniques, constrained random value generation, interfaces and clocking blocks. Engineers attending this workshop are expected to already have a strong foundation in the SystemVerilog language. This foundation can come from prior experience with SystemVerilog, or by completing Sutherland HDL's *"SystemVerilog Object Oriented Verification"* workshop.

Included Materials

- Full-color training binder with copies of all lecture slides, lab instructions, and supplemental information. (*eTutored*™ *self-paced* courses include an eBook instead of a training binder.)
- Lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

The Aldec *Riviera-Pro*™, Cadence *Incisive*™, Synopsys *VCS*™ or the Mentor Graphics *Questa*™ simulator can 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.

Licensed Training Materials

Sutherland HDL's training materials can be licensed for use in internal training programs. Licensed materials include presentation PowerPoint files, printable PDF files, and lab files. Train-the-trainer services are also provided.

Syllabus — Mastering SystemVerilog UVM

UVM Overview
- The purpose of UVM
- UVM testbench architecture
- UVM test phases
- UVM objects and components
- Lab: Simulate a simple UVM testbench and DUT

Rapid Review of SystemVerilog Object-Oriented Verification
- SystemVerilog's class data type and new() constructors
- Inheritance, data hiding
- Virtual methods and polymorphism
- Specialized (parameterized) classes
- Object handle assignments and down casting
- Constrained random value generation
- Functional coverage

UVM First Look
- UVM class library
- uvm_object class
- uvm_component class
- Registering UVM components with the factory
- Virtual interfaces and connecting to the DUT
- UVM print and debug utilities
- Lab: The big picture — examine all the parts of a complete UVM testbench

UVM Sequence items and Sequences
- UVM sequence_items (transactions)
- Defining sequence_item methods
- Using sequence_item field macros
- UVM sequences of transactions
- Sequence/Driver synchronization
- Lab: Define and simulate sequence_items and sequences

UVM Sequencers and Drivers
- UVM sequencers
- UVM drivers
- Transaction Level Modeling (TLM)
- TLM ports, exports, and analysis ports
- Lab: Define and simulate a UVM driver and sequencer

UVM Monitors and Agents
- UVM monitors
- Adding one or more monitor analysis ports
- UVM agents
- Agent active and passive modes
- Lab: Defining and simulating a UVM monitor and agent

UVM Functional Coverage
- A review of SystemVerilog functional coverage
- Coverage collectors
- Where to add coverage collectors
- Enabling and disabling coverage collectors
- Lab: Define, simulate, and examine coverage

UVM Environments, Predictors and Scoreboards
- Scoreboard fundamentals
- Predicting expected results
- Comparing expected and actual results
- Encapsulation in a test environment
- Lab: Define and simulate a UVM scoreboard and environment, and verifying the outputs of a (faulty) DUT

UVM Tests and Advanced Sequences
- Putting everything together in a UVM test
- Running multiple tests
- Virtual sequences and sequencers
- Sequential and parallel sequences
- Sequencer arbitration modes
- Layered sequences
- Driver to sequence feedback
- Top-level modules
- Lab: Define and simulate a test that runs multiple sequences

UVM Factory and UVM Configuration
- Understanding the UVM factory
- Registering and constructing verification components
- Factory overrides
- Using the UVM Configuration database
- Reuse and scalability considerations
- UVM messages and reports
- Lab: Define and simulate a configurable UVM test environment

UVM Register Layer Overview
- When and where to use verification registers
- Register packages
- Registers and register files
- Bus translators
- Back door access
- Front door access
- Register stimulus generation