SUTHERLAND HDL

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

SystemVerilog Object-Oriented Verification

Overview

Getting the most benefit from advanced verification methodologies such as UVM, OVM and VMM requires understanding the SystemVerilog constructs on which they are built. ***SystemVerilog Object-Oriented Verification*** provides that knowledge. Concepts presented include special testbench modeling constructs, clocking domains, object oriented verification, constrained random verification, coverage, an overview of SystemVerilog assertions, and an overview of the SystemVerilog Universal Verification Methodology (UVM). Engineers learn how to utilize object inheritance and polymorphism, mailboxes, semaphores, specifying randomization constraints, specifying functional coverage, and dynamic arrays. The workshop also covers the SystemVerilog programming and operator constructs and explains how these constructs are properly used in both testbenches and hardware models. Language subtleties such as blocking and nonblocking assignments and how to detect and correct simulation race conditions are discussed. Labs reinforce the concepts learned, with forty percent of the class time devoted to hands-on experience.

Course Length:

- As a stand-alone workshop for engineers who are already familiar with Verilog or SystemVerilog: 4-days on-site, 5-days *eTutored™ live*, 2 to 30 days *eTutored™ self-paced*.
- Combined with Sutherland HDL's *"Verilog and SystemVerilog Primer"* workshop: 5-days on-site, 5-days *eTutored™ live*.

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.

Prerequisite Knowledge (essential)

Attendees ***must*** have a working knowledge of the Verilog and/or SystemVerilog language. This familiarity can come from prior experience with Verilog, or by completing the *"Verilog and SystemVerilog Language Primer"* 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.

Quotes From Students

"Best workshop I've attended. Very informative and insightful into what SystemVerilog offers."

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.

(Company names and product names are trademarks or registered trademarks of their respective companies.)

SUTHERLAND HDL

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

Syllabus — SystemVerilog Object-Oriented Verification

Introduction to Verilog and SystemVerilog
- Overview and history of Verilog and SystemVerilog
- Synthesis and verification language subsets
- A simple Verilog test bench
- Lab: running simulations

Verilog and SystemVerilog Syntax and Semantics
- Identifier names
- Logic values and literal values
- Verilog and SystemVerilog data types
- Lab: Verification with 2-state data types

Procedures, Programming Statements and Operators
- Procedural blocks
- Tasks and functions
- Procedural assignments (blocking and non-blocking)
- Continuous assignments
- Programming statements and operators

Modeling RAMs and ROMs
- Modeling memories
- Loading programs into memory models
- Lab: model and verify a single-port SRAM

SystemVerilog User-defined Types and Packages
- User-defined types and enumerated types
- Structures and unions
- Casting
- Packages and $unit
- Lab: Model and verify an Instruction Stack

SystemVerilog Interfaces
- Using interfaces to simplify inter-module connections
- Specifying interface views (modports)
- Using tasks and functions in interfaces
- Using interfaces between the testbench and the DUT
- Lab: model and verify a master/slave interface bus

Verilog Verification Constructs
- Configurable test benches
- Structured tests
- Reading and Writing data files
- Lab: verify a design using test vectors

Program Blocks and Clocking Domains
- Program blocks
- Clocking domains
- Final blocks
- Lab: Developing a test program

Object-Oriented Programming, Part One
- SystemVerilog's class data type
- Defining class objects
- Class methods
- Class inheritance
- Lab: Creating a simple OO testbench

Object-Oriented Programming, Part Two
- Extending class definitions (inheritance)
- Virtual methods
- Virtual classes
- Public and private classes
- Lab: Creating an advanced OO testbench

Dynamic Arrays and Scoreboards
- Dynamic arrays
- Associative arrays
- Queues
- Strings
- Lab: Create a scoreboard using dynamic arrays

Process Synchronization
- Fork—join dynamic processes
- Built-in mailbox classes
- Built-in semaphore classes
- Enhanced event data types
- Lab: Using mailboxes for verification

Constrained Random Value Generation
- Built-in SystemVerilog random classes
- Defining constrained random values
- Constrained random verification methodologies
- Lab: Using constrained random test values

Functional Coverage
- Defining and constructing cover groups
- Defining cover points and coverage bins
- Coverage sampling
- Cross coverage
- Lab: Using coverage with constrained random tests

Overview of SystemVerilog Assertions
- Assertion concepts
- Immediate and concurrent assertions
- Assertion sequence definitions

Overview of SystemVerilog UVM
- Importance of verification methodologies
- UVM concepts
- UVM verification components