

SystemVerilog Verification Foundations

Overview

Getting the most benefit from advanced verification methodologies such as UVM, OVM and VMM require understanding the SystemVerilog constructs on which they are built. *SystemVerilog Verification Foundations* provides that knowledge. Concepts presented include special testbench modeling constructs, defining clocking domains, Object Oriented verification, constrained random verification, coverage, and an overview of SystemVerilog assertions. Engineers learn how to utilize dynamic arrays, dynamic processes, object oriented inheritance and polymorphism, mailboxes, semaphores, specifying randomization constraints, specifying functional coverage. The course 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 is discussed. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

Course length: 4-days on-site. 5-days *eTutored™ live*. 5 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)

Knowledge of digital hardware concepts is required, such as an understanding of combinational and sequential logic and binary arithmetic. Without this background, students cannot fully benefit from this course. Familiarity with other languages, such as VHDL, VERA, e, or SystemC is helpful but not required.

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.

Quotes From Students

“The instructor was clearly an expert and extremely good at presenting the material.”

“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.

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

Syllabus — SystemVerilog Verification Foundations

Day One

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
- Lab: model and verify an 8-bit ALU

Verilog Structural Models

- Module instantiation
- Parameterized models
- Generate blocks
- Lab: Using instance arrays and generate blocks

Day Two

Modeling RAMs and ROMs

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

Verilog Verification Constructs

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

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

Program Blocks and Clocking Domains

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

Day Three

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

Day Four

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 cover groups
- Defining cover points
- Defining cover bins
- Cross coverage
- Lab: Using coverage with constrained random tests

Assertions for Verification Engineers

- Assertion concepts
- Concurrent assertions and sequences
- Basic sequence definitions
- Disabling assertions during reset
- Controlling assertion messages
- Binding assertions to design blocks
- Lab: Using assertions at the testbench level