

SystemVerilog Testbench for non-Verilog Verification Engineers

Overview

SystemVerilog Testbench for non-Verilog Verification Engineers is a 4-day intense workshop covering both the Verilog language and the advanced testbench constructs in the SystemVerilog standard.

The focus of this workshop is on the verification aspects of Verilog and SystemVerilog. Concepts presented include Verilog data types and programming constructs, along with the plethora of SystemVerilog testbench constructs. The SystemVerilog verification constructs that are presented include dynamic arrays, dynamic processes, Object Oriented verification, mailboxes, semaphores, generating constrained random stimulus, functional coverage, and an overview of SystemVerilog assertions. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

SystemVerilog Testbench for non-Verilog Verification Engineers combines topics from Sutherland HDL's *Accelerated Verilog Primer* and the *SystemVerilog Testbench for Verilog Verification Engineers* workshops. Course materials include a *comprehensive student guide*, and a "*Verilog HDL Quick Reference Guide*" (\$15 value).

Course Objectives

At the conclusion of this workshop, engineers will understand how to take full advantage of the verification capabilities in the Verilog language combined with SystemVerilog to develop object-oriented testbenches that utilize constrained-random verification methodologies, functional coverage, mailboxes and scoreboarding.

Intended Audience

This workshop is for design and/or verification engineers who are familiar with other languages, such as VHDL, VERA, e, or SystemC. The workshop provides a foundation in both Verilog and SystemVerilog, with a focus on using these languages for verification of digital designs.

Prerequisites (essential)

Knowledge of digital hardware concepts is required. Without this background, students cannot fully benefit from this course. This course assumes an understanding of digital concepts such as combinational and sequential logic, setup and hold times, and binary arithmetic.

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. We also offer several open-enrollment workshops throughout the year. For more information, please refer to our web page, www.sutherland-hdl.com, or call us at +1-503-692-0898.

Syllabus — SystemVerilog Testbench for non-Verilog Verification Engineers

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