

Sutherland HDL offers expert-level training workshops on the Verilog and SystemVerilog hardware design and verification languages. These workshops bring years of experience to the classroom. Each workshop is developed and presented by design and verification engineers who are experts in their fields and who excel in teaching.

“This class is excellent, from materials to instructor. The instructor was able to convey heavy technical material and still make the class fun and full of energy.” — Sun Microsystems

“I learned more in this workshop than I could have learned in years of using Verilog” — Accelerant Networks

“The instructor's expert knowledge provides insight not possible from other courses.” — Hewlett-Packard

On-site Workshops

On-site workshops are held at your facilities, and can be customized to meet the needs of your company. All that is required is a conference room. Sutherland HDL can provide a portable lab environment, with computers, simulation and synthesis software. On-site workshops have a six student minimum.

Open-enrollment Workshops

Sutherland HDL conducts open-enrollment workshops at several locations in the United States. These are open to any number of students from several companies. Sutherland HDL provides the training facilities, computers, and the simulator and synthesis software. Visit www.sutherland-hdl.com for the current workshop schedule.

Sutherland HDL Workshop Offerings

Workshops for Design Engineers:

- **Comprehensive Verilog and SystemVerilog for Design and Synthesis** (4 days)
- **SystemVerilog Synthesis for Verilog Design Engineers** (2 days)
- **Advanced SystemVerilog Assertions for Design and Verification Engineers** (2 days)
- **Accelerated Verilog Primer** (2 days)

Workshops for Verification Engineers:

- **SystemVerilog Testbench for Verilog Verification Engineers** (3 days)
- **SystemVerilog Testbench for non-Verilog Verification Engineers** (4 days)
- **Advanced SystemVerilog Assertions for Design and Verification Engineers** (2 days)
- **Advanced SystemVerilog Open Verification Methodology (OVM)** (3 days)
- **Practical Application of the Verification Methodology Manual (VMM)** (1-day, in development)

Specialty workshops:

- **Advanced Verilog PLI 2.0 (VPI) with SystemVerilog DPI** (2 days)
- **Comprehensive VHDL for Synthesis and Verification** (4 days)

Contact Sutherland HDL at **1-503-692-0898** to enroll in an open-enrollment workshop, or to discuss holding an on-site workshop at your company.

Sutherland HDL Instructor Biographies

STUART SUTHERLAND

Stuart Sutherland is the founder and a principal engineer of Sutherland HDL, Inc., located in Portland, Oregon, where he provides expert SystemVerilog, Verilog and Verilog PLI design services, and presents advanced level training workshops. Mr. Sutherland has provided expert training at companies throughout the world, including the United States, Canada, England, Germany, Sweden, Japan, Malaysia, and Hong Kong. He brings more than 25 years of experience in hardware design and verification, and over 20 years experience with Verilog to the classroom. Prior to founding Sutherland HDL, Mr. Sutherland worked as an engineer for Sanders Display Products Division in New Hampshire, where he worked on high-speed graphics systems for the defense industry. In 1988, he became a senior applications engineer for Gateway Design Automation, the founding company of Verilog. Mr. Sutherland holds a Bachelor of Science in Computer Science with an emphasis in Electronic Engineering Technology from Weber State University (Utah) and Franklin Pierce University (New Hampshire). He is the co-author of the book "SystemVerilog for Design Engineers", and is the author of the "*The Verilog PLI Handbook*", and the popular "*Verilog HDL Quick Reference Guide*" and "*Verilog PLI Quick Reference Guide*". Mr. Sutherland is actively involved in the IEEE 1364 Verilog standardization working group and the IEEE 1800 SystemVerilog standardization working group.

DON MILLS

Don Mills, of LCDM Engineering, Inc., is an independent contractor who presents training courses for Sutherland HDL. Mr. Mills is a senior hardware design and verification engineer with over 20 years of experience; During this time he has worked on numerous ASIC's including CMOS devices, ECL devices (137 MHz and 274 MHz), and a high voltage (15 volt) mixed analog/digital device. He is proficient in both Verilog and VHDL. He has developed and implemented top-down ASIC design flows for a number of companies. Mr. Mills is an outstanding instructor, with a unique talent for presenting complex topics in a clear and entertaining manner. Mr. Mills holds a Bachelor of Science in Electrical Engineering from Brigham Young University, and is a member of IEEE. He has worked for L3 Communications, US Robotics, Honeywell and a number of other companies. He has also served as Technical Chair for the 1998, 1999 and 2000 Synopsys Users Group (SNUG) Conference, and as Technical Chair for the 2001 and 2002 European SNUG Conference. Mr. Mills has authored several papers, including "*How to Synthesize a Million Gate ASIC*" for the 1997 SNUG Conference and co-authored "*RTL Coding Styles that Yield Simulation and Synthesis Mismatches*" for the 1999 and 2001 SNUG Conferences.

Comprehensive Verilog and SystemVerilog for Design and Synthesis

Overview

Comprehensive Verilog and SystemVerilog for Design and Synthesis is a packed 4-day workshop covering the complete IEEE Verilog Hardware Description Language standard, including the synthesizable portions of the SystemVerilog-2009 extensions to Verilog. All aspects of Verilog are presented, with a focus on using Verilog for top-down design with synthesis and simulation. The synthesizable subset of the SystemVerilog extensions are integrated into the course topics. These powerful extensions include 2-state data types, user-defined types, enumerated types, structures, and self-verifying decision statements. The importance of proper coding styles is emphasized throughout the course. Special attention is given to subtleties, such as how blocking and non-blocking assignments affect simulation and synthesis. Several labs reinforce the principles presented. Forty percent of the class time is devoted to hands-on experience, including a 5-hour final project modeling a complete DSP processor. A handy “*Verilog HDL Quick Reference Guide*” (\$15 value) is included with the course materials.

Intended Audience and Objectives

This workshop is for digital engineers who will be designing with Verilog and SystemVerilog. This workshop will enable students to immediately be productive with Verilog and SystemVerilog languages, and with simulation and synthesis software tools. Both new Verilog users, as well as those who are familiar with Verilog and desire a more in-depth knowledge of the language, will benefit from this course.

Software Tools Used

Students will use both Verilog simulation and synthesis tools during class labs. The tools used can be the Cadence *NC-Verilog*[™], Synopsys *VCST*[™], or Mentor Graphics *ModelSim*[™] simulators, and the Synopsys *DCT*[™], Synplicity *Synplify*[™], or Mentor *Precision*[™] synthesis compilers.

Prerequisites (essential)

Knowledge of digital design engineering is required. Without this background, students cannot fully benefit from this course. Labs include writing models of digital circuits such as shift registers, arithmetic logic units, FIFOs and a DSP.

Comments From Students

“An excellent comprehensive study of Verilog with perspectives on Verilog as a simulation, modeling and synthesis language; backed with valuable labs.”

“Excellent balance between lecture and lab. Very good structure. For sure the best course I've ever had!”

“Excellent, wonderful class. Definitely worth the 4 days. The instructor was extremely well prepared. He kept my interest the entire time.”

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—Verilog and SystemVerilog for Design and Synthesis

Day One

Introduction to Verilog and SystemVerilog

- Concepts of top-down design
- Overview of RTL models
- Overview of gate/switch models

Design Verification Using Simulation

- Writing verification testbenches in Verilog
- Running your preferred Verilog simulator
- Debugging designs with simulation
- Lab: running your simulator and debug tools

Verilog HDL Syntax and Semantics

- Identifier names
- Logic values and numbers
- Data types
- SystemVerilog extensions to Verilog data types
- Exercise: selecting the correct data types

Procedures, Programming Statements and Operators

- Procedural blocks
- SystemVerilog enhanced procedural blocks
- Tasks and functions
- Continuous assignments
- Programming statements and operators
- Exercise: programming statement subtleties (“gotchas”)
- Lab: modeling a simple ALU and testbench

Day Two

Synthesizing RTL Models

- General synthesis guidelines
- Running your preferred synthesis compiler
- Lab: synthesize a shift/storage register

RTL Models of Combinational Logic

- Always procedures and sensitivity lists
- Continuous assignments
- Synthesis full case and parallel case statements
- SystemVerilog unique and priority decision statements
- Lab: model, verify and synthesize an ALU

RTL Models of Sequential Logic

- Flip-flops and latches
- Synchronous and asynchronous inputs
- Lab: model, verify and synthesize a Johnson counter

Modeling State Machines

- Modeling Mealy and Moore state machines
- Modeling state encoding sequences
- SystemVerilog enumerated types
- Lab: model, verify and synthesize a UART

Day Three

Modeling Structural Netlists—After Synthesis

- Design hierarchy
- Module instantiation
- Generating arrays of instances
- Parameterized models and redefining parameters
- Verilog constructs used in ASIC/FPGA libraries
- Delay calculation and backannotation
- SDF files
- Lab: model and verify a hierarchical design, simulate with SDF delay backannotation

Modeling RAMs and ROMs

- Modeling memories
- Modeling bi-directional ports
- Testing bi-directional ports
- Timing constraints
- Lab: model and verify a dual-port RAM

Day Four

Design Verification with Verilog

- Design documentation
- Using Verilog as a verification language
- Structured tests
- Configurable test benches
- Writing output files
- Reading test vector files
- Adding built-in error checking
- Lab: verify a design using test vectors

Verilog Wizardry (Lab Intensive Project)

- Using all aspects of Verilog in a design project
- Simulating and verifying larger designs
- Lab: model and verify an embedded DSP processor

“The IEEE 1800 SystemVerilog standard enables modeling larger, more complex designs, and, at the same time, makes it easier to write Verilog models that synthesize correctly and optimally. Every design engineer learning Verilog should also learn the synthesizable portions of SystemVerilog, and know how to take full advantage of these powerful extensions to Verilog.”

Stuart Sutherland, President of Sutherland HDL, Inc.

SystemVerilog Synthesis for Verilog Design Engineers

Overview

SystemVerilog Synthesis for Verilog Design Engineers is a 2-day intensive workshop covering the hardware modeling portions of the latest IEEE 1800-2009 SystemVerilog standard. SystemVerilog is an exciting and powerful set of extensions to the Verilog Hardware Description Language, that enable efficiently modeling and verifying large, complex designs. The synthesizable subset of SystemVerilog presented in this workshop allows engineers to model more functionality with fewer lines of code, and at the same time ensure that models synthesize correctly and optimally. Simply put, SystemVerilog makes the design engineers life much easier and enables greater design productivity. This workshop is *not* a simple (and boring) language course. The focus of this workshop is on the proper usage of the extensive set of advanced modeling capabilities offered by SystemVerilog. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

A *comprehensive student guide*, a handy “*Verilog HDL Quick Reference Guide*” (\$15 value) are included with the course materials.

Course Objectives

This workshop will enable design engineers who use Verilog to immediately be productive modeling complex hardware designs using the SystemVerilog extensions to the Verilog language.

Intended Audience

This workshop is for experienced Verilog design engineers. The course presupposes a working knowledge of Verilog, and only covers the SystemVerilog enhancements to Verilog. Design engineers who do not have a strong background in Verilog should instead take Sutherland HDL’s “*Comprehensive Verilog and SystemVerilog for Design and Synthesis*” workshop, or combine this workshop with Sutherland HDL’s “*Accelerated Verilog Primer*”.

Software Tools Used

The Cadence *NC-Verilog*[™], Synopsys *VCS*[™] or Mentor Graphics *ModelSim*[™] simulator will be used for labs.

Prerequisites (essential)

Knowledge of the Verilog HDL is mandatory. Without this background, students cannot fully benefit from this course. This course presents the substantial extensions to the Verilog HDL, and assumes a working knowledge of Verilog.

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 Synthesis for Verilog Design Engineers

Day One

Introduction to SystemVerilog

- The purpose of SystemVerilog
- Backward compatibility with Verilog
- The SystemVerilog synthesis subset

Design Hierarchy

- External declarations
- Packages
- Nested modules
- Enhanced module instances
- Parameterized data types
- Lab: Using globals and packages

SystemVerilog Data Types and Logic Values

- New data types
- Casting
- User-defined types
- 2-state modeling techniques and “gotchas”
- Lab: Modeling with 2-state data types

Enumerated Types

- Enumerated types
- Specifying label values
- Assignments with enumerated types
- Static and dynamic casting
- Enumerated type methods
- Lab: Modeling a state machine with enumerated types

Data Aggregates: Arrays, Structures and Unions

- Packed arrays
- Unpacked arrays
- Assigning to arrays / copying arrays
- Structures
- Unions
- Tagged unions
- Bit-stream casting
- Lab: Using structures, unions and arrays

Day Two

SystemVerilog Procedural Blocks

- Combinational logic procedural blocks
- Latched logic procedural blocks
- Sequential logic procedural blocks
- Why specialized procedural blocks are important
- Task and function enhancements
- Passing arguments by reference
- Lab: Using specialized procedures

Programming Statements and Operators

- New operators
- Enhanced looping constructs
- Bottom testing loops
- New jump statements
- Unique and priority decision statements
- Eliminating simulation versus synthesis mismatches
- Lab: Modeling a high-level ALU

SystemVerilog Interfaces

- Interface definitions
- Defining module ports and directions
- Tasks and functions in interfaces
- Procedural code in interfaces
- Synthesizing interfaces
- Parameterized interfaces
- Verification paradigms with interfaces
- Lab: Using interfaces

Assertions for Design Engineers

- Assertion concepts
- Immediate assertions
- Concurrent assertions and sequences
- Basic sequence definitions
- Disabling assertions during reset
- Controlling assertion messages
- Lab: Using assertions in interfaces

Accelerated Verilog Primer

Overview

Accelerated Verilog Primer is a 2-day workshop that provides a practical overview of the language is presented, with a focus on how Verilog is used for modeling hardware and for design verification using digital logic simulation. This workshop is a abridged version of our 4-day *Comprehensive Verilog and SystemVerilog for Design and Synthesis* workshop. The same general topics are covered, but this accelerated version does not go into as much depth on synthesis rules and the labs are shorter. In this accelerated workshop, engineers learn how to read and understand Verilog models, and how to use Verilog to verify correct model functionality. Several labs reinforce the principles presented, with about 25% of the class time devoted to lab work. and how to write small hardware models or to read and understand larger, more complex models. The workshop materials include a complete student guide and a useful “*Verilog HDL Quick Reference Guide*” (\$15 value).

Course Objectives

At the conclusion of this workshop, students will be able to read and understand Verilog code, and be able to write and small digital hardware models and testbenches. Students will know proper modeling styles and recognize subtle modeling style errors that can lead to simulation race conditions and poor synthesis results.

Intended Audience

This workshop is intended for verification engineers, engineering managers, technical writers and others who wish to understand how Verilog is used in today’s design flows. This accelerated workshop is *not* a replacement for—or a prerequisite for—Sutherland HDL’s 4-day *Comprehensive Verilog and SystemVerilog for Design and Synthesis* workshop. Engineers who will be creating Verilog models of complex hardware should attend the 4-day workshop instead of this accelerated workshop.

Prerequisites (essential)

Knowledge of digital design engineering is required. Without this background, students cannot fully benefit from this course. Labs include writing models of basic digital circuits such as shift registers and arithmetic logic units.

Presented By

Stuart Sutherland or Don Mills. Mr. Sutherland is an expert consultant in Verilog design, with more than fourteen years experience using the Verilog language and Verilog software products. Mr. Mills is an independent consultant with extensive experience in the top-down design, synthesis and verification of ASICs and FPGAs.

Software Tools Used

Students will use Verilog simulators for labs, such as Cadence *NC-Verilog*[™], Synopsys *VCST*[™], or Mentor Graphics *ModelSim*[™].

Quotes From Students

“I would strongly recommend this course to anyone that wants to learn more about Verilog, simulation, and modern design practices.”

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.

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

Syllabus — Accelerated Verilog Primer Syllabus

Day One

Introduction to Verilog and SystemVerilog

- Concepts of top-down design
- Overview of RTL models
- Overview of gate/switch models

Design Verification Using Simulation

- Writing verification testbenches in Verilog
- Running your preferred Verilog simulator
- Debugging designs with simulation
- Lab: running your simulator and debug tools

Verilog HDL Syntax and Semantics

- Identifier names
- Logic values and numbers
- Data types
- SystemVerilog extensions to Verilog data types
- Exercise: selecting the correct data types

Procedures, Programming Statements and Operators

- Procedural blocks
- SystemVerilog enhanced procedural blocks
- Tasks and functions
- Continuous assignments
- Programming statements and operators
- Exercise: programming statement subtleties (“gotchas”)
- Lab: modeling a simple ALU and testbench

Day Two

Modeling Structural Netlists—After Synthesis

- Design hierarchy
- Module instantiation
- Generating arrays of instances
- Parameterized models and redefining parameters
- Verilog constructs used in ASIC/FPGA libraries
- Delay calculation and backannotation
- SDF files
- Lab: model and verify a hierarchical design

Modeling RAMs and ROMs

- Modeling memories
- Modeling bi-directional ports
- Testing bi-directional ports
- Timing constraints
- Lab: model and verify a dual-port RAM

Design Verification with Verilog

- Design documentation
- Using Verilog as a verification language
- Structured tests
- Configurable test benches
- Writing output files
- Reading test vector files
- Adding built-in error checking
- Lab: verify a design using test vectors

SystemVerilog Testbench for Verilog Verification Engineers

Overview

SystemVerilog Testbench for Verilog Verification Engineers is a 3-day fast-paced workshop covering the testbench constructs in the IEEE 1800 SystemVerilog standard. SystemVerilog unifies the strengths of Verilog, VHDL, C, VERA and PSL in one language, making it a true “Hardware Description and Verification Language” (HDVL). SystemVerilog enables the verification of very large, complex designs using a single language for all aspects of the design cycle, including simulation, synthesis and formal verification.

The focus of this workshop is on the verification aspects of SystemVerilog. Concepts presented include new special testbench modeling constructs, defining clocking domains, Object Oriented verification, constrained random verification, coverage, and assertions. Just a few of the SystemVerilog verification constructs that are presented include dynamic arrays, dynamic processes, object oriented inheritance and polymorphism, mailboxes, semaphores, specifying randomization constraints, specifying 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.

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 rich set of advanced verification extensions to the Verilog language to develop object-oriented testbenches that utilize constrained-random tests, functional coverage, mailboxes and scoreboarding.

Intended Audience

This workshop is for experienced Verilog engineers. The course presupposes a working knowledge of Verilog, and only covers the SystemVerilog testbench enhancements to Verilog. Verification engineers who are not familiar with Verilog should instead take Sutherland HDL’s “*SystemVerilog Testbench for non-Verilog Verification Engineers*” workshop.

Prerequisites (essential)

Knowledge of the Verilog HDL is mandatory. Without this background, students cannot fully benefit from this course. This course presents the substantial extensions to the Verilog HDL, and assumes a working knowledge of Verilog.

Software Tools Used

The Cadence *Incisive*[™], Synopsys *VCST*[™] 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.”

“The labs were particularly effective — long enough to learn the topic and to make you think.”

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 Verilog Verification Engineers

Day One

Introduction to SystemVerilog

- The purpose of SystemVerilog
- Backward compatibility with Verilog
- Software tools supporting SystemVerilog

Strategies for High-level Verification Programs

- Black-box versus white-box testing
- Assertion based design methods
- Constrained random testing
- Test synchronization
- Object-oriented testing

Verilog and SystemVerilog Data Types

- Verilog's 4-state data types and rules
- SystemVerilog's 2-state data types and rules
- Casting
- User-defined types
- Enumerated types
- 2-state modeling and verification “gotchas”
- Lab: Verification with 2-state and enumerated data types

SystemVerilog Testbench Programming Constructs

- Time units and precision
- Packages
- Design hierarchy
- Interfaces
- Procedural blocks
- Enhanced tasks and functions
- Lab: Working with complex hierarchical designs

Program Blocks and Clocking Domains

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

Day Two

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
- Fork—join dynamic processes
- Lab: Create a scoreboard using dynamic arrays

Process Synchronization

- Built-in mailbox classes
- Built-in semaphore classes
- Enhanced event data types
- Synchronized verification methodologies
- Lab: Using mailboxes for verification

Day Three

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

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

SystemVerilog Assertions for Design and Verification Engineers

Overview

SystemVerilog Assertions for Design and Verification Engineers is a 2-day advanced workshop covering the IEEE 1800 SystemVerilog Assertions (SVA). SVA enables engineers to verify extremely complex logic using a concise, portable methodology. SystemVerilog Assertions offer improvements at every stage of design and verification process. This workshop provides a thorough examination of SVA and assertion-based verification methodologies. Both immediate and concurrent assertions are presented, with discussion on the appropriate usage of each type of assertion. SVA sequence and property blocks are covered in great detail, with a focus on the semantics and proper usage of the many sequence and property operators. The presentation materials are laden with practical examples of writing assertions for various types of hardware logic. Topics presented in this comprehensive study on SVA include the use of local variables, property and sequence arguments, multiple thread termination and uniqueness, assertion-based system functions, and using assertions with multi-clock designs. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

Students receive a *comprehensive student guide*, a versatile “*Verilog HDL Quick Reference Guide*” (\$15 value), and the book “*SystemVerilog Assertions Handbook for Formal and Dynamic Verification*” (\$150 value).

Course Objectives

This workshop will enable design and verification engineers to immediately be productive with assertion-based verification methodologies and to write assertions and sequences that describe and verify complex design functionality.

Intended Audience

This workshop is targeted towards both digital design engineers and digital verification engineers. The workshop is for experienced Verilog engineers. The course presupposes a working knowledge of Verilog, and only covers the SystemVerilog assertion enhancements to Verilog. Engineers who do not have a strong background in Verilog should also take Sutherland HDL’s “*Accelerated Verilog Primer*” workshop.

Prerequisites (essential)

Knowledge of the Verilog HDL is mandatory. Without this background, students cannot fully benefit from this course. This course presents the substantial extensions to the Verilog HDL, and assumes a working knowledge of Verilog.

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. 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 Assertions for Design and Verification Engineers

Day One

Introduction to SystemVerilog Assertions (SVA)

- A quick look at SystemVerilog Assertions
- Software tools supporting SVA
- Lab: Running simulations with SVA

Assertion Based Verification (ABV) Methodologies

- The traditional design process
- Using SVA in the definition of designs
- Using SVA in the definition of verification
- Using SVA with reusable IP
- Using SVA to facilitate coverage metrics
- Case study: a synchronous FIFO

Overview of the SystemVerilog language

- Design hierarchy: modules, interfaces, packages
- Program blocks and clocking blocks
- Procedural blocks
- Enhanced tasks and functions
- SystemVerilog data types
- Enumerated types
- Lab

Overview of SVA Properties and Sequences

- Immediate and concurrent assertions
- The SVA property construct
- The SVA sequence construct
- When to use properties versus sequences
- Antecedent, consequent and threads
- Assertion, assumption and verification directives
- Lab

Understanding Properties

- Property declaration syntax
- Using formal arguments
- Local variables in properties
- Clocking events
- Disabling condition
- Property expressions
- Property operators
- Lab

Understanding Sequences

- Sequence operators and built-in functions
- Capturing temporal behavior
- Implication operators
- First match operator
- Repetition operators
- Sequence composition operators
- Sequence methods
- Lab

Day Two

Advanced Properties and Sequences

- Data types in properties and sequences
- Proper use of assertion overlapping
- Multiple thread termination
- Unbounded ranges in properties
- Emulating PSL-like constructs in SVA
- Lab

SVA System Functions and System Tasks

- Using the \$sampled system function
- Using the \$past, \$fell and \$stable system functions
- Vector analysis system functions
- Severity level system functions
- Assertion control system tasks
- Lab

Clocked and Multi-clocked Assertions

- Clock specification for properties and sequences
- Clock resolution
- Multiple clocked sequences
- Multiple clocked properties
- Lab

Binding SVA to Design Blocks

- The SVA bind construct
- Binding to all instances of a module or interface
- Binding to a single instance of a module or interface
- Verifying VHDL models using SVA
- Lab

Verification Directives & Verification-based Coverage

- The assert, assume and cover directives
- SVA coverage
- Coverage metrics
- Lab

Formal Verification Using SVA

- Formal verification methodology
- SVA in formal specifications
- Formal verification coverage metrics
- Formal verification versus dynamic simulation with SVA
- Case study: a traffic light controller

SVA Coding Guidelines

- Naming conventions
- Where to write properties and assertions
- Proper usage of the property negation operator
- Proper usage of local variables (write before read)
- Proper usage of unbounded ranges
- Using a default clock
- Using dynamic data types with assertions

Advanced SystemVerilog Open Verification Methodology (OVM)

Overview

Advanced SystemVerilog Open Verification Methodology is a 3-day comprehensive training course on how to create complex high-level verification environments using SystemVerilog and the OVM class libraries.

The *Advanced SystemVerilog Open Verification Methodology* course is offered through Sutherland HDL, but is developed by—and is presented by—Willamette HDL. Sutherland HDL has a partnership agreement to arrange this workshop in behalf of Willamette HDL as an adjunct to Sutherland HDL's SystemVerilog verification workshops.

Course Objectives

At the conclusion of this workshop, engineers will understand how to use SystemVerilog to apply the complex verification methodology recommended in the book “*OVM Open Verification Methodology Cookbook*” by Mark Glasser.

Intended Audience

This workshop is for verification engineers who are already familiar with the SystemVerilog testbench constructs. The course presupposes a working knowledge of Verilog and the SystemVerilog testbench enhancements to Verilog. Engineers who are not familiar with SystemVerilog should first take Sutherland HDL's “*SystemVerilog Testbench for Verilog Verification Engineers*” or “*SystemVerilog Testbench for non-Verilog Verification Engineers*” workshop.

Prerequisites (essential)

Completion of Sutherland HDL's SystemVerilog Testbench workshop, or equivalent knowledge, is mandatory before taking this advanced workshop! Without this prerequisite knowledge, students cannot fully benefit from this workshop.

Course contents

A full description of this workshop can be found at: www.whdl.com.

Workshop Locations

This workshop is only available as on-site training held at your facilities.

Practical Application of the Verification Methodology Manual Using SystemVerilog

Overview

Practical Application of the Verification Methodology Manual Using SystemVerilog is a 1-day comprehensive training course on the practical definition of a SystemVerilog transaction-based testbench compliant to the “*Verification Methodology Manual for SystemVerilog*” (VMM) by Janick Bergeron, et. al. A FIFO design is used to illustrate using the VMM methodology for the creation of a comprehensive constrained-random verification environment. This includes the generation of transactions and the consumption of transactions via transactors, the definition of the verification environment, the monitor, the scoreboard, the factory and callback methods, directed tests, custom generator. Labs reinforce these VMM concepts. Students receive a comprehensive training guide that will serve as an invaluable aid in applying the concepts in the book “*Verification Methodology Manual for SystemVerilog*”.

Practical Application of the Verification Methodology Manual Using SystemVerilog is a VHDL/Cohen training course developed and presented by Ben Cohen. This course is offered through Sutherland HDL, but is not a Sutherland HDL training course. Ben Cohen is a well known author of VHDL, Verilog and SystemVerilog books, and an experienced design and verification engineer.

Course Objectives

At the conclusion of this workshop, engineers will understand how to use SystemVerilog to apply the complex verification methodology recommended in the book “*Verification Methodology Manual for SystemVerilog*” by Bergeron, Cerny, Hunter, and Nightingale.

Intended Audience

This workshop is for verification engineers who are already familiar with the SystemVerilog testbench constructs. The course presupposes a working knowledge of Verilog and the SystemVerilog testbench enhancements to Verilog. Engineers who are not familiar with SystemVerilog should first take Sutherland HDL's “*SystemVerilog Testbench for Verification Engineers*” workshop.

Prerequisites (essential)

Knowledge of the SystemVerilog testbench constructs is mandatory! Without this prerequisite knowledge, students cannot fully benefit from this workshop. Familiarity with SystemVerilog Assertions is beneficial.

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. 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 — Practical Application of the Verification Methodology Manual Using SystemVerilog

Day One

Section 1: VMM Transactions and Channels

- Why a verification framework
- Testbench architecture
- Elements of a testbench in framework
- Transactions
- Channels
- Environment
- Lab 1

Section 2: Transaction Generator and Transactor

- Elements of a testbench in framework
- Transaction generator
- Transactor
- Lab 2

Section 3: Building the Environment

- Verification environment
- Programs
- Testbenches
- Simulation
- Lab 3

Section 4: Using the Factory Method

- Factory definition
- Factory applications
- Factory method
- Allocation
- Lab 4

Section 5: Callback

- Callback definition
- Callback applications
- Callback method
- Lab 5

Section 6: Advanced Framework

- Custom generator
- Directed tests
- Monitors and Scoreboards
- Ending the simulation
- Lab 6

Advanced Verilog-2005 PLI 2.0 (VPI) and SystemVerilog DPI

Overview

The Verilog Programming Language Interface (PLI) is an important part of Verilog design. The PLI provides designers a means to extend the Verilog language, and to customize Verilog software tools to perform specific verification tasks. A basic premise of the creators of Verilog was to keep the Verilog language directly related to hardware design, and to provide a procedural interface to tie verification and abstract modeling tasks into a Verilog simulation.

Advanced Verilog-2005 PLI 2.0 (VPI) and SystemVerilog DPI is a comprehensive workshop on the IEEE 1364-2005 Verilog Programming Language Interface. The workshop teaches how to write PLI applications to extend the capabilities of Verilog software tools by reading test vector files, verifying test coverage, calculating power usage, annotating accurate delays, and interfacing to Bus Functional C models. Students also learn how to use the IEEE 1800-2005 Direct Programming Interface (DPI) that extends the Verilog PLI. About fifty percent of the class time is devoted to hands on experience writing several useful PLI applications.

Students receive a *comprehensive student guide*, and a handy “*Verilog PLI Quick Reference Guide*” (\$30 value).

Course Objectives

Students will learn how to write PLI DPI applications that are portable across all major simulators. Students will also understand the appropriate times to use the Verilog PLI and when to use the SystemVerilog DPI.

Intended Audience

This advanced workshop is for software engineers, verification engineers and hardware engineers who will be writing or maintaining PLI applications.

Prerequisites

Knowledge of the C programming language is mandatory — All labs involve writing small C programs. A basic knowledge of the Verilog HDL language is also mandatory.

Presented By

Stuart Sutherland. Mr. Sutherland is an expert engineering consultant, with extensive experience in Verilog design, simulation and verification. He is the author of “*The Verilog PLI Handbook*”, a comprehensive reference book on the PLI, and has served as the chairman and technical editor for the IEEE 1364 Verilog PLI standard.

Software Tools Used

During class labs, students will write PLI and DPI applications in C, and link these applications to Verilog simulators. The course includes information on using the Cadence *NC-Verilog*[™], Synopsys *VCS*[™], and Mentor Graphics *ModelSim*[™] simulators.

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 — Advanced Verilog-2005 PLI 2.0 (VPI) and SystemVerilog DPI

Day 1

Introduction to the PLI standard

- The purpose of the PLI
- The IEEE PLI 1.0 and 2.0 standards

VPI task function callbacks

- User-defined system tasks and functions
- Calltf and compiletf applications
- The VPI interface mechanism and callback registry
- Lab: compile and link a PLI application

Using VPI routines

- VPI object handles
- VPI object relationships
- VPI object properties
- Lab: write C programs to access task arguments

VPI routines to access object properties

- Reading integer and boolean properties
- Reading string properties
- Lab write a C program to display design data

Traversing a simulation data structure

- Object relationships
- Understanding the IEEE 1364 object diagrams
- Traversing single levels of design hierarchy
- Traversing multiple levels of design hierarchy
- Lab: write a C program to extract data on ASIC cells

Day 2

VPI routines to read values

- Reading logic values
- Reading delay values
- Lab: display design debugging information

VPI routines to modify values

- Modifying logic values
- Modifying delay values
- Lab: write system function to calculate square roots

VPI routines to schedule PLI callbacks

- Simulation event callbacks
- Simulation time callbacks
- User-defined work areas
- Lab: write a C program to read test vector files

Interfacing to C models using VPI routines

- Callbacks based on value changes
- Passing inputs and outputs to C models
- Lab: write an interface to a C model

The SystemVerilog Direct Programming Interface

- How the DPI works
- Importing C functions into Verilog models
- Exporting Verilog tasks and functions to C
- When to use the PLI and when to use the DPI
- Lab: Directly importing a C function into Verilog

Comprehensive VHDL for Synthesis and Verification

Overview

Comprehensive VHDL for Synthesis and Verification is a 4-day workshop covering the aspects of the IEEE 1076 VHDL standard and IEEE 1164 Standard Logic standard that is useful for synthesis modeling and test benches. The design style focus is on using VHDL for top-down design with synthesis and simulation. Special attention is given to working with subtleties of the language, such as how to work with different types, operator overloading and usage of pre built packages. The importance of coding style, design verification and documentation are also emphasized. A student guide, reference sheets and the book *“The Designer's Guide to VHDL”* by Peter J. Ashenden are included.

The primary focus of the workshop is two fold. First is on writing synthesizable VHDL models for simulation. Second is on writing the simulation test bench for each lab. Several simulation labs reinforce the principles presented. Sixty five percent of the class time is devoted to hands-on experience, as engineers model and simulate a number of small hardware circuits. At the conclusion of this workshop, engineers will have the modeling and verification skills to immediately be productive in writing, simulating and synthesizing VHDL models of hardware designs.

Presented By

Don Mills. Mr. Mills is an independent consultant with extensive experience in the top-down design and synthesis of ASICs, using CMOS and ECL technologies.

Software Tools Used

Students will be using VHDL simulation tools during class labs. On-site training will cover what ever simulator is provided in the training lab. Workshops using Sutherland HDL's portable lab equipment use the Mentor Graphics *ModelSim* simulator.

Intended Audience

VHDL for Synthesis and Verification workshop is for digital engineers who will be designing ASICs, FPGAs or systems with VHDL. The workshop enables students to immediately be productive with the VHDL language and VHDL software tools. Both new VHDL users, as well as those who are familiar with VHDL and desire a more in-depth knowledge of the intricacies of VHDL, will benefit from this course.

Prerequisites (essential)

Knowledge of digital design engineering is required. Without this background, students cannot fully benefit from this course. Labs include writing VHDL models of digital circuits such as shift registers, arithmetic logic units, and FSMs.

Comments From Students

“The best part of the course was state machines, type conversions, composite data types, test benches, and large design techniques.”

“Lecture half day and lab half day is perfect.”

“I liked the one on one personal help with labs and answering questions.”

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 **1-866-HDL-XPRTS** (1-866-435-9778). From outside the US, please call +1-503-692-0898.

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

Syllabus — Comprehensive VHDL for Synthesis and Verification

Day One

Introduction to VHDL

- Concepts of top-down design
- Basic building blocks of VHDL
- Entity declarations
- Architecture bodies
- Formatting notes
- Comments
- Identifiers
- Numbers
- Characters and strings
- Bit strings

Scalar Data Types and Operations

- VHDL objects
- Declarations
- Constants and variables
- Types and type declarations
- Operators
- Operator overloading
- Std_Logic 1164 type
- Sub types
- Type conversion

Processes, Sequential Statements, and Test Benches

- Process basics
- Sequential statements - IF, CASE, LOOP
- Test bench introduction
- Test bench configuration
- Labs: Combinational logic designs

Day Two

Composite Data Types and Operations

- Composite types
- Arrays
- Array aggregates
- Array attributes
- Bit vectors and Std_LOGIC arrays
- Array operations and referencing

Putting all the Pieces Together

- Entities
- Architectures
- Processes
- Signal declarations, assignments, and attributes
- Introduction to FF models
- Synchronous and asynchronous FFs
- More on test benches
- Labs: Combinational and sequential logic designs

Day Three

Timing Delays in Processes

- Wait statements
- Delta delays
- Transport and inertial delays

Concurrent Signal Assignment Statements

- Direct assignments
- Conditional assignment
- Selected assignment

Subprograms

- Procedures
- Procedure declaration
- Procedure body
- Procedure parameters
- Functions
- Function declaration and body
- Pure and impure functions
- Overloaded functions
- Labs: More sequential logic designs

State Machines

- Coding styles for state machines
- Labs: State machine designs

Day Four

VHDL Miscellaneous Topics

- Packages and package bodies
- Library and use clauses
- Pre-built packages
- Resolved data types and signals
- Components
- Configurations

Generics and Generate Statements

- Generic constants declaration
- Generic constant redefinition
- For generate statement
- IF generate statement
- Labs: Generics and generates

ASIC Design Methods & Introductions to Synthesis

- Latch coding styles
- How to avoid latches
- Clock zones and synchronous design
- Synthesis constraints
- Synthesis reports
- Synthesis guidelines