SUTHERLAND HDL

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

Sutherland HDL offers **on-site** and **online** expert-level training workshops on the SystemVerilog hardware design and verification language, backed by many years of design and verification experience. Typical feedback includes:

> *"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."*
>
> *"The instructor's expert knowledge provides insight not possible from other courses."*

On-site Workshops

Sutherland HDL on-site training workshops are held at your facilities and presented by expert instructors.

- Training can be scheduled at a time and location that is most effective for an engineering team.
- Course topics can be customized to meet the needs of the engineering team.
- Engineers are encouraged to discuss how to best apply SystemVerilog in their specific projects during both class time and lab time. Engineers work on the labs in teams of two or three to facilitate lots of discussion.
- All that is required is a meeting room, with a laptop or workstation for every other attendee, and access to a SystemVerilog simulator. Optionally, Sutherland HDL can provide a portable lab environment, with a server and isolated network with licensed simulation software.

eTutored™ live Online Workshops

Sutherland HDL *eTutored™ live* online workshops are instructor-led workshops that provide all the same learning benefits as classroom based training, but with greater flexibility to mix expert training and work responsibilities.

- Engineers log on to a 4-hour online interactive classroom-like discussion each day.
- An expert instructor with real experience using SystemVerilog presents vital concepts on the proper usage of SystemVerilog. *eTutored™ live* courses use the same professional materials, lecture and labs as on-site workshops.
- Following the lecture block, each student works independently to complete 2 to 3 hours of lab projects at their own time and pace. The instructor is available by phone, e-mail or Skype to assist with the labs.
- Lab projects are submitted to the Sutherland HDL expert instructor for review and recommendations on best coding practices.

Private and Customized Workshops

Sutherland HDL on-site and online training can be presented as private workshops, where attendance is restricted to engineers from just your company. This allows engineers to freely discuss how the concepts presented during the training will be applied in your design or verification projects. The Sutherland HDL instructor can also provide recommendations specific to your projects. The instructor can sign an NDA agreement if required. Sutherland HDL can also customize a private workshops to meet your engineering team's needs. An extra cost involved to customize the content of a training workshop if additional materials need to be developed for the training.

Private workshops can be scheduled at a date and class hours that meet the needs of your engineering team's needs. Online workshops can also be scheduled for non-contiguous days. A low minimum of 5 attendees is required to schedule a private on-site training workshop, and 4 attendees to schedule a private *eTutored™ live* online workshop.

Public Online Workshops

Sutherland HDL *eTutored™ live* online training is also presented as public, open-enrollment workshops. Engineers from several companies can attend a public workshop. These workshops are scheduled on a regular basis during the year, and can be ideal when your company has a new-hire or less than 4 engineers who need training.

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 available.

Visit www.sutherland-hdl.com or call +1-503-692-0898 for public *eTutored™ live* online workshop schedules, or to discuss scheduling an on-site workshop or a private *eTutored™ live* online workshop for 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 UVM consulting 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 30 years of experience in hardware design and verification, and over 25 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 Masters of Education with an emphasis in online learning from Northcentral University (Arizona) and 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 "*Verilog and SystemVerilog Gotchas*", 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 1800 SystemVerilog standardization working group, and is the technical editor of the IEEE P1800 Language Reference Manual (LRM).

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 25 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 Verilog, SystemVerilog and VHDL, and 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.

Verilog and SystemVerilog Language Foundations

Overview

Verilog and SystemVerilog Language Foundations is a fast-paced workshop designed to help engineers read, understand, and maintain digital hardware models and conventional verification testbenches written in Verilog and SystemVerilog. Critical language foundations are presented, including the proper use of data types, 2-state versus 4-state modeling, enumerated types, structures, and SystemVerilog's versatile set of programming statements. Hands-on labs reinforce the concepts discussed during the class lectures. This workshop provides a ***critical language foundation*** for more advanced training on SystemVerilog, including writing object-oriented verification testbenches, and coding SystemVerilog Assertions. ***Not just for beginners!*** Engineers who have years of experience with Verilog often comment regarding how much they learn and benefit from the details provided in this workshop.

Course Length:
2-days on-site, 2-days *eTutored™ live*.

Intended Audience and Objectives

This workshop is ideal — and important — for three primary groups:

1. Engineers and managers who need to read and understand Verilog/SystemVerilog hardware models or verification testbenches, but who will not be writing large amounts of code.
2. Engineers who will be attending advanced training on verification. The knowledge gained in this workshop provides the prerequisite foundation for Sutherland HDL's *"SystemVerilog Object Oriented Verification"* and "SystemVerilog Assertions for Design Engineers and Verification Engineers" workshops.
3. Verilog users who need to learn the important enhancements SystemVerilog added to traditional Verilog.

(The information presented in this workshop can be combined with other workshops, and is integrated into Sutherland HDL's *"Verilog/SystemVerilog for Design and Synthesis"* and *"SystemVerilog Object Oriented Verification"* workshops.)

Prerequisite Knowledge (essential)

Attendees should be familiar with general digital hardware engineering concepts such as the difference between combinational logic and sequential logic, binary and hexadecimal number systems, and binary operations.

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

Engineers will use SystemVerilog simulation tools during class labs. The tools used can be the Aldec *Riviera-Pro™*, Cadence *NC-Verilog™*, Synopsys *VCS™*, or Mentor Graphics *ModelSim™* or *Questa™* simulators.

Comments From Students

"Excellent, wonderful class. Definitely worth the time. 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 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*.

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 — Verilog and SystemVerilog Language Foundations

Introduction to Verilog and SystemVerilog
- Concepts of top-down design
- Overview of RTL models
- Overview of gate/switch models
- The RTL design flow with simulation and synthesis

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 and 2-state vs. 4-state guidelines
- Enumerated types
- User-defined types
- Casting
- Lab: Experiment with 2-state and 4-state data types

Procedures and Assignment Statements
- Procedural blocks
- SystemVerilog enhanced procedural blocks
- Blocking and nonblocking assignments
- Continuous assignments
- Tasks and functions
- Lab: Experiment with blocking and nonblocking assignments

Programming Statements and Operators
- Programming statements and language rules
- Operators and language rules
- Avoiding subtle programming “gotchas”
- Guidelines for best coding practices
- Lab: Model a simple ALU

SystemVerilog Compound Types and Packages
- Arrays and array assignments
- Structures and unions
- Packages
- 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
- Interfaces as an RTL modeling construct
- Interfaces as a verification construct
- Lab: Model and verify a master/slave interface bus

“SystemVerilog is a significant enhancement to Verilog and includes extensions into abstract design, testbench, formal, and C-based APIs... These extensions provide significant new capabilities to the designer, verification engineer and architect, allowing better teamwork and coordination between project members.”

Phil Moorby, creator of the original Verilog HDL

Verilog/SystemVerilog for Design and Synthesis

Overview

Verilog/SystemVerilog for Design and Synthesis is a comprehensive workshop covering the complete Verilog Hardware Description Language and the synthesizable portions of SystemVerilog, including user-defined types, enumerated types, structures, and self-verifying decision statements. The workshop all of the topics presented in the *"Verilog and SystemVerilog Language Foundations"* workshop and adds detailed discussion and labs on best coding practices for writing synthesizable RTL models that work correctly in both simulation and synthesis. Special attention is given to language subtleties and avoiding "gotchas" that can lead to mismatches between simulation and synthesis. About forty percent of the course is devoted to hands-on experience in labs that reinforce the principles presented, including an extensive final project modeling a small Digital Signal Processor (DSP).

Course Length: 4-days on-site, 5-days *eTutored™ live*.

Intended Audience and Objectives

This workshop is for digital engineers who will be designing with Verilog and SystemVerilog. Students will be immediately productive in using Verilog and SystemVerilog for modeling, simulation and synthesis. Both new Verilog/SystemVerilog users, as well as those who are familiar with Verilog/SystemVerilog and desire a more in-depth knowledge of the language, will benefit from this workshop.

Prerequisite Knowledge (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.

Included Materials

- Full-color training binder with copies of all lecture slides, lab instructions, and supplemental information.
- Lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

Engineers will use both Verilog simulation and synthesis tools during class labs. The simulators that can be used are the Aldec *Riviera-Pro*™, Cadence *NC-Verilog*™, Synopsys *VCS*™, or Mentor Graphics *ModelSim*™ or *Questa*™ simulators. Synthesis tools include the Cadence *Encounter RTL Compiler*™, Synopsys *DC*™ or *SynplifyPro*™, or Mentor *Precision*™ synthesis compiler.

Comments From Students

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

"I just wanted to say how great your Verilog and SystemVerilog training was. Thank you for a brilliant Verilog/ SystemVerilog education. I don't think designers should be allowed to write code until they have this course!"

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*.

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 — Verilog/SystemVerilog for Design and Synthesis

Introduction to Verilog and SystemVerilog
- Concepts of top-down design
- Overview of RTL models
- Overview of gate/switch models
- The RTL design flow with simulation and synthesis

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 and 2-state vs. 4-state guidelines
- Enumerated types
- User-defined types
- Casting
- Lab: Experiment with 2-state and 4-state data types

Procedures and Assignment Statements
- Procedural blocks
- SystemVerilog enhanced procedural blocks
- Blocking and nonblocking assignments
- Continuous assignments
- Tasks and functions
- Lab: Experiment with blocking and nonblocking assignments

Programming Statements and Operators
- Programming statements and language rules
- Operators and language rules
- Avoiding subtle programming “gotchas”
- Guidelines for best coding practices
- Lab: Model and verify a simple ALU

SystemVerilog Compound Types and Packages
- Arrays and array assignments
- Structures and unions
- Packages
- 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
- Interfaces as an RTL modeling construct
- Lab: Model and verify a master/slave interface bus

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
- Lab: Model, verify and synthesize a UART

Modeling Structural Netlists—After Synthesis
- 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 an ASIC netlist and use SDF backannotation

Modeling RAMs and ROMs
- Modeling memories
- Modeling bi-directional ports
- Timing constraints
- Lab: Model and verify a dual-port RAM

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 SystemVerilog standard enables modeling larger, more complex designs, and, at the same time, simplifies writing models that synthesize correctly. 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 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, and the use of SystemVerilog interfaces as a verification construct. An overview of SystemVerilog assertions, and an overview of the SystemVerilog Universal Verification Methodology (UVM) are also included. Engineers learn how to utilize object inheritance and polymorphism, mailboxes, semaphores, specifying randomization constraints, specifying functional coverage, and dynamic arrays. The workshop integrates in the topics from Sutherland HDL's *"Verilog and SystemVerilog Foundations"* workshop to provide engineers with a strong basis in SystemVerilog's programming and operator constructs, and how these constructs are properly used in both testbenches and hardware models that are to be verified. Labs reinforce the concepts learned, with approximately forty percent of the class time devoted to hands-on experience.

Course Length: 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 Foundations"* workshop.

Included Materials

- Full-color training binder with copies of all lecture slides, lab instructions, and supplemental information.
- 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*™ simulators can be used for labs.

Quotes From Students

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

"Excellent class. Definitely worth the time. The instructor kept my interest the entire time."

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 — SystemVerilog Object-Oriented Verification

Introduction to Verilog and SystemVerilog
- Concepts of top-down design
- Overview of RTL models
- Overview of gate/switch models
- The RTL design flow with simulation and synthesis

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 and 2-state vs. 4-state guidelines
- Enumerated types
- User-defined types
- Casting
- Lab: Experiment with 2-state and 4-state data types

Procedures and Assignment Statements
- Procedural blocks
- SystemVerilog enhanced procedural blocks
- Blocking and nonblocking assignments
- Continuous assignments
- Tasks and functions
- Lab: Use blocking and nonblocking assignments

Programming Statements and Operators
- Programming statements and language rules
- Operators and language rules
- Avoiding subtle programming “gotchas”
- Guidelines for best coding practices
- Lab: Model a simple ALU

SystemVerilog Compound Types and Packages
- Arrays and array assignments
- Structures and unions
- Packages
- Overview of SystemVerilog interfaces
- Lab: Model and verify an Instruction Stack

Verilog Verification Constructs
- Verification-specific constructs
- Configurable test benches
- Reading and Writing data files
- SystemVerilog interfaces as a verification construct
- Lab: Use an interface between the testbench and DUT

Verification and Clocking Domains
- Synchronizing testbench and DUT activity
- SystemVerilog event scheduling regions
- Clocking blocks
- Lab: Developing a test program

Object-Oriented Programming, Part One
- SystemVerilog’s class definitions
- Class methods
- Object inheritance
- Lab: Create a simple OO testbench

Object-Oriented Programming, Part Two
- Extending class definitions
- Virtual methods
- Virtual classes
- Public and private classes
- Lab: Create a more advanced OO testbench

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

Process Synchronization
- Fork—join dynamic processes
- Mailboxes
- Semaphores
- Lab: Use mailboxes for verification

Constrained Random Value Generation
- SystemVerilog randomization
- Defining constrained random values
- Lab: Generate 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

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

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:

- 3-days on-site, 4-days *eTutored™ live* as a stand-alone workshop for engineers who are already familiar with SystemVerilog object-oriented programming:
- 5-days on-site (not available as an online *eTutored™ live* workshop) when combined with an accelerated version of Sutherland HDL's *"SystemVerilog Object Oriented Verification"* 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)

A working knowledge of any programming language is mandatory. A knowledge of digital design engineering is expected. The training materials, lectures and labs include discussion of programming and of hardware specific concepts such as combinational logic, sequential logic, setup/hold times and other hardware principles. Without an understanding of this terminology, attendees cannot fully benefit from this course.

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

SystemVerilog Assertions for Design and Verification Engineers

Overview

SystemVerilog Assertions for Design and Verification Engineers is an 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 and training guide are filled 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 and clock domain crossing. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

Course Length: 2-days on-site, 3-days *eTutored™ live*.

Intended Audience and Objectives

This workshop is targeted towards both digital design engineers and digital verification engineers. The workshop is for experienced engineers and 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.

Prerequisite Knowledge (essential)

A working knowledge of SystemVerilog is essential in order to fully benefit from this workshop. In order to fully understand and utilize the concepts presented in this course, students should have first completed the Sutherland HDL ***SystemVerilog Object-Oriented Verification*** course or equivalent.

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*™ simulators 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.

SUTHERLAND HDL

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

Syllabus — SystemVerilog Assertions for Design and Verification Engineers

Introduction to SystemVerilog Assertions (SVA)
- A first look at SystemVerilog Assertions
- The traditional design process
- Using SVA in the definition of designs
- Using SVA in the definition of verification
- Using SVA to facilitate coverage metrics
- Naming conventions
- Lab: Running simulations with SVA

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

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

Advanced Properties and Sequences
- Data types in properties and sequences
- Proper use of assertion overlapping
- Chaining implication operators
- Multiple thread termination
- Unbounded ranges in properties
- 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
- Using a default clock
- Multiple clocked sequences
- Multiple clocked properties
- Lab

Verification Directives & Verification-based Coverage
- The assert, assume and cover directives
- SVA coverage
- Coverage metrics
- 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

Assertion Verification Plans
- What goes into an assertion verification plan
- Planning the who, what and where
- Analyzing the design specification
- Final project: Define assertions for a small DSP design