

Sutherland HDL offers expert-level training workshops on the SystemVerilog hardware design and verification languages, bringing years of real design and verification experience to the class. 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.” — Oracle (formerly Sun Microsystems)

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

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.
- During and after class, engineers can discuss how to best apply SystemVerilog in their specific projects.
- All that is required is a conference room. Sutherland HDL can provide a portable lab environment, with computers, simulation and synthesis 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 an online Webinar meeting for 4 hours each day (typically 8 AM to noon Pacific time).
- 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 1 to 2 hours of lab projects anytime during the afternoon. The instructor is available by phone, e-mail or Skype to assist with the labs.
- Lab projects are submitted at the end of each day to the Sutherland HDL expert instructor for review and recommendations on best coding practices.
- *eTutored™ live* online workshops can be scheduled for dates that are most effective for your engineering team (4 student minimum). Sutherland HDL also holds periodic public *eTutored™ live* workshops.

eTutored™ self-paced Online Workshops

Instructor-assisted comprehensive training anytime and anywhere — the utmost in schedule flexibility. Sutherland HDL *eTutored™ self-paced* online courses provide one-on-one tutoring from experienced SystemVerilog experts.

- Engineers can start an *eTutored™ self-paced* workshop on any date, and have 60 days to complete the course.
- Engineers log on to Sutherland HDL’s online campus to complete training modules that are custom designed for effective online learning (Mr. Sutherland holds a Masters degree in eLearning).
- A Sutherland HDL instructor is available by phone, e-mail or Skype to answer questions or assist with labs.
- Lab work is submitted to the instructor for review and recommendations on best coding practices, ensuring that concepts are correctly understood and that there are no misconceptions that could be disastrous in a real project.
- A prior knowledge evaluation at the beginning of most course modules allows each engineer to jump over what he or she already knows and focus on what needs to be learned. That can’t be done in a lecture-based group class.

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* 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 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 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 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 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.

SystemVerilog for Design and Synthesis

Overview

SystemVerilog for Design and Synthesis is a comprehensive workshop covering the complete Verilog Hardware Description Language and the synthesizable portions of the SystemVerilog extensions to Verilog. The synthesizable subset of SystemVerilog includes 2-state data types, user-defined types, enumerated types, structures, and self-verifying decision statements. Emphasis is placed on proper SystemVerilog coding styles for top-down design with synthesis and simulation. Special attention is given to subtleties, such as how blocking and non-blocking assignments affect simulation and synthesis. About forty percent of the course is devoted to hands-on experience in labs that reinforce the principles presented, including a 5-hour final project modeling a small DSP processor.

Course length: 4-days on-site. 5-days *eTutored™ live*. 5 to 60 days *eTutored™ self-paced*.

Intended Audience and Objectives

This workshop is for digital engineers who will be designing with Verilog and SystemVerilog. This workshop will enable students to immediately be productive 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 course.

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.

Included Materials

- Comprehensive binder with all PowerPoint slides (printed in color), lab instructions, and supplemental white papers. (*eTutored™ self-paced* courses use an eBook and other online materials instead of a training binder.)
- A handy “*Verilog HDL Quick Reference Guide*” (\$15 value).
- All lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

Engineers will use both Verilog simulation and synthesis tools during class labs. The tools used can be the Cadence *NC-Verilog™*, Synopsys *VCS™*, or Mentor Graphics *ModelSim™* simulators, and the Synopsys *DC™* or *SynplifyPro™*, or Mentor *Precision™* synthesis compilers.

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 or as an *eTutored™ live* online class. We also offer several public *eTutored™ live* workshops throughout the year. For more information, please visit www.sutherland-hdl.com, or call us at +1-503-692-0898.

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

Syllabus — SystemVerilog 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 Verification Foundations

Overview

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

Course length: 4-days on-site. 5-days *eTutored™ live*. 5 to 60 days *eTutored™ self-paced*.

Intended Audience and Objectives

This workshop is for design and/or verification engineers who will be creating test environments for the verification of digital designs. At the conclusion of this workshop, engineers will understand how to take full advantage of the verification capabilities in the SystemVerilog language in order to develop object-oriented testbenches that utilize constrained-random verification methodologies, functional coverage, mailboxes and scoreboarding.

Prerequisites (essential)

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

Included Materials

- Comprehensive binder with all PowerPoint slides (printed in color), lab instructions, and supplemental white papers. (*eTutored™ self-paced* courses use an eBook and other online materials instead of a training binder.)
- A handy “*Verilog HDL Quick Reference Guide*” (\$15 value).
- All lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

The Cadence *Incisive™*, Synopsys *VCS™* or the Mentor Graphics *Questa™* simulator will be used for labs.

Quotes From Students

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

“Best workshop I’ve attended. Very informative and insightful into what SystemVerilog offers.”

Workshop Locations

This workshop can be presented on-site at your facilities or as an *eTutored™ live* online class. We also offer several public *eTutored™ live* workshops throughout the year. For more information, please visit www.sutherland-hdl.com, or call us at +1-503-692-0898.

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

Syllabus — SystemVerilog Verification Foundations

Day One

Introduction to Verilog and SystemVerilog

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

Verilog and SystemVerilog Syntax and Semantics

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

Procedures, Programming Statements and Operators

- Procedural blocks
- Tasks and functions
- Procedural assignments (blocking and non-blocking)
- Continuous assignments
- Programming statements and operators
- Lab: model and verify an 8-bit ALU

Verilog Structural Models

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

Day Two

Modeling RAMs and ROMs

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

Verilog Verification Constructs

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

SystemVerilog User-defined Types and Packages

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

Program Blocks and Clocking Domains

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

Day Three

Object-Oriented Programming, Part One

- SystemVerilog's class data type
- Defining class objects
- Class methods
- Class inheritance
- Lab: Creating a simple OO testbench

Object-Oriented Programming, Part Two

- Extending class definitions (inheritance)
- Virtual methods
- Virtual classes
- Public and private classes
- Lab: Creating an advanced OO testbench

Dynamic Arrays and Scoreboards

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

Process Synchronization

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

Day Four

Constrained Random Value Generation

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

Functional Coverage

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

Assertions for Verification Engineers

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

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 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. 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 (such as from the SystemVerilog Verification Foundations workshop)

Course length: 3-days on-site. 4-days *eTutored™ live*. 3 to 60 days *eTutored™ self-paced*.

Intended Audience and Objectives

This workshop is for design and/or verification engineers who will be creating test environments for the verification of digital designs. At the conclusion of this workshop, engineers will understand how to take full advantage of the verification capabilities in the SystemVerilog language in order to develop object-oriented testbenches that utilize constrained-random verification methodologies, functional coverage, mailboxes and scoreboarding.

Prerequisites (essential)

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 Verification Foundations* course or equivalent.

Included Materials

- Comprehensive binder with all PowerPoint slides (printed in color), lab instructions, and supplemental white papers. (*eTutored™ self-paced* courses use an eBook and other online materials instead of a training binder.)
- A handy “*Verilog HDL Quick Reference Guide*” (\$15 value).
- All lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

The Cadence *Incisive™*, Synopsys *VCST™* or the Mentor Graphics *Questa™* simulator will 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.

Syllabus — Mastering UVM (Universal Verification Methodology)

Day One

UVM Overview

- The purpose of UVM
- Transaction based verification
- UVM testbenches
- UVM test phases
- UVM objects and components
- Lab: Examining a UVM testbench

UVM First Look

- uvm_object class
- uvm_component class
- UVM class library
- UVM utilities
- UVM configuration
- Lab: Defining a simple UVM testbench

Review of SystemVerilog Verification Constructs

- SystemVerilog's class data type
- Class methods
- Class inheritance
- Virtual methods
- Virtual classes
- Constrained random value generation
- Functional coverage
- Lab: Creating a simple OO testbench

UVM Transactions and Sequences

- Transaction Level Modeling (TLM)
- TLM ports, exports, and analysis ports
- Sequences of transactions
- UVM sequencers
- Connecting sequences to drivers
- Lab: Defining a sequence of transactions

Day Two

UVM Drivers, Monitors and Agents

- UVM drivers
- Virtual interfaces
- UVM monitors
- UVM agents
- Active and passive modes
- Lab: Defining and configuring a UVM agent

UVM Environments, Predictors and Scoreboards

- Scoreboard fundamentals
- Predicting expected results
- Comparing expected and actual results
- Encapsulation in a test environment
- Lab: Verifying outputs of a (faulty) DUT

UVM Tests and Top-level Testbenches

- Putting everything together in a UVM test
- Top-level modules
- Running multiple tests
- Using SystemVerilog program blocks with UVM
- Reuse and scalability considerations
- Lab: Define and run a series of UVM tests

UVM Factories

- Understanding the UVM factory
- Registering verification components
- Using the UVM Configuration database
- UVM messages and reports
- Lab: Working with factories

Day Three

UVM Functional Coverage

- Types of coverage
- Where coverage can be specified
- Enabling coverage
- Lab: Defining and examining coverage

UVM Compound Sequences

- Sequences of sequences
- Layered sequencers
- Virtual sequencers
- Configuration of compound sequences
- Lab: Working with layered sequencers

UVM Register Layer

- Register packages
- Registers and register files
- Memories
- FIFOs
- Register stimulus generation
- Backdoors
- Lab: working with registers

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 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.

Course length: 2-days on-site. 3-days *eTutored™ live*. 2 to 60 days *eTutored™ self-paced*.

Intended Audience and Objectives

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

Prerequisites (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 Verification Foundations* course or equivalent.

Included Materials

- Comprehensive binder with all PowerPoint slides (printed in color), lab instructions, and supplemental white papers. (*eTutored™ self-paced* courses use an eBook and other online materials instead of a training binder.)
- The reference book “*SystemVerilog Assertions Handbook*” by Ben Cohen, et al. (\$150 value).
- A handy “*Verilog HDL Quick Reference Guide*” (\$15 value).
- All lab files, including example solutions that illustrate proper and efficient coding styles.

Software Tools Used

The Cadence *Incisive™*, Synopsys *VCS™* or the Mentor Graphics *Questa™* simulator will be used for labs.

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.

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