Comprehensive SystemVerilog
SystemVerilog design and verification for Verilog users

Course Description:
SystemVerilog (IEEE 1800™) is a significant new language based on the widely used and industry-standard Verilog® hardware description language. The SystemVerilog extensions enhance Verilog in a number of areas, providing productivity improvements for RTL designers, verification engineers and for those involved in system design and architecture.

Comprehensive SystemVerilog provides a complete and integrated training program to fulfill the requirements of verification engineers and those wishing to evaluate SystemVerilog’s applicability to both design and verification applications. It is structured to enable engineers to develop their skills to cover the full breadth of SystemVerilog features for both design and verification. This includes the requirements of verification engineers who wish to exploit the potential of class-based verification and object oriented techniques using SystemVerilog, as well as RTL coding, assertions and test benches. Design engineers who do not intend to use SystemVerilog for class-based verification should attend the shorter training course SystemVerilog for Designers, which shares the same content as Days 1 to 3 of Comprehensive SystemVerilog.

Workshops comprise approximately 50% of class time, and are based around carefully designed exercises to reinforce and challenge the extent of learning.

Level: Standard Level
Training Duration: 5 days

Who should attend?
• Design engineers who wish to make full use of SystemVerilog’s class-based verification capabilities for test bench development as well as learning SystemVerilog for RTL design.
• Verification engineers aiming to deploy coverage driven verification approaches for the first time using SystemVerilog
• Verification engineers wishing to migrate to SystemVerilog class-based verification from other established verification languages and test bench automation techniques
• Engineers and managers who wish to evaluate the full range of SystemVerilog's capabilities for design and verification
• EDA support engineers who wish to gain a comprehensive understanding of how their customers' engineering teams can most productively use SystemVerilog in both design and verification domains

Prerequisites:
• A good working knowledge of Verilog is essential.
• For engineers with no Verilog knowledge but with working experience of VHDL, there is a Fast Track Verilog for VHDL Users class in a format tailored to equip delegates with the necessary foundation for SystemVerilog. This class is usually scheduled in the same location prior to the Comprehensive SystemVerilog course. See the Course Schedule for the latest scheduling information.
• For onsite courses, precursor training in Verilog can be tailored to the specific team profile and combined with appropriate SystemVerilog modules to fully address team needs (see Modular SystemVerilog). What will you learn? The course is structured into four distinct sections.

What will you learn? The course is structured into four distinct sections.
• Fundamentals of SystemVerilog for Design (1¼ day) lays the foundations for learning the SystemVerilog language for design and for verification, including the synthesizable RTL constructs.
• SystemVerilog Assertions (¾ day) teaches the principles of assertion-based verification and design, key features of the SystemVerilog assertion language for creating your own custom assertions, and how to package and deploy libraries of assertion checkers.
• Module-based SystemVerilog Verification (1 day) shows how to use SystemVerilog to build effective block-level testbenches, building on best-practice testbench architecture based on Verilog modules.
• Class-based SystemVerilog Verification (2 days) describes how to write sophisticated object-oriented testbenches using SystemVerilog’s testbench automation capabilities, which support a constrained-random, coverage-driven verification methodology. These features enable you to write testbenches at higher levels of abstraction and be more productive than is possible with standard hardware description languages. The material leverages Doulos’s years of experience in teaching object-oriented verification concepts, making these challenging topics accessible to engineers with a wide variety of backgrounds and providing ideal preparation for your subsequent adoption of a sophisticated verification methodology.

Comprehensive SystemVerilog provides the essential SystemVerilog language foundations for learning the OVM, VMM, or UVM verification methodologies. Doulos also offers follow-on training in each of these specific methodologies. For further details, see OVM Adopter Class, VMM Adopter Class, and UVM Adopter Class.

Structure and Content
SystemVerilog for Designers  2. The SystemVerilog data type system
1. Fundamentals of SystemVerilog for Design
• enum

Continued …
Comprehensive SystemVerilog
SystemVerilog design and verification for Verilog users

Course Content:

... Continued

- typedef
- struct union
- packed/unpacked
- packages and $unit
- using arrays in SystemVerilog
- array and structure literals, assignment patterns

3. Nets and variables
- Key changes in Verilog-2005 and SystemVerilog
- continuous assignment to variables
- modified driver and connection rules
- data types on ports and nets

4. Modules and processes
- Port connection shorthand
- type parameters
- synthesis idioms for processes
- miscellaneous improvements to the language

5. Design applications of interfaces
- The interface construct
- interfaces to encapsulate communication
- modports
- synthesis of interfaces and modports
- imported functions for design

6. SystemVerilog Assertions

7. Introduction to assertions
- Assertions, properties, sequences
- clocking and sampling
- property implication
- uses of assertions
- simulation of assertions
- formal tools

8. Assertion methodology
- Methodology consequences of assertion-based design and verification
- assertion and assumption
- benefits of assertions to the designer
- protocol checkers

9. A brief introduction to SVA syntax
- Writing simple assertions of your own
- sequences and the # operator
- repetition and time ranges
- sequence fusion
- overview of temporal operators
- local variables and actions in assertions

10. Packaging Assertions
- Assertions in interfaces and modules
- the bind construct
- deploying verification IP, particularly assertion-based IP

11. Module-based SystemVerilog Verification

12. Verification for designers
- Bus functional models
- testbench architecture in classic Verilog
- stimulus and response timing

13. Using SystemVerilog to construct module-level testbenches
- Clocking blocks to manage timing
- testbench applications of interfaces
- task and function enhancements in SystemVerilog
- decoupling test cases from the testbench

14. Dynamic data types
- strings
- queues
- dynamic arrays
- associative arrays
- queue and array methods
- foreach loop

15. Testbench automation
- Introduction to testbench automation concepts
- randomisation, checking and coverage
- the need for constraints
- randomisation of stimulus data using std::randomize and traditional Verilog distribution functions
- procedural randomisation: randcase, randsequence
- collecting functional coverage data

Class-based SystemVerilog Verification

16. Introducing classes
- SystemVerilog’s class syntax
- describing stimulus data and a stimulus generator
- randomization of class members (without constraints)
- objects and references
- constructors and new
- shallow copy using new
- writing a custom copy method

17. Hooking classes to the DUT
- Dynamically-constructed test environment vs. statically-elaborated DUT and test harness
- using virtual interface and class-based BFM s
- the role of clocking and program blocks
- appropriate structure for DUT, clock generators and other structural elements
- constructing and launching the test environment using program+initial
- simple class-based testbench architecture

18. Varying the Stimulus
- Generator template objects
- introduction to constraints
- implication constraints

Continued...
Comprehensive SystemVerilog
SystemVerilog design and verification for Verilog users

Course Content:

... Continued

• derived classes
• upcasting and the is-a relationship
• virtual methods

19. Components and Channels
• FIFO channels to decouple components
• base class for transaction data
• downcasting and $cast
• parameterized classes and macros for specialization
• running self-contained components with fork...join

20. Reusable Testbench Components
• Maintaining a component instance hierarchy
• virtual base class for components
• launching a task with fork...join_none
• testbench component architecture
• preview of standard methodologies (OVM, VMM)

21. Monitor and Check Components
• Passive monitors and unbounded FIFOs
• checker components and scoreboards
• stopping the test cleanly
• semaphore for mutual exclusion

22. Coverage in Classes
• Coverage-driven TBA methodology
• coverage planning as the first step in a verification process
• analysing and interpreting coverage data
• SystemVerilog coverage constructs in detail
• covergroup sampling
• per-instance coverage in testbench components
• covergroup options
• transition and cross coverage
• controlling bins structure
• coverage reports