SystemVerilog for Verification Specialists

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.
SystemVerilog for Verification Specialists provides a 4 day training program to fulfil the requirements of verification engineers or those wishing to evaluate SystemVerilog's applicability for complex verification application. It is structured to enable engineers to develop their skills to utilise the full breadth of SystemVerilog features for verification. This includes how to exploit the potential of class-based verification and object oriented techniques using SystemVerilog, as well as application for standard test bench development and module-based verification.
The course assumes Verilog knowledge but no prior SystemVerilog knowledge. VHDL users preparing to use SystemVerilog should consider preparatory training with the 2 day Fast Track Verilog for VHDL Users.
The course includes an introduction to UVM (and OVM) but full scope project readiness in UVM requires follow-on training with the 3 day UVM Adopter Class.
Design engineers (FPGA or ASIC) who intend to use SystemVerilog for RTL design and basic test bench development should attend the companion training course SystemVerilog for FPGA/ASIC Design.
Workshops comprise approximately 50% of class time, and are based around carefully designed exercises to reinforce and challenge the extent of learning.
Doulos is an independent company, enabling delegates to receive the benefit of objective tuition while learning in the context of their chosen tool and methodology.

Level: Standard Level
Training Duration: 4 days

Who should attend?
- 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

Prerequisites:
A good working knowledge of Verilog is essential.
For engineers with no HDL knowledge or experience the Comprehensive Verilog course or equivalent is an essential precursor.
For engineers with no Verilog knowledge but with working experience of VHDL, Doulos offer 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 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.
- Fundamentals of SystemVerilog for Verification (½ day) provides verification engineers with the necessary background in SystemVerilog to embark on the remaining verification modules. (Engineers who do not have a background in software programming may be better served by attending Comprehensive SystemVerilog.)
- SystemVerilog Assertions (½ 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.

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

SystemVerilog for Verification Specialists

Structure and Content

1. Fundamentals of SystemVerilog for Verification

2. Types and Arrays
- enum
- typedef
- struct
- union
- packed/unpacked
- packages
- data types on ports and nets
- using arrays in SystemVerilog
- array and structure literals, assignment patterns

3. Modules and Interfaces
- Continuous assignment to variables
- modified driver and connection rules
- Port connection shorthand
- type parameters
- miscellaneous improvements to the language
- the interface construct
- modports

4. SystemVerilog Assertions

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

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

7. 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

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

9. Module-based SystemVerilog Verification (Day 2)
- Bus functional models
- testbench architecture in classic Verilog
- stimulus and response timing

10. 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

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

12. 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

13. 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

14. Hooking classes to the DUT
- Dynamically-constructed test environment vs. statically-elaborated DUT and test harness
- using virtual interface and class-based BFMs
- 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

15. Varying the Stimulus
- Generator template objects
- introduction to constraints
- implication constraints
- derived classes
- upcasting and the is-a relationship
- virtual methods

16. 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

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

Continued …

SystemVerilog for Verification Specialists

Course Content:

… Continued

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

19. 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