SystemVerilog for Designers

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 Designers provides a compact and focused training program to fulfil the requirements of design groups. It is structured to enable designers to develop their capability by exploiting SystemVerilog features for mainstream design and verification requirements, including RTL coding, assertions and test benches. It is not intended to fulfil the deeper requirements of verification specialists who will wish to exploit the potential of class-based verification and object-oriented techniques using SystemVerilog. (Such requirements are covered in Days 4 and 5 of the Comprehensive SystemVerilog course, which includes the content of SystemVerilog for Designers as its first three days.)
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: 3 days

Who should attend?
- Design engineers with a working knowledge of RTL design and basic verification techniques (see Verilog pre-requisite below) who wish to migrate to or use SystemVerilog for RTL design, assertions and block-level test benches
- Engineers and managers who wish to evaluate SystemVerilog for ASIC or FPGA design and block-level verification
- EDA support engineers who wish to understand how their customers' design teams can most productively use SystemVerilog

Prerequisites:
A 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 offers 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.
The course is structured into several distinct sections.
- Fundamentals of SystemVerilog for Design trains engineers in the practical use of SystemVerilog for synthesisable RTL design, and lays the foundations for use of the language in verification.
- SystemVerilog Assertions 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 shows how to use SystemVerilog to build effective block-level testbenches, building on best-practice testbench architecture based on Verilog modules.

Training materials
Doulos training materials are renowned for being the most comprehensive and user friendly available. Their style, content and coverage are unique in the HDL training world, and have made them sought after resources in their own right. The materials include:
- Fully indexed class notes creating a complete reference manual
- Workbook full of practical examples and solutions to help you apply your knowledge

Structure and Content

1. Fundamentals of SystemVerilog for Design
2. The SystemVerilog data type system
- enum
- typedef
- struct
- union
- packed/unpacked

Continued …

SystemVerilog for Designers

Course Content:

... Continued

- 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 design teams
- 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