

SystemVerilog Synthesis for Verilog Design Engineers

Overview

SystemVerilog Synthesis for Verilog Design Engineers is a 2-day intensive workshop covering the hardware modeling portions of the latest IEEE 1800-2009 SystemVerilog standard. SystemVerilog is an exciting and powerful set of extensions to the Verilog Hardware Description Language, that enable efficiently modeling and verifying large, complex designs. The synthesizable subset of SystemVerilog presented in this workshop allows engineers to model more functionality with fewer lines of code, and at the same time ensure that models synthesize correctly and optimally. Simply put, SystemVerilog makes the design engineers life much easier and enables greater design productivity. This workshop is *not* a simple (and boring) language course. The focus of this workshop is on the proper usage of the extensive set of advanced modeling capabilities offered by SystemVerilog. Several labs reinforce the principles presented, with forty percent of the class time devoted to hands-on experience.

A *comprehensive student guide*, a handy “*Verilog HDL Quick Reference Guide*” (\$15 value) are included with the course materials.

Course Objectives

This workshop will enable design engineers who use Verilog to immediately be productive modeling complex hardware designs using the SystemVerilog extensions to the Verilog language.

Intended Audience

This workshop is for experienced Verilog design engineers. The course presupposes a working knowledge of Verilog, and only covers the SystemVerilog enhancements to Verilog. Design engineers who do not have a strong background in Verilog should instead take Sutherland HDL’s “*Comprehensive Verilog and SystemVerilog for Design and Synthesis*” workshop, or combine this workshop with Sutherland HDL’s “*Accelerated Verilog Primer*”.

Software Tools Used

The Cadence *NC-Verilog*[™], Synopsys *VCS*[™] or Mentor Graphics *ModelSim*[™] simulator will be used for labs.

Prerequisites (essential)

Knowledge of the Verilog HDL is mandatory. Without this background, students cannot fully benefit from this course. This course presents the substantial extensions to the Verilog HDL, and assumes a working knowledge of Verilog.

Workshop Locations

This workshop can be presented on-site, at your facilities. We also offer several open-enrollment workshops throughout the year. For more information, please refer to our web page, www.sutherland-hdl.com, or call us at +1-503-692-0898.

Syllabus — SystemVerilog Synthesis for Verilog Design Engineers

Day One

Introduction to SystemVerilog

- The purpose of SystemVerilog
- Backward compatibility with Verilog
- The SystemVerilog synthesis subset

Design Hierarchy

- External declarations
- Packages
- Nested modules
- Enhanced module instances
- Parameterized data types
- Lab: Using globals and packages

SystemVerilog Data Types and Logic Values

- New data types
- Casting
- User-defined types
- 2-state modeling techniques and “gotchas”
- Lab: Modeling with 2-state data types

Enumerated Types

- Enumerated types
- Specifying label values
- Assignments with enumerated types
- Static and dynamic casting
- Enumerated type methods
- Lab: Modeling a state machine with enumerated types

Data Aggregates: Arrays, Structures and Unions

- Packed arrays
- Unpacked arrays
- Assigning to arrays / copying arrays
- Structures
- Unions
- Tagged unions
- Bit-stream casting
- Lab: Using structures, unions and arrays

Day Two

SystemVerilog Procedural Blocks

- Combinational logic procedural blocks
- Latched logic procedural blocks
- Sequential logic procedural blocks
- Why specialized procedural blocks are important
- Task and function enhancements
- Passing arguments by reference
- Lab: Using specialized procedures

Programming Statements and Operators

- New operators
- Enhanced looping constructs
- Bottom testing loops
- New jump statements
- Unique and priority decision statements
- Eliminating simulation versus synthesis mismatches
- Lab: Modeling a high-level ALU

SystemVerilog Interfaces

- Interface definitions
- Defining module ports and directions
- Tasks and functions in interfaces
- Procedural code in interfaces
- Synthesizing interfaces
- Parameterized interfaces
- Verification paradigms with interfaces
- Lab: Using interfaces

Assertions for Design Engineers

- Assertion concepts
- Immediate assertions
- Concurrent assertions and sequences
- Basic sequence definitions
- Disabling assertions during reset
- Controlling assertion messages
- Lab: Using assertions in interfaces