SUTHERLAND HDL

22805 SW 92nd Place., Tualatin, Oregon 97062 • Phone: +1-503-692-0898 • URL: www.sutherland-hdl.com

Verilog/SystemVerilog for Design and Synthesis

Overview

Verilog/SystemVerilog for Design and Synthesis is a comprehensive workshop covering the complete Verilog Hardware Description Language and the synthesizable portions of SystemVerilog, including user-defined types, enumerated types, structures, and self-verifying decision statements. The workshop integrates in topics from *"Verilog and SystemVerilog Language Primer"* and adds detailed discussion and labs on best coding practices for writing synthesizable RTL models that work correctly in both simulation and synthesis. Special attention is given to language subtleties, such as how blocking and non-blocking assignments, programming statements and operators 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 Digital Signal Processor (DSP).

Course Length: 4-days on-site, 5-days *eTutored™ live*, 2 to 30 days *eTutored™ self-paced*.

Intended Audience and Objectives

This workshop is for digital engineers who will be designing with Verilog and SystemVerilog. Students will be immediately productive in 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.

Prerequisite Knowledge (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

- Full-color training binder with copies of all lecture slides, lab instructions, and supplemental information. (*eTutored™ self-paced* courses include an eBook instead of a training binder.)
- *"Verilog HDL Quick Reference Guide"* booklet.
- 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 Aldec *Riviera-Pro™*, Cadence *NC-Verilog™*, Synopsys *VCS™*, or Mentor Graphics *ModelSim™* simulator, and the Cadence *Encounter RTL Compiler™*, Synopsys *DC™* or *SynplifyPro™*, or Mentor *Precision™* synthesis compiler.

Comments From Students

"An excellent comprehensive study of Verilog, with perspectives on Verilog as a simulation, modeling and synthesis language, backed with valuable labs."

"I just wanted to say how great your Verilog and SystemVerilog training was. Thank you for a brilliant Verilog/SystemVerilog education. I don't think designers should be allowed to write code until they have this course!"

"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.

Licensed Training Materials

Sutherland HDL's training materials can be licensed for use in internal training programs. Licensed materials include presentation PowerPoint files, printable PDF files, and lab files. Train-the-trainer services are also provided.

SUTHERLAND HDL

22805 SW 92nd Place., Tualatin, Oregon 97062 • Phone: +1-503-692-0898 • URL: www.sutherland-hdl.com

Syllabus — Verilog/SystemVerilog for Design and Synthesis

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

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

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

SystemVerilog Interfaces
- Using interfaces to simplify inter-module connections
- Specifying interface views (modports)
- Using tasks and functions in interfaces
- Lab: model and verify a master/slave interface bus

Design Verification Fundamentals
- 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 SystemVerilog standard enables modeling larger, more complex designs, and, at the same time, simplifies writing models that synthesize correctly. 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.