Verilog - Expanded

4 Days, Basic Level

"Excellent course! The instructor made a great presentation."

Overview
The Verilog hardware description language plays a key role in design flows for ASICs and FPGAs. Yet many designers lack skill with this language.

This course provides a comprehensive presentation of the main features of the Verilog language and the extensions introduced by Verilog 2001 (IEEE Std. 1364-2001). The course will bridge the gap between the designer's prior background in digital logic (schematics, Boolean equations, truth tables, state transition graphs, ASM charts) and related Verilog constructs used in modern design flows. Several examples (e.g., FIFO, three-state bus, synchronization across clock domains) will be used to illustrate language features, including an introduction to correct modeling styles for synthesis and verification. A comprehensive lab on the final day will reinforce the material by using a real-world design.

Benefits
Upon completion of this course, students will be able to:

- write models of combinational and sequential logic
- write and execute a test plan and develop a testbench for verifying a model
- write models of pipelined and other concurrent logic
- write alternative models of finite-state machines
- write RTL and algorithm-based models of combinational and sequential logic
- write parameterized, re-useable models
- select alternative state assignments for finite state machines
- write race-free models of state machines and datapath controllers
- write synthesis-friendly and/or simulation-efficient models
- write a self-checking testbench
- write, simulate, and verify structural and behavioral models of digital logic
- exploit the features of Verilog 2001

Intended Audience
Verilog for Hardware Designers is recommended for people with little or no knowledge of Verilog who need to gain a working understanding of the language for hardware design.

Prerequisites
Students need to be familiar with the basics of digital design such as shift registers, adders, multiplexers and finite state machines.

Suggested follow-on course:

Advanced Verilog Coding Styles for Synthesis & Verification

Training Approach
This is an intensive, interactive course, which is approximately 50% lecture and 50% lab. Questions are highly encouraged. The final day concludes with a comprehensive lab project where the students will have a chance to take a design from beginning to end using most of the information they have learned in the class.

Course Outline

Day 1: Introduction to Modeling and Verification with Verilog

Introduction and Overview
- Course Objectives
- Verilog-Based ASIC design flow
- Verilog-Based FPGA Design Flow
- Benefits of HDL-Based Design

Getting Started with Verilog
- Characteristics of HDLs
- Lexical conventions
- Representation of numbers
- Identifiers
- Scalar and vector signals
- Verilog primitives
- Port rules for primitives
- Verilog's logic system
- Expressions and operators
- Data types (nets and registers)
- Signal contention
- Memories and strings

Structural Models of Combinational and Sequential Logic
- Design encapsulation with modules
- Module ports
- Module instantiation
- Port connections by position and name
- Unconnected ports
- Arrays of primitives and modules
- User-defined primitives
- modes (input, output, inout)

Hierarchical Decomposition and Top-down design
- Hierarchical decomposition
- Nested modules
- Design hierarchy
- Example: four-bit-slice 16-bit adder

Propagation Delay
- Delay models in Verilog
- Gate propagation delay
- Rules for propagation delays
- Inertial delay for gates
- Wire delay

Testbenches, Simulation, and Model Verification
- Importance of simulation
- Event-driven simulation
- Basic simulator organization
- Zero-delay simulation
- Top-down design
- Testbench elements
- Stimulus generator
- Response monitor
- Unit-delay simulation
- GUI-based output
- Development of a test plan

Lab Exercises

Day 2: Behavioral Modeling with Verilog

Language Constructs for Behavioral Modeling

- Behavioral models and procedural statements
- Constructs for behavioral models
- Assuagements to variables
- Cyclic and single-pass behaviors
- Timing control
- Delay (#), event (@), and wait
- Procedural (blocked) assignment
- Nonblocking assignment
- Procedural continuous assignment
- Sequential and parallel blocks
- Conditional and case statements
- Loops (for, repeat, while, forever)
- Disabled blocks

Behavioral Models of Combinational Logic

- Implicit combinational logic
- Three-state bus model
- Level-sensitive behavior
- Cyclic behavior (always block)
- RTL models
- Algorithm-based models

Behavioral Models of Sequential Logic

- Models of level-sensitive sequential logic
- Models of edge-sensitive sequential logic
- Synchronous and asynchronous reset
- Concurrent behaviors
- Indeterminism, concurrent behaviors, and race conditions
- Nonblocking assignments and concurrent RTL models
- Nonblocking vs. blocking assignments

Additional Topics for Testbenches

- Modeling with single-pass behaviors
- Named events
- Black holes for simulators
- Race conditions
- Intra-assignment delay
- Model portability and reuse
- Inline and indirect parameter redefinition

Tasks and Functions

- Examples and rules
- Overview of system tasks

Lab Exercises

Day 3: Sequential Machines, Datapath Controllers, and Test Strategies

Finite-State Machines

- Modeling styles for Mealy FSMs
- Modeling styles for Moore FSMs
- Guidelines for modeling FSMs
- Registered outputs
- State transition graphs
- Algorithmic state machine charts

Datapath Controllers

- ASMD charts
- Preventing race conditions
- Comprehensive design examples

Sequential Algorithms and Pipelined Machines

- Rules for race-free design
- Comprehensive examples

Compiler Directives and System Functions and Tasks

- Timescale directive
- Simulation display
- Timing verification
- File I/O

Testing for Model Verification

- Verification plan
- Approaches to verification
- Top-down verification
- Bottom-up verification
- Integration strategy
- Hierarchical dereferencing
- Common errors

- Self-checking testbench
- Exhaustive pattern generators
- Regression testing
- Simulation efficiency

Lab Exercises

Day 4: Advanced Topics

Specify Blocks

- Module path delay
- Pin-pin connections
- Edge-sensitive module paths
- State-dependent module paths
- Path polarity
- Specify block parameters
- Pulse rejection on module paths
- Comparison of parameters

Simulator Delay Modes

Programming Language Interface (PLI)

Standard Delay format (SDF)

Verilog 2001

- ANSI C style changes
- Initialization of variables
- Re-entrant tasks
- Recursive tasks
- Constant functions
- Variable part selects
- Signed data types, ports, literal integers, functions
- Arithmetic shift operator, Exponentiation operator
- Sensitivity list for event control
- Sensitivity list for combinational logic
- Parameters
- Instance generation

Lab Exercises

For more information, contact:

Tom Wille • tw@tm-associates.com
TM Associates, Inc. • 503-656-4457
www.tm-associates.com