Introduction to Verilog-2001 & Best Coding Practices

by

Recognized Verilog & SystemVerilog Guru, Cliff Cummings of Paradigm Works, Inc.

Cliff Cummings is the only Verilog & SystemVerilog Trainer who helped develop every IEEE & Accellera Verilog, Verilog Synthesis and SystemVerilog Standard.

Course Summary

Course Description

Sunburst Design – Introduction to Verilog-2001 & Best Coding Practices is a 2-day, fast-paced, intensive course on the IEEE 1364-2001 Verilog Hardware Description Language and its use for hardware design and verification. This course emphasizes RTL modeling and efficient testbench techniques while teaching Verilog-2001 syntax. Unlike traditional Verilog courses that start with gate-level design and work towards behavioral coding styles, the Verilog-2001 Design course focuses on important behavioral design and synthesis coding styles first, along with high-level verification techniques, to instill and reinforce good coding habits early and often during the course. This course prepares engineers to take SystemVerilog training.

Upon completion of this course, students will:

- Write efficient Verliog-2001 RTL models
  - have a detailed understanding of the important Verilog-2001 language features
  - understand the necessary constructs for
    - design, synthesis and verification
    - simulation of gate-level netlists
  - know how to write and simulate
    - complex hardware models
    - simple synthesizable models
    - self-checking testbenches
  - know how to run efficient Verilog simulations
    - be immediately productive at modeling and simulating complex Verilog designs

Duration	2 days
Breakdown	70% Lecture, 30% Lab
Level	Beginner to Intermediate
Prerequisites	No prior HDL experience required, but knowledge of digital design concepts is strongly recommended.
Online Details	The course delivery can be in-person or virtual (virtual courses are convenient for both U.S. and non-U.S. engineers).

Course Syllabus

Half Day #1

(1) Introduction to Verilog Modeling

- An introduction and overview of major Verilog-2001 modeling basics.
  - Overview of Verilog Resources
  - Modules
  - Port and net declarations
  - V2K1 ANSI-C style module headers
  - Instantiation with positional and named ports
  - Procedural blocks: initial & always
  - Hierarchy
  - Introduction to synthesis design flows
  - Power-user guidelines

(2) Verilog HDL Syntax & Semantics

- Detailed instruction of important Verilog-2001 (V2K1) language syntax.
  - Good formatting = fewer bugs & better documentation
  - Comments
  - V2K1 attributes
  - Identifier names
  - Name scopes
  - Language tokens
  - Numbers and logic values
  - Net & variable types
  - Scalars & vectors
  - Multi-dimensional arrays
  - Port declaration styles
  - Parameters
  - Verilog's 4+ logic values

(3) Design Verification & Running Simulations

- An introduction to writing Verilog testbenches and running Verilog simulations.
  - Strength basics
  - Simulation times, delays, timescales and time formatting
  - Writing Verilog testbenches
  - Repeat-loop & forever loop
  - Important compiler directives

- Display and formatting commands
- System tasks for simulation control
- Using multiple Verilog source files & Verilog command files
- Running a Verilog simulation
- Lab: basic testbench development

Half Day #2

(4) Continuous Assignments, Operators, Verification & Running Simulations

- Detailed discussion of continuous assignments with design examples, followed by an overview of Verilog-2001 operators, also with examples.
  - Continuous assignments
  - Procedural continuous assignments (do not use these!)
  - Required net declarations
  - ?: conditional operator

(5) Programming Statements & Timescales

- Detailed discussion of blocking and nonblocking assignments, followed by an overview of Verilog-2001 programming statements with examples. This section concludes with a discussion of Verilog timescales and their impact on simulation efficiency.
  - Verilog operators - arithmetic, bitwise, logical, unary reduction, more
  - Sequential & parallel statement groups
  - Blocking & nonblocking assignments (introduced)
  - Time delays
  - Level-sensitive timing controls
  - Edge-sensitive timing controls
  - Sensitivity lists (V2K1)
  - If-else & case statements
  - For, while, repeat & forever loops
  - Tasks, functions and automatic (V2K1)
  - Rise, fall, min, max delays
  - `timescale & $timeformat
  - Lab: (optional) `timescale & $timeformat capability and efficiency

Half Day #3

(6) Combinational Logic Modeling

- Behavioral & synthesizable coding styles for modeling combinational logic. Includes multiple Verilog-2001 enhancements.
  - Sensitivity lists
  - Continuous assignments
  - Procedural combinational blocks
  - V2K1 comma-separated and @* sensitivity lists
  - Inertial & transport delays
  - Correct methods for adding behavioral timing delays

(7) Sequential Logic Modeling

- Behavioral & synthesizable coding styles for modeling sequential logic
  - Sensitivity lists
  - Flip-flops and latches
  - Synchronous and asynchronous inputs
  - Where to add timing delays
  - Lab: combinational model and verify an 8-bit ALU
  - Lab: sequential model and verify a pipeline
  - Lab: (optional) extra labs

Half Day #4

(8) RAM and ROM Modeling

- Behavioral coding styles for modeling RAMs and ROMs
  - Modeling memories
  - Array declarations
  - System tasks to load memories
  - Preferred coding style for read operations
  - Preferred coding style for write operations
  - Testing bi-directional ports
  - Basic timing constraints
  - Lab: model and verify a RAM

(9) Structural Netlists

- How to construct a hierarchical Verilog netlist.
  - Design hierarchy
  - Module instantiation
  - Port & net mismatches
  - Parameterized models
  - Redefining parameters
  - V2K1 Multi-dimensional arrays
  - Arrays of Instance
  - Generate statements
  - Lab: model and verify a hierarchical design

(10) Verilog Gates and SDF Timing

- Introduction to gate primitives, User Defined Primitives, specify blocks and SDF backannotated timing.
  - Component models
  - Gate primitives
  - Time delays
  - User Defined Primitives (UDPs)
  - Specify blocks
  - Timing checks
  - SDF back annotation - $sdf_annotate command
  - Optional Lab: SDF Backannotation
  - Optional Lab: UDP (User Defined Primitive)