

Accelerated 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 - Accelerated Introduction to Verilog-2001& Best Coding Practices is a 1-day fast-paced intensive course on the Verilog-2001 syntax, usage and best-known coding styles.

The 1-day Accelerated Introduction to Verilog-2001 & Best Coding Practices is intended for design and verification engineers who require an accelerated introduction to the Verilog-2001 RTL syntax & coding styles, prior to taking SystemVerilog training.

Students will learn, use, and understand the following:

- Write efficient Verilog-2001 RTL models
 - includes a broad introduction to important Verilog syntax concepts
 - includes good Verilog coding techniques
 - includes fundamental Verilog testbench techniques
 - includes Best Coding Practices for Verilog coding

Duration	1 day
Breakdown	80% Lecture, 20% Lab
Level	Beginner
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.
 - Comments
 - V2K1 attributes
 - Identifier names
 - Name scopes
 - Language tokens
 - Numbers and logic values
 - Net & variable types
 - Verilog's 4+ logic values
 - Strength basics

(3) 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. An introduction to writing Verilog testbenches and running Verilog simulations.
 - Continuous assignments
 - Procedural continuous assignments (do not use these!)
 - Required net declarations
 - Verilog operators
 - Strength basics
 - Writing Verilog testbenches
 - Important compiler directives

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

Half Day #2

(4) 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.
 - 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

(5) Basic RTL Modeling

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