Expert SystemVerilog Design & Synthesis

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

There are two versions of this course. If engineers already know SystemVerilog, they should choose ***this*** 2-day Sunburst Design - Expert SystemVerilog Design & Synthesis course.

The course objective is to give engineers world-class SystemVerilog language, expert design & synthesis training using award-winning materials developed by renowned Verilog & SystemVerilog Guru, Cliff Cummings

Duration	2 days
Breakdown	60% Lecture, 40% Lab
Level	Expert
Prerequisites	**This is an expert SystemVerilog training class that assumes engineers already have a good working knowledge of SystemVerilog Fundamentals.** Engineers with no prior HDL training or experience will struggle in this class.
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) Latches & Priority Encoders. Unique & Priority - full_case & parallel_case

- Introduction to Verilog synthesis design flows. Detailed description of two synthesis problem areas: latches and priority encoders. Detailed description of the synthesis directives "full_case" and "parallel_case", and why they should generally be avoided. This section also details how unique and priority are the new SystemVerilog replacements for the dangerous "Evil Twins," full_case and parallel_case.
  - always_latch
  - always blocks & sensitivity lists
  - Generating latches
  - Generating priority encoders
  - Latch & priority encoder guidelines
  - full_case parallel_case, "the Evil Twins"
  - What is full_case?
  - What is parallel_case?
  - unique & priority case
  - unique & priority if
  - unique0 (SystemVerilog-2009 enhancement)
  - One example using case modifiers (+ 2 more reference examples)
  - LABS: simple SystemVerilog combinational and sequential logic labs

(2) Combinational Logic I & II

- RTL coding styles for combinational logic, including problems and inefficiencies that arise from poor coding styles. Numerous combinational labs identify potential problem areas associated with common combinational coding styles. Verilog-2001 (V2K1) enhancements are discussed, including reasons to avoid over-usage of generate statements. An example of poor usage is I/O pad instantiation (use the more concise and better-supported Array of Instances). More combinational labs highlight many potential problem areas associated with common combinational coding styles.
  - always_comb
  - Continuous assignments
  - Always blocks
  - V2K1 @* and comma-separated sensitivity lists
  - Instantiated library elements
  - Instantiated primitives
  - Synthesizable and non-synthesizable Verilog constructs
  - Bitwise -vs- logical operators

- Tasks & functions
- Tri-state drivers
- Bi-directional buses
- Instantiating ASIC/FPGA library primitives
- Guidelines
- LABS: Combinational logic labs I & II

Half Day #2

(3) Sequential Logic

- This section covers coding styles for sequential logic. Efficient design inference using adders and other large resources is also detailed. Also discusses and includes the advantages and disadvantages of instantiation.
  - always_ff
  - Edge-sensitive sensitivity list
  - Basic asynchronous & synchronous resets
  - Additional flip-flop coding styles
  - Simulation/synthesis differences
  - Simulation efficiency
  - Register banks
  - Memories
  - Instantiating Blocks
  - Resource sharing
  - LABS: Sequential logic labs

(4) Synchronous & Asynchronous Reset Design

- Detailed material for selection and usage of synchronous and asynchronous reset design taken from actual design experiences.
  - Synchronous vs. asynchronous resets
  - Reset removal metastability
  - Asynchronous reset synchronizer circuitry
  - Reset distribution trees and techniques
  - Multiple clock domains reset synchronization

(5) SDF Back Annotation *(Optional - included for engineering teams that want to learn SDF timing-based simulation techniques)*

- This section details SDF (Standard Delay Format) file generation, writing Verilog gate-level netlists, gate-level simulations, and gate-level simulations using SDF timing (may be useful for full-board and system design simulations where Static Timing Analysis is not generally practical).

- SDF file basics
- Gate-level netlists
- SDF back annotation of gate-level designs
- Directory structures and command files for gate-level simulation
- Simulating with SDF files
- Conditional compilation of SDF files
- LAB: Post-synthesis gate-level netlist generation and simulation with SDF timing

Half Day #3

(6) Synthesis Labs Review

- An in-depth review of all the coding styles used in the combinational & sequential labs and the synthesized results. Conclusions are drawn about which coding styles infer the most efficient logic implementations.
  - Review of previous synthesis lab results

(7) SystemVerilog FSM Design Techniques *(Included in SystemVerilog Fundamentals Training*

- Seven different FSM coding styles, enhanced with new SystemVerilog constructs, are detailed and compared for coding and synthesis efficiency. Multiple FSM designs are benchmarked for coding style efficiency.
  - FSM coding goals
  - Moore & Mealy
  - Binary & Onehot
  - ASIC -vs- FPGA FSM design
  - Review proven FSM coding styles
  - Three always blocks - recommended
  - Two always blocks – recommended (combinational outputs)
  - One always block - avoid this
  - Four always blocks – recommended (better synthesis results)
  - Onehot case(1'b1) - recommended
  - Onehot parameters - avoid this
  - Output encoded - recommended
  - Coding & synthesis efficiency
  - SystemVerilog FSM enhancements
  - Advanced enumerated types
  - LABS: SystemVerilog FSM design lab

Half Day #4

(Expert Clock Domain Crossing (CDC) & FIFO Design Techniques Course)

(8) Multi-clock Domain Crossing (CDC) Design Techniques using SystemVerilog

- Very advanced design techniques from Cliff's award-winning presentations on the efficient implementation of multi-clock CDC designs. These materials are not specific to SystemVerilog, but solutions are shown using SystemVerilog syntax (advanced techniques that all design engineers should know - the stuff you did not learn in college).
  1. Metastability & synchronizers - synchronizing 1-bit signals
  2. Passing multiple control signals - synchronizing multi-bit signals or buses
     a) Consolidation
     b) Controlled- multicycle path formulations (MCP)
     c) FIFO synchronizer
     d) Gray codes & Gray code counters
  3. Design Partitioning - design & synthesis techniques
     a) Naming conventions
     b) Synthesis scripting & timing analysis issues
  4. Simulation issues
     a) X-propagation issues
     b) Synopsys command for SDF files
     c) Multi-SDF files
     d) ASIC/FPGA vendor cells and models
     e) Simulation model to expose synchronization problems
  5. LAB: MCP-controlled synchronization

(9) Multi-clock FIFO Design Techniques using SystemVerilog

- Very advanced design techniques from Cliff's award-winning presentations on the efficient implementation of FIFO designs. These materials are not specific to SystemVerilog, but solutions are shown using SystemVerilog syntax (advanced techniques that all design engineers should know - the stuff you did not learn in college).
  1. Multi-clock FIFO design - a large section on FIFO design and FIFO issues
  2. LAB: 2-clock FIFO