World Class Verilog & SystemVerilog Training

Sunburst Design - Advanced SystemVerilog for Design

by Recognized Verilog & SystemVerilog Guru, Cliff Cummings of Sunburst Design, Inc.

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

3 Days
70% Lecture, 30% Lab
Advanced Level

Course Objective

Simply stated, to give engineers *world class* SystemVerilog language & advanced design training using award winning materials developed by renowned Verilog & SystemVerilog Guru, Cliff Cummings

Upon completion of this course, students will:

- Write efficient synthesizable SystemVerilog-2005 RTL models
  - includes new SystemVerilog data types and capabilities
  - includes new SystemVerilog RTL and abstraction capabilities
  - includes six different FSM coding styles
  - includes multi-clock and FIFO design techniques
- Gain exposure to new SystemVerilog modeling capabilities
  - includes use of dynamic types and arrays for behavioral modeling
  - includes inclusion of C-models using the new SystemVerilog DPI
  - includes using proven techniques for generating self-checking tests

Course Overview

Sunburst Design - Advanced SystemVerilog for Design is a 3-day fast-paced intensive course focuses on proven and new features SystemVerilog for design, simulation and synthesis. Efficient and proven coding styles are combined with frequent exercises and insightful labs to demonstrate the capabilities of new SystemVerilog features. You will discover that SystemVerilog capabilities are fully backward compatible with Verilog-2001 designs.

This SystemVerilog training was developed and is frequently updated by the renowned SystemVerilog guru and IEEE SystemVerilog committee member, Cliff Cummings, who has presented at numerous SystemVerilog seminars and training classes world wide, including the 2003-2004 SystemVerilog NOW! Seminars and 2010 ModelSim SystemVerilog Assertion Based Verification Seminars.

The 1000+ page binder and 140+ page lab guide for this 3-day course covers all of the important SystemVerilog coding styles for RTL & behavioral design. These materials are constantly being updated with the latest clarifications and corrections passed by the IEEE SystemVerilog committee, of which Cliff is an active participant. Numerous proven usage guidelines are taught and explained.

Target Audience

Sunburst Design - Advanced SystemVerilog for Design is intended for design engineers who require in-depth knowledge on the IEEE SystemVerilog-2005 standard with an emphasis on the new RTL & behavioral design capabilities.

Prerequisites (mandatory)

This is a very advanced SystemVerilog class that assumes engineers already have a good working knowledge of the Verilog language.

This course assumes that students have a practical working knowledge of Verilog HDL or have completed Verilog HDL training. Engineers with VHDL synthesis experience and some Verilog exposure will do well in this class. Engineers with no prior HDL training or experience will struggle in this class. Engineers with weak Verilog knowledge or experience should consider adding the 1-day, *Sunburst Design - Accelerated Introduction to Verilog-2001 & Best Known Coding Practices* course to fully prepare for advanced SystemVerilog training.

The Sunburst Design - Advantage

Who is teaching your "expert" and "advanced" classes? Most companies will not tell you because their instructors might not have much design experience or may never have participated on any of the Verilog or SystemVerilog Standards groups or presented at industry recognized conferences. Go to our web site and read about the Sunburst Design - Instructors - they are the best and they have the experience and qualifications to offer best-in-class training.

Sunburst Design Courses:

- *Sunburst Design - Advanced SystemVerilog for Design & Verification* - 4-5 days
  - *Sunburst Design - Advanced SystemVerilog for Design* - 3 days
  - *Sunburst Design - Advanced SystemVerilog for Verification* - 3-4 days
- *Sunburst Design - Expert Verilog-2001 for Synthesis & Verification* - 4 days
  - *Sunburst Design - Expert Verilog-2001 & Coding for RTL Design & Synthesis* - 2 days
  - *Sunburst Design - Expert Verilog-2001 Design, RTL Synthesis & Verification Techniques* - 2 days
- *Sunburst Design - Comprehensive Verilog-2001 Design & Best Coding Practices* - 4 days
  - *Sunburst Design - Introduction to Verilog-2001 & Best Coding Practices* - 2 days
  - *Sunburst Design - Advanced Verilog-2001 Knowledge & Design Practices* - 2 days
  - *Sunburst Design - Accelerated Introduction to Verilog-2001 & Best Known Coding Practices* - 1 day

Course Customization? - Sunburst Design courses can be customized to include **your** company's coding guidelines or to modify the course for a different audience. Sections can be added or deleted from a course to meet you company's needs.

Course Syllabus

Day One

SystemVerilog Enhancements & Methodology Overview

- Includes a quick review of SystemVerilog resources available to design & verification engineers.

- Verilog & SystemVerilog Keywords
- SystemVerilog Books & Resources
- SystemVerilog Enhancements Strategy & High-Level Methodology

Data Types & Typedefs

- Includes data types, enumerated types, compilation units, packages, casting and randomization functions.

- Nets & Variables Fundamentals & Guidelines
- Blocking & Nonblocking Assignment Fundamentals & Guidelines

- SystemVerilog data types
- Enhanced literal numbers syntax
- Resolved & Unresolved types
- 4-state & 2-state types
- Typedefs
- Near-Universal types
- SystemVerilog type usage guidelines
- Enumerated types
- Struct data type intro
- Type parameters
- Intro to the SystemVerilog program construct
- $unit & $root
- Compilation units & separate compilation
- Packages & :: (package scope operator)
- SystemVerilog package strategies
- Strings
- Static & dynamic type-casting
- Random number generation: $random -vs- $urandom -vs- $urandom_range
- Simulation command aliases & switch definitions
- LABS: Multiple SystemVerilog types, typedefs, type-casting and logic labs

SystemVerilog Operators, Loops, Jumps. Intro to Logic-Specific Processes, Enhanced functions & tasks

- New always_type blocks help RTL designers. To help verification engineers understand design constructs, the always_type blocks are briefly introduced in this section. Enhancements to tasks and functions make them more useful and easier to use.

- New SystemVerilog operators
- Enhanced loops & jumping statements
- always_comb / always_latch / always_ff - (1-slide introduction only)
- always @* -vs- always_comb
- SystemVerilog enhancements to tasks & functions
- SystemVerilog priority & unique - modifiers for case- & if-statements
- `timescale directive
- SystemVerilog timeunit & timeprecision

Implicit .* and .name Port Instantiation

- Implicit port connections can reduce top-level ASIC and FPGA coding efforts by more than 70% and simultaneously enforce greater port type checking.

- Verilog-2001 positional & named ports
- SystemVerilog .* implicit ports
- SystemVerilog .name implicit ports
- Implicit port connection rules & comparisons - includes IEEE 1800 latest updates
- Strong port-type checking
- New debugging techniques - automatic expansion of .* ports
- Block-level testbenches with implicit ports
- Advantages & disadvantages
- LABS: implicit port instantiation labs
- LABS (optional) : SystemVerilog random numbers

Day Two

Nonblocking Assignments, Race Conditions & SystemVerilog Event Scheduling
- SystemVerilog is fully backward compatible with Verilog-2001 (it is also fully race backward compatible!) This section describes in detail how the new SystemVerilog event scheduling works and how it will reduce race conditions between RTL designs and verification suites.

- Verillog-2001 Event Scheduling
- 8 guidelines for RTL coding & nonblocking assignments
- SystemVerilog enhanced scheduling - includes IEEE 1800 latest updates
- Verilog -vs- SystemVerilog race conditions
- Scheduling of new SystemVerilog commands
- * Blocking & Nonblocking Assignment Details
- * Mixed RTL & Gate simulations

Structs, Unions, Packed & Unpacked Arrays
- Packed & unpacked arrays, unions and structs allow greater abstraction and more concise coding. The new dynamic array types facilitate behavioral modeling and assist in the development of verification environments.

- Structs & assignment patterns
- Packed & unpacked arrays
- Array indexing
- Structs & packed structs
- Unions & packed unions
- Dynamic arrays & methods
- foreach loop
- Associative arrays & methods
- Queues & concatenation operations
- Queue methods

Interfaces
- Interfaces are a powerful new form of abstraction and this section details how they work for design and verification. This section also discusses when and when not to use interfaces. Virtual interfaces are described after the introduction of virtual classes and virtual methods on day three.

- Interface usage overview
- Introduction to generic interfaces
- Interfaces -vs- records
- How interfaces work
- 4 requirements for good interface usage
- Interfaces - legal & illegal usage
- Interface constructs
- Interface modports
- Generic interfaces
- LABS: multiple interface and interface-protocol labs

DPI - Direct Programming Interface - SystemVerilog's C-Language Interface
(Optional section - may be omitted to give more time to other topics and labs)
- The Direct Programming Interface (DPI) can be used to simulate C-code with SystemVerilog code. This section describes how this can be done and how DPI programming differs from PLI programming.

- DPI layers
- function import
- function export
- task export
- Using SystemVerilog simulation timing in a C model
- DPI -vs- PLI example
- No PLI required
- How to compile and simulate C-code with SystemVerilog designs
- SystemVerilog & SystemC
- LAB: SystemVerilog using C-code functions

SVA - SystemVerilog Assertions
- This section details how the SystemVerilog Assertion (SVA) syntax works and how assertions can be used for design and verification. Special macro-techniques are shown to reduce assertion coding effort by up to 80%.

- What is an assertion? / Who should add assertions?
- Assertion benefits - bug detection efficiency
- SystemVerilog assertion types
- SystemVerilog immediate assertions
- SystemVerilog concurrent assertions
- Assert & cover properties & labels
- Properties and assert property
- Overlapping & non-overlapping implications
- Edge testing functions
- Sequences
- Vacuous success
- Property styles
- Reduced assertion coding effort using macros
- Macros with default arguments (SystemVerilog-2009 update)
- Assertion coding style efficiency benchmarks
- SystemVerilog assertion system functions
- Sampled value functions
- Assertion severity tasks
- Assertion and coverage example of an FSM design
- Binding SVA to an existing model
- Bind command details and guidelines
- LABS: SystemVerilog Assertions with synchronous FIFO design

Day Three

Logic Specific Processes, Unique & Priority - full_case & parallel_case
- The new always_type blocks show design intent and help ensure construction of proper hardware designs. The always_type blocks are discussed in detail in this section. This section also details how unique and priority are new SystemVerilog replacements for the dangerous "Evil Twins," full_case parallel_case.

- Logic specific processes (always_type blocks) document designer intent
- always_comb
- always_latch
- always_ff
- Added design checks using always_type blocks
- always @* -vs- always_comb
- void functions
- always_comb & void functions
- Combinational sensitivity
- Design encapsulation through void functions
- always_ff for DDR? (SystemVerilog-2009 enhancement)
- 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)
- Three examples using case modifiers
- * LABS: simple SystemVerilog combinational and sequential logic labs
- * Multiple small synthesis examples

SystemVerilog FSM Design Techniques
- Six 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
- One always block - avoid this
- Two always blocks - recommended
- Three always blocks - recommended
- Onehot case(1'b1) - recommended
- Onehot parameters - avoid this
- Output encoded - recommended
- Coding & synthesis efficiency

- Verilog-2001 FSM enhancements
- SystemVerilog FSM enhancements
- Advanced enumerated types
- LABS: SystemVerilog FSM design labs

Multi-clock Clock Domain Crossing (CDC) & FIFO Design Techniques using SystemVerilog

- Very advanced design techniques from Cliff's award-winning presentations on the efficient implementation of multi-clock CDC & 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).

- Metastability
- Multi-clock Clock Domain Crossing (CDC) design & synthesis strategies
- Multi-signal CDC techniques
- MTBF (Mean Time Before Failure)
- Syncing before passing multiple CDC signals
- Multiple CDC signals - consolidation
- Multiple CDC signals - synchronization
- Multiple CDC signals - Multi-Cycle Path (MCP) Formulation
- Synchronizing counters
- Gray codes
- Gray code counters
- CDC Design partitioning
- CDC simulation issues
- CDC gate-level simulation X-avoidance techniques
- Multi-clock FIFO design techniques from Cliff's award-winning presentations

Classroom Details

Training is generally conducted at your facilities. For maximum effectiveness, we recommend having one workstation or PC for every two students, with your preferred SystemVerilog simulator licenses (we often can help provide the simulator and temporary training licenses).

For more information, contact:

Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 503-641-8446