World Class SystemVerilog & UVM Training

Sunburst Design - SystemVerilog Fundamentals

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.

2 Days
70% Lecture, 30% Lab
Advanced Level

Course Objective

Introduce engineers to *world class* SystemVerilog language capabilities using award winning materials developed by renowned Verilog & SystemVerilog Guru, Cliff Cummings

Upon completion of this course, students will understand:

- SystemVerilog language fundamentals
  - includes new SystemVerilog data types and capabilities
  - includes new SystemVerilog RTL and abstraction capabilities
  - includes use of dynamic types and arrays for behavioral modeling
  - includes 6 different SystemVerilog FSM coding styles
  - optionally - includes inclusion of C-models using the new SystemVerilog DPI
  - includes using SystemVerilog Assertions (SVA) for design and verification
- Students will have a good understanding of SystemVerilog RTL design features.

Course Overview

Sunburst Design - SystemVerilog Fundamentals is a 2-day fast-paced intensive course that introduces new SystemVerilog features 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

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996

2003-2004 SystemVerilog NOW! Seminars and 2010 ModelSim SystemVerilog Assertion Based Verification Seminars.

Target Audience

Sunburst Design - SystemVerilog Fundamentals is intended for design & verification engineers who require an introduction to IEEE SystemVerilog capabilities.

Prerequisites (mandatory)

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.

Classroom Details

Training is generally conducted at customer facilities and is sometimes offered as an open-enrollment training class. For maximum effectiveness, it is recommended to have at least 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. - 801-960-1996

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 - and why you should avoid it.
- $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

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996

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

- 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. Enhancements to tasks and functions make them more useful and easier to use.

- New SystemVerilog operators
- Enhanced loops & jumping statements
- 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)
- SystemVerilog enhancements to tasks & functions
- `timescale directive
- SystemVerilog timeunit & timeprecision
- * LABS: simple SystemVerilog combinational and sequential logic labs

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 - auto-schematic generation
- Block-level testbenches with implicit ports
- Advantages & disadvantages
- LABS: implicit port instantiation labs
- LABS (optional) : SystemVerilog random numbers

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996

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 are used more in verification and have been moved to the UVM training course.

- Structs & assignment patterns
- Packed & unpacked arrays
- Array indexing
- Structs & packed structs
- Unions & packed unions

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

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996

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 in the UVM training class.

- 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
- LABS: multiple interface and interface-protocol labs

DPI - Direct Programming Interface - SystemVerilog's C-Language Interface

(Reference Material Only - Not Lectured unless requested) - 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

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996

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%. Other recommendations to reduce SVA coding errors are included in this section.

- 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

For more information, contact:
Cliff Cummings - cliffc@sunburst-design.com - Sunburst Design, Inc. - 801-960-1996