Sunburst Design - SystemVerilog Verification

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

2 Days
70% Lecture, 30% Lab
Prerequisites (mandatory) - ***This is an advanced course using SystemVerilog that assumes engineers have already an understanding or working knowledge of Verilog-HDL.***

Course Syllabus

(~10 minute breaks near the top of each hour)
(Lab time is scheduled for "Lunch & Lab" and again near the end of the day)

This course may be customized by client companies on a WebEx conference call with Cliff Cummings.

Day One

SystemVerilog Enhancements & Methodology Overview

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

Data Types & Typedefs

- 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

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

Includes materials from Cliff's SNUG 2016 & SNUG 1999 award-winning papers on these topics. Also includes materials from Cliff's SNUG 2005 paper on priority & unique.

- 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

Includes materials from Cliff's SNUG 2007 award-winning paper on implicit port enhancements.

- 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

Day Two

Nonblocking Assignments, Race Conditions & SystemVerilog Event Scheduling

Includes materials from Cliff's SNUG 2002 & SNUG 2000 award-winning papers these topics.
Also includes materials from Cliff's SNUG 2006 paper on SystemVerilog event regions.

- 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

SystemVerilog FSM Design Techniques

- Includes materials from Cliff's SNUG 2019, SNUG 2003, ICU 2002 & SNUG 2000 award-winning papers on these topics.

- 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

Structs, Unions, Packed & Unpacked Arrays

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

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

SVA - SystemVerilog Assertions

Includes materials from Cliff's SNUG 2016 & SNUG 2009 SVA papers. Both were voted Best Paper 1st Place at their respective conferences.

- 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