Verilog for Experienced VHDL Designers

Accelerated Overview of Verilog, RTL Synthesis, and Verification

3 days, Intermediate/Advanced Level

"The content was very comprehensive and the materials were well prepared."

Overview
The learning curve for Verilog is not as steep as for VHDL, but it is longer. As a less restrictive language, Verilog has both strengths and weaknesses. This workshop covers the essential syntax of the Verilog language, the corresponding constructs with VHDL and alerts the users to some potential pitfalls.

The workshop then accelerates into more advanced topics like coding to enhance speed and reduce area. Several larger case studies are included to highlight principles of coding for synthesis. Previews of SystemVerilog features are included, wherever significant future enhancements are anticipated (e.g. C-like structs to replace the need for faking records in Verilog).

This course draws material from the following courses:

- *Verilog Expanded - 4 days*
- *Advanced Verilog for Synthesis & Verification - 3 days*
- *Verilog Verification Methodology – 3 days*

Audience:
The course is customized for engineers who are experienced VHDL designers and familiar with Verilog but have not used Verilog for design.

Outline

Day 1 - Accelerated Overview and Synthesis

Unit 1. Overview

- HDL-based synthesis and simulation.
- Unifying CNTR8 case study.
- Concurrent process model in Verilog.
- RTL vs. behavioral coding.

Unit 2. Elements of Syntax

- Lexical elements.
- Bit vectors and literals, signed and unsigned.

- Data types.
- Declaration syntax.
- Operators.
- Expressions.

Unit 3. Combinational Logic

- Conditional constructs (if-else, case, casez, ?:).
- Insights into case (full, parallel, reverse).
- Iterative constructs (for, while).
- Disabling blocks of code.
- Large case study: Flash priority-encoder.

Unit 4. Sequential Logic

- Sequential building blocks.
- Registers with synch/asynch resets and clock enable.
- Parallel/serial converter.
- Using blocking vs. nonblocking assignments.
- Non-synthesizable constructs and workarounds.
- Large case study: Digital modulator for cell phone IC, with RTL Manchester encoder model.

Day 2 - Synthesis

Unit 5. Block Integration

- Chip-level netlisting of major modules.
- Instantiation.
- Multiple and arrayed instantiation.
- Instantiating I/O cells using generate loops.
- 2-D arrays.
- Large case study: Triangular array generation. Automatic functions and tasks. Constant functions; localparams.

Unit 6. FSM Coding

- Coding FSMs in Verilog.
- Baseline dual-process approach.
- Modified Mealy FSM with registered next-outputs.
- One-hot-encoded FSM.
- Hierarchical FSMs.
- Controller for cell-phone chip.
- Large case study: synthesizable CRC (clock recovery circuit).

Unit 7. Coding for Area

- Classic area/delay trade-off.
- Avoiding excess logic.
- Reducing ASIC gate count.

- Minimizing algebraic tree nodes.
- Sharing arithmetic resources.
- Sharing non-arithmetic logic like array indexing.
- Cacheing recomputed quantities.
- Scheduling over multiple clock cycles.

Unit 8. Coding for Speed

- Parallelizing operations.
- Minimizing algebraic tree height.
- Resource implementation selection.
- Exploiting concurrency.
- Large case study: array comparator. Accommodating late input arrivals.

Day 3 - Verification

Unit 9. Verilog Testbenches

- Verilog testbench architecture.
- Clock generation and common pitfalls.
- Timescales in Verilog.
- Simple stimulus generation.
- Sampling response at regular intervals or on change.
- Fork-join for specifying waveforms.

Unit 10. Applying Stimulus

- Hard-coded and random stimuli.
- Parameterized values.
- Input from files.
- Using $readmem.
- Verilog 2001 file I/O enhancements.

Unit 11. Monitoring Response

- Verilog system tasks.
- Response checkers
- VCD files
- Large case study: Booth multiplier with self-checking testbench.

Unit 12. Special Topics

- Records in Verilog
- Named events

For additional information, contact: Tom Wille, 503-656-4457, tw@tm-associates.com