

## **Advanced VHDL Coding Styles** **for Synthesis & Verification**

### **3 days, Advanced Level**

*“My time was well spent. Lots of information, no time to slack off.”*

#### **Overview**

Staying competitive in today's ASIC/FPGA market means designing ICs with greater functionality, higher speed, and lower cost. Effective VHDL coding techniques can make all the difference between designs that meet tough synthesis targets and verification schedules, versus those requiring re-spins. The goal of this hands-on workshop is to enhance your mastery of VHDL language features which drive the synthesis tool and maximize verification productivity. Most of the material is tool-neutral, although case studies include results for popular ASIC/FPGA tools. Each key principle or coding insight is shown in the context of a realistic design situation. You'll be exposed to a wireless-telephony chip design, both in lectures and labs. Digital video and graphics applications are also explored.

This course can be customized or condensed to meet the specific needs of your design team or available schedule.

#### **Benefits**

Upon completion of this course, you will:

- Understand and apply VHDL coding techniques which synthesize efficiently, while avoiding common pitfalls.
- Be able to visualize the synthesized logic which results from specific VHDL constructs and coding styles.
- Gain experience in coding with a specific architecture in mind, which implements the required logic function while meeting speed or area targets -- through parallelization, resource sharing, and other HDL coding techniques.
- Be able to tap into the full diversity of simulatable VHDL constructs for verification, following practical IC verification strategies and writing more effective testbenches.

#### **Intended Audience**

This course is recommended for IC designers already familiar with basic VHDL syntax, but who want to use the features of VHDL more effectively to synthesize blocks that meet specific timing and area targets. The third day of the workshop focuses on VHDL coding for

testbenches, and is positioned for designers who need to understand verification issues, as well as verification engineers who want a firmer grasp of language subtleties.

### Prerequisites

Students need to have a working knowledge of basic VHDL.

### Training Approach

This is an intensive, interactive course, which is approximately 50% lecture and 50% lab. Questions are highly encouraged.

### Course Outline

#### Day 1

Unit 1: Introduction Coding and synthesizing a typical VHDL design unit to be used in the wireless chip.

- Synthesis-friendly features of VHDL
- Migrating the module from an FPGA prototyping technology to a submicron ASIC technology
- Wireless chip design spec

Unit 2: Combinational Logic

- Effective use of conditional constructs (if-else, case, when)
- Decoding. Priority encoding
- Code conversion. ROM usage
- Multiplexing/demultiplexing
- Iterative constructs (for, while)
- Signed/unsigned arithmetic
- Using concurrency

Unit 3: Sequential Logic

- Sequential building blocks
- Registers with synch/asynch reset and clock enable
- Parallel/serial converter
- Ring counter
- Edge detector
- Signals vs. Variables
- Non-synthesizable constructs and workarounds
- ASM (algorithmic state machine) charts as an aid to sequential-machine design

Unit 4: Block Integration

- Chip-level design and integration issues
- Coding above the block level
- Multiple clock domains
- Partitioning an entire chip into modules
- Separating blocks with different design goals
- Registering outputs
- Maximizing optimization
- Instantiating IP blocks such as Synopsys DesignWare
- Instantiating I/O cells using generate loops

#### Day 2

Unit 5: FSMs and Controllers

- Coding FSM-oriented designs
- ASM (algorithmic state machine) chart usage
- Mealy vs. Moore insights
- Modified Mealy FSM with registered next-outputs
- Hierarchical FSMs
- Controller for wireless chip
- Datapath/controller paradigm

Unit 6: Getting the most out of your tools

- Synthesizable HDL subset
- Unsupported constructs
- Excluding simulator-oriented code
- Using generics
- Name-based redefinition

- Text substitution
- Managing code modules
- Using package files
- Coding for reuse and portability

#### Unit 7: Coding for Area

- Classic area/delay trade-off
- Avoiding excess logic
- Reducing ASIC gate count or FPGA LUT usage
- 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 Performance

- Parallelizing operations
- Minimizing algebraic tree height
- Resource implementation selection
- Exploiting concurrency
- Accommodating late input arrivals

### Day 3

#### Unit 9: Verification

- Verification definition, methodology
- Testbench architecture
- Clock generation
- Simulator resolution

- Stimulus generation
- Sampling response at regular intervals or on change
- Comparison of responses

#### Unit 10: Testbench Techniques

- Encapsulating tests
- Self-checking testbenches
- File-oriented testbenches
- Fixed vectors
- Bus functional models
- Synchronizing stimuli
- Named events

#### Unit 11: Avoiding Simulation Pitfalls

- Weaknesses of VHDL
- Avoiding race conditions during simulation
- Avoiding simulation-synthesis mismatches
- Avoiding simulator bottlenecks and improving performance

#### Unit 12: Advanced Topics

- Bottom-up vs. top-down verification methodology
- Emergence of static verification
- Coding guidelines for formal equivalence
- Scan-based testing
- DFT guidelines
- Future directions.

### All Days, Labs 1—12

You will apply coding techniques covered in lecture to synthesize and verify, block by block, a wireless telephony chip that meets functional specs and achieves target clock frequency.

### For more information, contact:

Tom Wille

[tw@tm-associates.com](mailto:tw@tm-associates.com)

503-656-4457

TM Associates, Inc.

[www.tm-associates.com](http://www.tm-associates.com)