Verilog Verification Methodology

3 days, Intermediate Level

"Good mix of lecture and lab; lecture and presentation very well done."

Overview
Verification now takes 70% of the design cycle. To stay competitive in today's ASIC/FPGA market, you need to use the latest techniques to verify ICs. Creating a reusable test environment will allow you to create test cases and checkers quickly, and help you meet your project deadline. The goal of this hands-on class is to enhance your mastery of Verilog language, and use it to verify today's complex designs. Most of the material is tool-neutral, and verification techniques are limited in scope to the Verilog language itself-- no extra tools or languages to learn! 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 Verilog coding techniques to verify a design efficiently, while avoiding common pitfalls.
- Create reusable verification IP, including bus functional models, checkers, and messengers.
- Use cutting edge techniques to create a verification plan, based on the functional specification.

Intended Audience
This course is recommended for verification engineers already familiar with Verilog syntax, but who want to learn new verification methodologies.

Prerequisites
Students need to have a working knowledge of Verilog.

Suggested Follow on Course

Advanced Verilog for Synthesis & Verification

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 to Functional Verification

- Verification Overview
  - Terminology
  - Different Coverage Metrics
  - Verification tools
- Verification plans
  - What information does a Verification Plan contain?
  - How to create a Verification Plan
- BEC100 Packet Router overview

Lab 1 Verification Plan

Unit 2: Verification Environment and Hard-Coded Stimulus

- Differences between RTL and behavioral code
- Module versus system level verification
- Create a test bench
- Hard code a test case
  - Asynchronous transactions
  - Synchronous transactions

Lab 2 Testbench and test case

Unit 3: Asynchronous Bus Functional Models (BFMs)

- Asynchronous BFMs
  - What are BFMs?
  - Creating task-based BFMs
  - Creating state machine-based BFMs
- Instantiating BFMs
  - Hard coding
  - 'include
  - Instantiating a module

Lab 3 Asynchronous BFMs

Day 2

Unit 4: Synchronous Bus Functional Models (BFMs)

- Synchronous BFMs
- Creating task-based BFMs
- Creating state machine based BFMs
- Avoiding common problems with BFMs

Lab 4 Synchronous BFMs

Unit 5: Applying Stimulus

- Stimulus
- Hard-coded values
- Random values
  - Distributions
- Passing values

Lab 5A Applying random data

- Reading from a file
- Passing Values

Lab 5B Reading stimulus from a file

Unit 6: Errors, Irritators, and Records

- Introducing Errors
- Conditionally compiled code
- Randomly inserting errors
- DUV irritators
- DUV monitors
- Faking records in Verilog

Lab 6 Using Verilog Records

Day 3

Unit 7 Messaging and Timing Checkers

- Displaying messages
  - Consistent messaging
- Termination
- Timing Checks
- Setup, hold, and width checks

Lab 7 Checks and messages

Unit 8 Checkers and Scoreboards

- Checkers
  - Explicit checkers
  - Continuous checkers
- Scoreboarding
- Post simulation checks
  - Writing to files
- Verification environment and gate-level netlists

Lab 8 Scoreboard and checkers

All Days

Labs
You will apply coding techniques covered in lecture to create a reusable verification environment for a packet router.

For more information contact:

Tom Wille
tw@tm-associates.com
503-656-4457
TM Associates, Inc.
www.tm-associates.com