Content Owner/Lead Developer: Ameesh Oza
Workshop Title: SystemVerilog Testbench
Workshop Version: 2019.06
Number of days: 3
This is: **___ a. A NEW course** **_x_ b. Replacement Information for an Existing Course** **Delivery Method:** **_x_ a. Instructor Led ___ b. Virtual Classroom**

MINI-SUMMARY

A hands-on introduction to the SystemVerilog language to verify a device under test using VCS with object-oriented methodologies targeting coverage-driven constrained-random test environments.

SUMMARY

This practical hands-on workshop introduces students to the SystemVerilog language to verify a device under test using VCS. After finishing this workshop, students will be able to write, use, compile and debug SystemVerilog testbenches. These testbenches will use object-oriented programming based verification methodologies to develop coverage-driven constrained-random test environments. The students will be able to understand and use methodologies like UVM/VMM/OVM.
This workshop is geared towards Verification and Design Engineers.

OVERVIEW

In this intensive, three-day workshop, you will learn the key features and benefits of the SystemVerilog testbench language and its use in VCS. At the end of this workshop, you should have the skills required to write an object-oriented SystemVerilog testbench to verify a device under test with coverage-driven constrained-random stimulus using VCS.

You will learn how to develop an interface between the SystemVerilog test program and the Device Under Test (DUT). The workshop will explain how the intuitive object-oriented technology in SystemVerilog testbench can simplify verification problems. Randomization of data is covered to show how different scenarios for testing may be created. This workshop concludes with an in-depth discussion of functional coverage including a uniform, measurable definition of functionality and the SystemVerilog constructs that allow you to assess the percentage of functionality covered, both dynamically and through the use of generated reports.

A lab on packages and methodology will introduce you to the basic concepts used in methodologies such as UVM and VMM. To reinforce the lecture, and accelerate mastery of the material, you will complete a challenging test suite for a real-world system-based design.

This workshop does not cover basic Verilog or VHDL concepts like modules/entities, initial and always blocks, processes etc.

OBJECTIVES

At the end of this workshop you should be able to:

- Build a SystemVerilog verification environment
- Define testbench components using object-oriented programing
- Develop a stimulus generator to create constrained random test stimulus
- Develop device driver routines to drive DUT input with stimulus from generator
- Develop device monitor routines to sample DUT output
- Develop self-check routines to verify correctness of DUT output
- Abstract DUT stimulus as data objects
- Execute device drivers, monitors and self-checking routines concurrently
- Communicate among concurrent routines using events, semaphores and mailboxes
- Develop functional coverage to measure completeness of test
- Use SystemVerilog Packages

AUDIENCE PROFILE

Design or Verification engineers who write SystemVerilog testbenches at the block or chip level

PREREQUISITES

To benefit the most from the material presented in this workshop, you should have:

- A basic understanding of digital IC design
- Familiarity with UNIX workstations running X-windows
- Familiarity with vi, emacs, or other UNIX text editors
- A basic understanding of Verilog and/or VHDL and their use in simulations.

COURSE OUTLINE

Day 1

- The Device Under Test
- SystemVerilog Verification Environment
- SystemVerilog Testbench Language Basics - 1
- SystemVerilog Testbench Language Basics - 2

Day 2

- Managing Concurrency in SystemVerilog
- Object Oriented Programming: Encapsulation
- Object Oriented Programming: Randomization

Day 3

- Object Oriented Programming: Inheritance
- Inter-Thread Communications
- Functional Coverage
- SystemVerilog UVM preview

SYNOPSYS TOOLS USED

- VCS 2019.06
- Verdi 2019.06