SystemVerilog for Verification

Course Description

SystemVerilog is a significant new enhancement to Verilog and includes major extensions into abstract design, test-bench, formal, and C-based APIs.

SystemVerilog also defines new layers in the Verilog simulation strata. These extensions provide significant new capabilities to the designer, verification engineer and architect, allowing better teamwork and co-ordination between different project members.

This course provides all necessary theoretical and practical know-how to write test-benches using SystemVerilog standard language.

The course goes into great depth, and touches upon every aspect of the standard with directly connected to the topics needed in the industry today.

The course combines 50% theory with 50% practical work in every meeting.

The practical labs cover all the theory and also include practical test-bench design.

The course also teaches how to write test-bench programs and employ a simulation and tools, how to build coverage-driven test-bench, use of object-oriented programming methods, use of classes, functional coverage and randomization techniques.

Course Duration

4 days

Goals

1. Become familiar with the verification and testing methodology
2. Use SystemVerilog declaration spaces
3. Connect test-bench program to the design
4. Use OOP programming
5. Build test-bench programs with randomization
6. Use threads and inter-process communication

7. Use of functional coverage
8. Become familiar with assertions
9. Become familiar with SystemVerilog Interface with C language

Intended Users

Hardware or software engineers who would like to design test-bench and employ verification techniques with SystemVerilog

Previous Knowledge

A basic background in digital logic, Verilog

Course Material

1. Simulator: Modelsim
2. Questa (MentorGraphics)

Table of Contents

Day #1

- **SystemVerilog Procedural Statements & Routines**
  - Procedural statements
  - Tasks, functions, and void functions
  - Task and function overview
  - Routine arguments
  - Returning from a routine
  - Local data storage
  - Time values

- **Connecting the Testbench and Design**
  - Separating the testbench and design
  - The interface construct
  - Stimulus timing
  - Interface driving and sampling
  - Connecting it all together
  - Top-level scope
  - Program-module interactions
  - SystemVerilog assertions
  - The ref port direction
  - The end of simulation

Day #2

- **Basic OOP**
  - Introduction to OOP
  - Where to define a class
  - OOP terminology
  - Creating new objects
  - Object deallocation
  - Using objects
  - Static variables versus global variables
  - Class methods
  - Defining methods outside of the class
  - Scoping rules

- Using one class inside another
- Understanding dynamic objects
- Copying objects
- Public versus local
- Straying off course
- Building a testbench

- **Randomization**
  - Introduction to randomization
  - What to randomize
  - Randomization in SystemVerilog
  - Constraint details
  - Solution probabilities
  - Controlling multiple constrained blocks
  - Valid constraints
  - In-line constraints
  - The pre_randomize and post_randomize functions
  - Random number functions
  - Constraints tips and techniques
  - Common randomization problems
  - Iterative and array constraints
  - Atomic stimulus generation versus scenario generation
  - Random control
  - Random number generators
  - Random device configuration

Day #3

- **Threads and Interprocess Communication**
  - Working with threads
  - Disabling threads
  - Interprocess communication
  - Events
  - Semaphores
  - Mailboxes
  - Building a testbench with threads and IPC

- **Advanced OOP and Testbench Guidelines**
  - Introduction to inheritance
  - Downcasting and virtual methods
  - Composition, inheritance, and alternatives
  - Copying an object
  - Abstract classes and pure virtual methods
  - Callbacks
  - Parameterized classes
  - Polymorphism

- **Functional Coverage**
  - Coverage types
  - Functional coverage strategies
  - Anatomy of a cover group
  - Triggering a cover group
  - Data sampling
  - Cross coverage
  - Generic cover groups
  - Coverage options
  - Analyzing coverage data
  - Measuring coverage statistics during simulation

Day #4

- **Advanced Interfaces**
  - Interface concept
  - Virtual interfaces
  - Connecting to multiple design configurations
  - Procedural code in an interface

- **Interfacing with C**
  - Passing simple values
  - Connecting to a simple C routine
  - Connecting to C++

- Simple array sharing
- Open arrays
- Sharing composite types
- Pure and context imported methods
- Communication from C to SystemVerilog
- Connecting other languages

- **Assertions**
  - Specifying assertions
  - Assertions on internal DUT signals
  - Assertions on external interfaces
  - Assertions coding guidelines
  - Reusable assertions-based checkers
    - Simple checkers
    - Assertion-based verification IP
    - Architecture of assertion-based IP
    - Documentation and release items
  - Qualification of assertions