SystemVerilog for Design and Verification

Course Description
This comprehensive 4-day hands-on intensive course provides complete and integrated training program. It provides the participants with a deep knowledge of 1800-2005 SystemVerilog. The goal of this course is to fulfill the needs and requirements of engineers wanting to exploit the breadth of SystemVerilog features for both design and verification.

Level: Intermediate to Advanced

Training Duration: 4 days

Who Should Attend?
Experienced Verilog design and verification engineers wanting to use System Verilog 1800-2005 features for modeling, synthesis and verification of digital designs.

Prerequisites:
Digital design knowledge, Verilog 1364-1995 and Verilog 1364-2001

Skills Gained: After completing this training, you will be able to:
- Use System Verilog design advanced techniques
- Create libraries and configurations
- Build abstract models for verification of digital designs, using class-based object oriented constructs
- Create random, coverage driven simulation environment

Course Outline
- Verilog short history brief
- SystemVerilog data types
- User-defined types, structs and unions
- Procedural statements and flow control
- Arrays and lists
- Interfaces and Clocking Blocks
- SystemVerilog new operators
- Packages and configuration libraries
- Programs and module binding
- Transaction Level Modeling (TLM)
- Object oriented modeling - structs and classes
- Random and constrained simulation
- Direct Programming Interface (DPI)
- Assertions
- Coverage

Topics breakdown
- Verilog Evolution
- Data Types
- Strings
- Enumerated Data Types
- User-defined types
- Structs and Unions
- Arrays
  - Arrays breakdown as a sample
    - Packed Arrays
    - Unpacked Arrays
    - Array Querying System Functions
    - Dynamic array
    - Dynamic Arrays’ Pre-defined Methods
      - Arrays Assignment
    - Arrays Assignments Examples
    - Associative Arrays
    - Associated Arrays Pre-defined Methods
    - Array Assignment Patterns
    - Queues
- New Operators
- Unique and priority decision statements
- Functions
- Procedural Statements and Flow Control
- Fork-join Enhancements
- Instances and Port Connection Enhancements
- Programs
- Clocking Blocks
- Bind Operator
- Packages
- Object Oriented Modeling
- Random Simulation
- Random Sequence
- Assertions
  - Assertions breakdown as a sample
    - Immediate Assertions
    - Concurrent Assertions
    - Assertions Structure
    - Boolean Expression
    - Sequence
      - Sequence – Time Range Operators
      - Sequence Declaration
      - Sequences as Containers
      - Consecutive Repetitions in Sequences
      - Goto Repetition in Sequences
      - Non-consecutive Repetition in Sequences
        - Sampled Value Functions
      - The AND Operation
      - The Intersect Operation
      - The AND and INTERSECT examples
      - OR Operation
      - first_match operator
      - Throughout Operator
        - Throughout Example
      - Sequence Contained Within Another Sequence
      - ended Sequence Method
        - ended Sequence Method Example
      - triggered Sequence Method
      - Sequence Operators Summary
      - Manipulating Data in Sequences
      - Calling Sub-Routines on Sequence Match
      - System Tasks and Functions
    - Properties
      - Properties Kinds
      - disable if Within Properties
      - Property’s Arguments

Continued…
SystemVerilog for Design and Verification

...Continued

- Implications Within Properties
  - Implication Example
  - Properties Examples
- Multi-clock support
- Concurrent Assertions
- Assert Statement
- Assume Statement
- Cover Statement
- Clock Resolution
- Properties Declaration Technique
- Expect Statement
- Coverage

Lab Descriptions
- Manipulating packed, unpacked, static and dynamic arrays and lists
- Writing short design and test-bench using interfaces
- Writing SystemVerilog packages and library configurations
- Write short design and test-bench using programs and bind techniques
- Write SystemVerilog programs using abstract object oriented modeling
- Create program using random and constrained variables
- Writing assertions and debugging short design
- Testing coverage of existing small verification environment