



**OREN HANDS ON TRAINING & DEVELOPMENT LTD.**  
20 YAIR ROZENBLUM ST. KFAR SABA 4464601 ISRAEL

## SystemVerilog for Verification

### Course Description

This 5-days course designed for ASIC & FPGA verification engineers that would like to use the SystemVerilog and UVM to verify digital designs.

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

5 days

### Goals

1. Become familiar with SystemVerilog language
2. Use SystemVerilog user defines & Enumerated types
3. Use SystemVerilog arrays, structures and Unions
4. Become familiar with SystemVerilog procedural blocks
5. Become familiar with SystemVerilog interfaces
6. Become familiar with the verification and testing methodology



When innovation meets expertise...



**OREN HANDS ON TRAINING & DEVELOPMENT LTD.**  
20 YAIR ROZENBLUM ST. KFAR SABA 4464601 ISRAEL

---

7. Use SystemVerilog declaration spaces
8. Connect test-bench program to the design
9. Use OOP programming
10. Build test-bench programs with randomization
11. Use threads and inter-process communication
12. Use of functional coverage
13. Become familiar with assertions

## **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. Course book
2. Virtual Machine with labs handbook
3. Simulator: Questa (Siemens) or Xcelium (Cadence)



When innovation meets expertise...

## Table of Contents

### Day #1

- **Introduction to SystemVerilog**
  - SystemVerilog history and revisions
  - Key SystemVerilog enhancements for verification design
  
- **SystemVerilog New Declarations**
  - Package
  - \$unit
  - Simulation time units and precision
  
- **SystemVerilog Literal Values & Built-in Data Types**
  - Enhanced literal value assignments
  - `define enhancements
  - Variables
    - Object types and data types
    - 4-state variables
    - 2-state variables
    - Explicit and implicit variable and net data types
    - Synthesis guidelines
  - Using 2-state types in RTL models
    - 2-state type characteristics
    - 2-state types versus 2-state simulation
    - Using 2-state types with case statements
  - Relaxation of type rules
  - Signed and unsigned modifiers
  - Static and automatic variables
    - Static and automatic variable initialization
    - Synthesis guidelines for automatic variables
    - Guidelines for using static and automatic variables
  - Deterministic variable initialization
    - Initialization determinism



When innovation meets expertise...

- Initializing sequential logic asynchronous inputs
- Type casting
  - Static casting
  - Dynamic casting
  - Synthesis guidelines
- Constants
- **SystemVerilog Procedural Blocks & Statements**
  - Verilog general purpose always procedural block
  - SystemVerilog specialized procedural blocks
  - New operators
    - Increment and decrement operators
    - Assignment operators
    - Equality operators with don't care wildcards
    - Set membership operator (inside)
  - Operand enhancements
    - Operations on 2-state and 4-state types
    - Type casting
    - Size casting
    - Sign casting
  - Enhanced for loops
    - Local variables within for loop declarations
    - Multiple for loop assignments
    - Hierarchically referencing variables declared in for loops
    - Synthesis guidelines
  - Bottom testing do...while loop
  - The foreach array looping construct
  - New jump statements
    - The continue statement
    - The break statement
    - The return statement
    - Synthesis guidelines
  - Enhanced block names
  - Statement labels
  - Enhanced case statements

- Unique case decisions
  - Priority case statements
  - Unique and priority versus parallel\_case and full\_case
  - Enhanced if...else decisions
    - Unique if...else decisions
    - Priority if decisions
  - Blocking and non-blocking assignments
- **Hands-On Labs**

## Day #2

- **SystemVerilog Enumerated Types**
  - Creating new types with typedef
  - Enumerated types
    - Enumerated type label sequences
    - Enumerated type label scope
    - Enumerated type values
    - Base type of enumerated types
    - Typed and anonymous enumerations
    - Strong typing on enumerated type operations
    - Casting expressions to enumerated types
    - Special system task and methods for enumerated types
  - Printing enumerated types
- **SystemVerilog Arrays, Structures, Unions & Queues**
  - Structures
    - Structures declarations
    - Assigning values to structures
    - Packed and unpacked structures
    - Passing structures through ports
    - Passing structures as arguments to tasks and functions
    - Synthesis guidelines



When innovation meets expertise...



**OREN HANDS ON TRAINING & DEVELOPMENT LTD.**  
20 YAIR ROZENBLUM ST. KFAR SABA 4464601 ISRAEL

- Unions
  - Unpacked unions
  - Tagged unions
  - Packed unions
  - Synthesis guidelines
- Arrays
  - Unpacked arrays
  - Packed arrays
  - Using packed & unpacked arrays
  - Initializing arrays at declaration
  - Assigning values to arrays
  - Copying arrays
  - Copying arrays and structures using bit-stream casting
  - Arrays of arrays
  - Using user-defined types with arrays
  - Passing arrays through ports and to tasks and functions
  - Arrays of structures and unions
  - Arrays in structures and unions
  - Synthesis guidelines
- The foreach array looping construct
- Array querying system functions
- The \$bits “sizeof” system function
- Dynamic arrays, associative arrays, sparse arrays and strings
- **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
- **Hands-On Labs**



When innovation meets expertise...

## Day #3

- **Verification Guidelines**
  - The verification process
  - The verification methodology manual
  - Basic testbench functionality
  - Directed testing
  - Methodology basics
  - Constrained-random stimulus
  - What should you randomize?
  - Functional coverage
  - Testbench components
  - Layered testbench
  - Building a layered testbench
  - Simulation environment phases
  - Maximum code reuse
  - Testbench performance
  
- **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
- **Hands-On Labs**

## Day #4

- **Threads and Interprocess Communication**
  - Working with threads
  - Disabling threads
  - Interprocess communication
  - Events
  - Semaphores
  - Mailboxes



**OREN HANDS ON TRAINING & DEVELOPMENT LTD.**  
20 YAIR ROZENBLUM ST. KFAR SABA 4464601 ISRAEL

---

- 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
- **Hands-On Labs**



When innovation meets expertise...

## Day #5

- **Advanced Interfaces**

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

- **Assertions**

- Why Assertions? What are the advantages?
- What type of assertions should I add?
- How do I know I have enough assertions?
- Assertion types
- Concurrent 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

- **File I/O**

- How to open and close a file?
- How to open in read and append modes?
- How to read and write to a file?
- How to read until end of file?
- How to parse a line for values?

- **Hands-On Labs**



When innovation meets expertise...