Suggested Teaching Guidelines for

System Verilog PG-DVLSI March 2024

Duration: 40 class room hours + 50 Lab hours

Objective: To introduce Design and Verification Concepts, and implementing them using System Verilog.

Prerequisites: Knowledge of Verilog and Object-Oriented Programming (OOP)

List of Books / Other training material

Evaluation method: CCEE Theory exam– 40% weightage
Lab exam – 40% weightage
Internal exam – 20% weightage

Courseware:

No specific courseware for modules, faculty may share some course materials.

Reference:

1. System Verilog for Design by Sutherland, Davidmann and Flake / Springer
2. System Verilog for Verification by Chris Spear / Springer
3. Writing Testbenches using System Verilog by Janick Bergeron
4. System Verilog Language Reference Manual 3.1a

Session 1 & 2
Lecture: Introduction to System Verilog

- System Verilog: Origins, Overview, Need and Importance
- System Verilog Declaration Spaces
- Data types
  - 4-state and 2-state data type
  - String
  - Event
  - Type casting
  - User defined date types (typedef)
  - Enumerated data types (enum)
- Data Declaration
- Operators and Expression

Assignment – Lab:

Review Exercises – Handling variables, types, basic operations in System Verilog (SV)

Session 3 & 4
Lecture: Introduction to System Verilog

- Arrays
  - Packed and unpacked arrays
  - Array initialization at declaration
  - Multiple dimensions
  - Array Handling
  - Dynamic Array
  - Associative Array
  - Synthesis guidelines
- Overview of Structures

  - Declarations, assigning values to structures
  - Types: Packed and unpacked structures
  - Passing structures through ports, Passing structures as arguments to tasks and functions
  - Synthesis guidelines
- Overview of Unions
  - Types: Unpacked unions, Tagged unions, Packed unions
  - Synthesis guidelines
- An example of using structures and unions

Assignment – Lab:

Review Exercises – Array declaration, Array Handling exercises

Session 5

Lecture: Procedural Blocks and Statements

- Procedural Statements
  - Initial, final and always block
  - Loop
- Process
  - Combinational
  - Latched and sequential
  - Continuous assignment
  - Fork-join and its variant
- Clocking Domains

Assignment – Lab:

Review Exercises – Using procedural statements in SV coding

Session 6

Lecture: Tasks and Functions

- Introduction to Task and function
- Scope and life time of Task and function
- Argument passing in Task and function
  - Pass by value
  - Pass by reference
  - Default argument value
  - Argument passing by name
  - Optional argument list

Assignment – Lab:

Review Exercises – Lab assignments of SV coding using tasks and functions

Session 7 & 8

Lecture: Case Studies – Design

- Design of combinational circuit, sequential circuit and Finite State Machine using System Verilog.
- Case study of Traffic Light Controller / ATM Design / AHB Peripheral using System Verilog

Session 9
Lecture: Review of OOP Concepts

- OOP terminology
- Data hiding and encapsulation
- Introduction to inheritance
- Introduction to polymorphism
  - Compile time polymorphism
  - Run time polymorphism
- Introduction to Class
- Objects and properties
- Object methods
- Constructor
- Static Properties and methods
- This
- Assignment and re-naming and copying

Assignment – Lab:

- Review Exercises – OOPs practice codes in C++/SV

Session 10
Lecture: OOP in System Verilog

- Inheritance in System Verilog
- Overridden member
- Constant properties
- Abstract class and virtual methods
- Polymorphism
- Scope resolution operator
- Difference between classes, structures and unions
- Memory management

Assignment – Lab:

Review Exercises – Practice codes for using OOPs in SV.
Note: More lab practice should be taken for this session.

Session 11
Lecture: Introduction to Verification

- The Verification Process
- Importance of Verification
- Types of Verification
  - Formal Verification
  - Equivalence Checking
  - Model Checking
  - Functional Verification
- Functional Verification approach
  - Black-box
  - White –box
  - Grey-box
- Comparison between testing and Verification
- Verification reuse
  - Costing behind the verification

Session 12
Lecture: Coverage

- Understanding Coverage
- Methodology
- Code Coverage
  - Statement Coverage
  - Path Coverage
  - Expression Coverage
  - FSM Coverage
- Functional Coverage
  - Item Coverage
  - Cross Coverage
  - Transition Coverage
- Case Study

Assignment – Lab:

Review Exercises - some lab assignments which taking care of code coverage

Session 13
Lecture: Randomization - I

- Introduction
- Randomization in real world
- Randomization in System Verilog
- Random variables
- Constraint blocks
  - Inside
  - Weighted distribution
  - 'dist'
  - Array constraints
  - Conditional constraints
  - Bidirectional constraints

Assignments:

Review Exercises –

- Create a System Verilog program class to prove that variables declared with rand keyword are standard random variables.
- Create and run a SV project to check status of randomization return value.
- Write SV program in which if an extended object is casted to base object, all the constraints in extended object are solved along with the constraints in base object.
- Write SV program in array of objects are created before calling randomize.
- Write a SV program to implicit the weighted distribution in constraint expression.

-

Session 14
Lecture: Randomization - II

- Controlling multiple randomization
  - Solve-before
  - Constraint_mode()
  - If-else constraining
- Pre and post randomize functions
- Random Number function
- Tips and techniques of handling the randomization

Assignments:

Review Exercises –

- Create a project to add a new process without disturbing the random number generator state.
- Write a SV program on pre_randomize and post_randomize methods that can be used as hooks for the user to perform operations such as setting initial values and performing functions after assigning random variables.
- Create a project in which the constraint blocks are defined externally in the same file or other files.
- Specification: Create a test_1 file which requires Var between 0 to 100 and tets_2 file requires Var between 50 to 100.
- Declare the constraint block as empty and define them in other files.
- Write a SV program to randomize the class instances using global constraints.
- A simple testbench may use a data class with just a few constraints. Write SV program on how to have two tests with very different flavors of data.
- Create a project which implies that the constraints can be on or off with Constraint mode.

Session 15 & 16
Lecture: Case Studies – Randomization

- In general, randomization does not depend on life time of variable.
- Even if a variable is static, randomization is specfic to object.
- So rand_mode() on static variable, only switches off the randomization on the variable of that object.
- Implement it with the following:
  - Initialize Var1 as static and Var2 as automatic.
  - Var1 and Var2 in obj_2 should be nonrandomized
  - Var1 and Var2 in obj_1 are getting randomized.
  - Show that the only difference between Var1 and Var2 is that new random value for Var1 is appears on both objects.

Session 17 & 18
Lecture: Case Studies – Verification

- Case studies of Verification Environments in SystemVerilog for combinational designs, sequential designs and finite state machines

Session 19
Lecture: Assertions Property

- Definition and concept
- Implication
- Immediate assertions
- Concurrent assertions

Session 20
Lecture: Assertions timing

- Assertion sampling
- Multiple clock support
- Clock resolution