Suggested Teaching Guidelines for

Verification using UVM PG-DVLSI March 2024

Duration: 30 class room hours + 60 Lab hours

Objective: To introduce Verification Concepts and implementing them using UVM.

Prerequisites: Knowledge of System Verilog concepts, C programming, OOP concepts.

List of Books / Other training material

Evaluation method: 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. Universal Verification Methodology (UVM) 1.2 User's Guide
2. Universal Verification Methodology (UVM) 1.2 Class Reference

Session 1
Lecture: Introduction to Verification Methodologies

- Need of Verification Methodologies
- Comparative study of Verification Methodologies
  - Verification Methodology Manual (VMM)
  - Open Verification Methodology (OVM)
  - Universal Verification Methodology (UVM)

Session 2
Lecture: Overview of UVM

- Introduction to Universal Verification Methodology (UVM)
- Coverage-Driven Verification (CDV)
- UVM testbenches and environments
- Verification components:
  - Data Item (Transaction)
  - Driver (BFM)
  - Sequencer
  - Monitor
  - Agent
  - Environment
- UVM Class library
  - UVM object class
- UVM factory

Session 3 & 4
Lecture: Transaction Level Modeling (TLM)

- Basics of TLM
- TLM Overview, TLM, TLM-1, TLM 2.0
- UVM reporting & transaction
- Basic TLM communication
- Communicating between processes
- Blocking versus Nonblocking
- Connecting Transaction-Level Components
- Peer-to-Peer connections
- Port/Export Compatibility

Session 5
Lecture: TLM-1

- Encapsulation and Hierarchy
  - Hierarchical Connections
  - Connection Types
- Analysis Communication
  - Analysis Ports
  - Analysis Exports

Session 6
Lecture: Introduction to TLM 2.0 Theory Concepts

- Generic Payload
  - Attributes
  - Accessors
  - Extensions
- Core Interfaces and Ports
- Blocking Transport
- Nonblocking Transport
- Sockets
- Time
- Use Models

Session 7 & 8
Lecture: Developing Reusable Verification Components – I

  - Modeling Data Items for Generation
- Transaction-Level Components
- Creating the Driver
- Creating the Sequencer
- Connecting the Driver and Sequencer
- Creating the Monitor
- Instantiating Components
- Creating the Agent
- Creating the Environment
- UVM Configuration Mechanism

Session 9 & 10
Lecture: Developing Reusable Verification Components – II

- Enabling Scenario Creation
  - Declaring User-Defined Sequences
  - Generating Stimulus with Sequences and Sequence Items
  - Configuring the Sequencer's Default Sequence
  - Overriding Sequence Items and Sequences
- Managing End of Test
- Implementing Checks and Coverage
- Implementing Checks and Coverage in Classes
- Implementing Checks and Coverage in Interfaces
- Controlling Checks and Coverage

Session 11 & 12
Lecture: Using Verification Components

- Using a Verification Component
- Testbench Class
- Instantiating Verification Components
- Test Class
- Verification Component Configuration
- Creating and Selecting a User-Defined Test
- Creating Meaningful Tests
- Virtual Sequences
- Checking for DUT Correctness
- Scoreboards
- Implementing a Coverage Model

Session 13
Lecture: Introduction to Register Layer Classes Theory Concepts

- Overview
- Usage Model
- Access API
- Coverage Models
- Constructing a Register Model
- Back-door Access
- Special Registers
- Integrating a Register Model (RM)
- Randomizing Field Values
- Pre-defined Sequences

Session 14
Lecture: Advanced UVM Concepts

- The uvm_component Base Class
- The Built-In Factory and Overrides

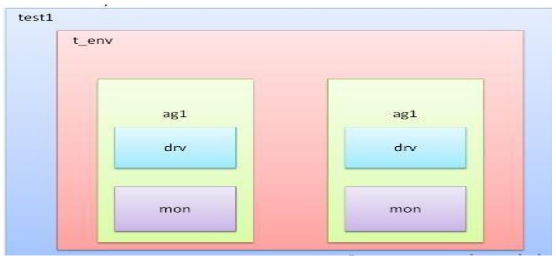
- Callbacks
- The Sequence Library
- Advanced Sequence Control
- Command Line Interface (CLI)

Session 15
Lecture: Case studies of UVM Environment

- Device Under Test (DUT) and its connection with environment, Scoreboards, coverage, predictors, monitors
- Verification of Traffic Light Controller / ATM Design / AHB Peripheral using UVM

Assignments:

- More lab practice as well as assignments should be taken of the following topics:
  - config_db
  - sequential circuits in UVM
  - scoreboard for FSM logic
  - verification for memory designs
  - multiple agent
  - environmental hierarchy
  - hands on examples of environment creation for combinational as well as sequential circuits
- Write code to create a proper UVM environment class which contains the following phases.
  - virtual function void build();
  - virtual function void connect();
  - virtual function void end_of_elaboration();
  - virtual function void start_of_simulation();
  - virtual task run();
  - virtual function void extract();
  - virtual function void check();
  - virtual function void report();
- Create an environment with primary reporting methods in UVM.Note that,In the top module, 3 objects of reporting must be created and different verbosity levels should be set.
- Write programs for the following requirements using field automation macros.
  - Define the enumerated types for packet types
  - Method to calculate the fcs
  - Test to check the packet implementation
- Create a UVM with classes that represent complete UVM sequence generation.
- Write a program to create a link between generated UVM sequence and driver.
- uvm_test is derived from uvm_component class and there is no extra functionality is added. select the testcase by passing the testcase name as string to uvm_root::run_test(<testcase_string>) method. Implement the environment with following topology



- Extend uvm_env class and define user environment.
- Utility macro should be declared.
- Declare the objects for agents.
- Define the constructor.
- Define build method.
- Define the phases
- Test case should be implemented.