UVM Advanced

Course Description

This 4-days course designed for advanced ASIC & FPGA verification engineers that would like to enhance their UVM skills to verify complex digital designs more efficiently.

The training is loaded with extensive practical hands-on labs to verify that the theory is understood plus introduce more use cases than covered in theory slides.

The first day introduces the self-checking testbench approach and how to create a scoreboard, then virtual sequences and virtual sequencers concept is introduced in details for complex use cases.

The second day teaches how to create a complex stimuli generation with various advanced techniques. Master-slave protocol is covered in details.

The third day starts with introduction of the advanced synchronization using the callback class, along with barrier synchronization, and the uvm_event class.

The day continuous by covering the end-of-test mechanism, how to raise and drop objections, how to debug UVM objections and how to set TB drain time.

The fourth day covers the connection reusability between DUT and top TB including BFM, port coerction, how to extract RTL parameters vs using package, the bind construct, UVM harness, and races between TB and DUT.

The training ends by covering how to handle exceptional situation such as reset, error injection and the uvm_heartbeat.

Course Duration

4 days

Goals

1. Implement an advanced scoreboard
2. Use in order and out of order comparator types
3. Buffer your transactions
4. Create virtual sequences and virtual sequencers
5. Generate complex stimuli using master-slave protocol, streaming operator to pack/unpack transactions, matching and different responses, packing dynamic data
6. Synchronize your transactions using the callback class and macros
7. Control the end-of-test mechanism
8. Reuse TB-DUT connectivity
9. Handle exceptional situations

Intended Users

Hardware/Software verification engineers who would like to enhance their skills for ASIC/FPGA designs with advanced UVM techniques.

Prerequisites

1. UVM fundamentals
2. SystemVerilog language
3. Verification guidelines
4. Experience with simulator

Course Material

1. Course book
2. Lab handbook (Phyton notebooks)
3. Virtual Machine with all necessary tools
4. Trainer solutions to all labs

Table of Contents

Day #1

Day #2

❖ Advanced Stimuli Generation

- Active and Passive Agent
- Master-Slave Protocols in UVM Environment
- A Proactive Master Agent
- A Reactive Slave Agent
- Data Flow in Proactive and Reactive Agents
- Sending Data Back to the Sequencer
- Stimulus with Matching and Different Response Types
- Creating Reactive Sequences
- **Lab #3: Reactive Slave**
- The SystemVerilog Streaming Operators
- Concatenating Class Fields Into Arrays
- The uvm_packer Class
- Packing and Unpacking Methods
- Packing Dynamic Structures - Metadata
- Child Sequencers
- Translator Sequences
- Layered Agents
- **Lab #4: Layered Sequences**

Day #3

❖ Advanced Synchronization

- Defining a Callback Class
- Registering the Callback
- Invoking the Callback - the `uvm_do_callbacks macro
- Barrier Synchronization
- The uvm_barrier Class
- The uvm_barrier Class Examples
- SystemVerilog Named Events
- Waiting For an Event - Races
- Persistent Trigger vs. the @ Operator
- The uvm_event Class
- The uvm_event Class Examples
- **Lab #5: UVM Synchronization**

❖ End-of-Test Mechanism

- Objection Mechanism
- Raising and Dropping Objections
- Objection Propagation and Performance
- Automatic Raising and Dropping Objections in UVM-1.2
- Debugging UVM Objection
- Setting TB Drain Time
- Hidden Components Functionality - The phase_ready_to_end() Method
- **Lab #6: UVM Objections**

Day #4

❖ TB-DUT Connection Reusability

- Port Coercion in SystemVerilog
- Making Interfaces Versatile with Ports Coercion
- Bus Functional Model
- Extracting RTL Parameters
- Tuning TB with RTL Parameters
- Traditional DUT to TB Connection
- SystemVerilog bind Construct
- UVM Harness
- Ports Width Mismatch
- Races between TB and DUT
- SystemVerilog Clocking Blocks
- **Lab 7: UVM Harness**

❖ Exceptional Situations

- Reset handling strategies
- Error injection
- Is my TB still alive? The uvm_heartbeat class
- **Lab 8: Handling Exceptions**