Design Verification Training Program

An advanced training program focusing on ASIC, FPGA, and SoC verification using Verilog, SystemVerilog, and UVM methodologies.

Detailed Syllabus

Module 0 – Fundamentals

- ASIC Design Flow Overview
- Introduction to Verification
- Verification Flow & Methodologies
- Testbench Architecture Basics
- Writing a Verification Plan

Module 1 – Verification with Verilog

- Basics of Verilog for Verification
- Modeling & Simulating Digital Designs
- Writing Testbenches in Verilog
- Verilog in FPGA & ASIC Verification

Module 1.1 – Introduction to SystemVerilog

- Why SystemVerilog?
- Key Extensions Over Verilog
- Unified Language for Design & Verification
- Behavioral and Structural Modeling

Module 2 – SystemVerilog Basics & Data Types

- Modules, Ports & Blocks (always, initial)
- Blocking vs Non-Blocking Assignments
- SystemVerilog Data Types:
- Bit-vectors, Integers, Reals
- Enums, Arrays, Structures, Classes
- Writing Synthesizable & Verification-Oriented Code

Module 3 – Interfaces & Interprocess Communication

- Interfaces
- Need for Interfaces
- Interface Ports & Modports
- Clocking & Procedural Blocks
- Connecting DUT & Testbench via Interfaces
- Interprocess Communication
- Fork-Join & Controls
- Semaphores
- Mailboxes

Module 4 – Object-Oriented Programming (OOP) in SystemVerilog

• Classes & Objects in SV
• Inheritance & Polymorphism
• Encapsulation & Modularity
• Building Reusable & Maintainable Verification Components

Module 5 – Randomization & Constraints

• Random Stimulus Generation
• Types of Randomization (Variables, Objects)
• Writing and Applying Constraints
• Corner-Case Testing with Randomization
• Debugging & Controlling Randomization

Module 6 – Introduction to UVM

• What is UVM & Why Use It?
• UVM Libraries & Guidelines
• Constrained-Random Stimulus in UVM
• Coverage-Driven Verification
• Best Practices for Debugging in UVM

Module 7 – UVM Testbench Architecture

• UVM Testbench Components:
• Test, Sequence, Driver, Monitor, Scoreboard
• Layered Testbench Design
• Connecting Components for Reuse & Scalability
• Managing Test Scenarios in UVM

Module 8 – Verification Component Development

• Sequencers: Generating Transactions
• Drivers: Stimulus to DUT
• Monitors: Capturing DUT Activity
• Scoreboards: Checking Results
• Agents: Bundling Components for Reuse

Module 9 – SoC Verification

• Introduction to SoC Verification Challenges
• Building SoC-Level Testbenches
• Protocol & IP-Level Verification
• Performance & Functional Verification
• Formal Verification in SoCs
• Debugging Complex SoC Issues

Module 10 – Projects

• Industry-Standard Verification Projects

- Complete UVM Environment Development
- SoC Verification Case Studies
- Final Evaluation & Best Practices