U2 - UVM

Universal verification methodology

Objectives

- Understand the role of UVM in modern verification environments
- Learn UVM architecture and testbench structure
- Master UVM components (test, env, agent, driver, monitor, sequencer, scoreboard)
- Understand UVM phasing mechanism and simulation flow
- Use UVM factory for flexible and reusable testbench design
- Apply configuration and resource databases effectively
- Implement transaction-level communication using TLM
- Develop sequences and control stimulus generation
- Build reusable and scalable UVM verification environments
- Use command-line options and reporting for debugging
- Understand and apply UVM Register Abstraction Layer (RAL)
- Apply advanced UVM features and best practices

Course environment

- Theoretical course
  - PDF course material (in English)
  - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - Example code, labs and solutions
  - Use of industry-standard tools such as ModelSim, QuestaSim, or Vivado Simulator for simulation and debugging

Prerequisites

- Strong knowledge of SystemVerilog
- Understanding of object-oriented programming (OOP)
- Familiarity with digital design and RTL concepts
- Basic programming knowledge (C/C++ or similar is a plus)
- Basic knowledge of testbench architecture
- Prior exposure to simulation and debugging tools

Environnement du cours

- Cours théorique
  - Support PDF (en anglais), imprimé en présentiel ; à distance via Teams.
  - Assistance du formateur tout au long de la formation.
- Activités pratiques (40-50% de la durée)
  - Exemples de code, exercices et solutions.
  - À distance : un PC Linux en ligne par stagiaire, avec carte émulée ou physique selon le cours.
  - Présentiel / sur site : un PC (un par binôme au-delà de 6 stagiaires), carte cible et manuel d'installation si nécessaire.
- Machine virtuelle préconfigurée téléchargeable pour refaire les TP après le cours.
- Chaque session débute par un point avec les stagiaires.

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

First Day

Introduction to UVM

- What is UVM
- Evolution OVM to UVM
- UVM architecture overview
- UVM testbench structure
- UVM hierarchy

Exercise : First sample UVM example

UVM Components

- Overview of UVM Components
- Base class
- Core Components
  - Test
  - Environment
  - Agent
  - Driver
  - Monitor
  - Sequencer
  - Scoreboard
- Component communication
  - TLM Connections
  - Analysis port

Exercise : Build a simple UVM testbench
Exercise : Run a basic simulation

Second Day

UVM Phasing

- Overview of UVM phases
- Build-time phases
  - Build phase
  - Connect phase
  - End of elaboration phase
- Run-time phase
  - Start of simulation phase
  - Run phase
  - Extract phase
  - Check phase
  - Report phase
- Phase mechanism
  - Objections
  - Phase synchronization

UVM Factory

- Factory overview
- Why factory

- Factory Mechanism
  - Registration
  - Creation
  - Overrides
    - Type override
    - Instance override
  - Factory classes
    - UVM object wrapper
    - Registry classes
  - Factory Debugging

Exercise : Register UVM components using factory
Exercise : Create components
Exercise : Apply type and instance overrides
Exercise : Debug factory behavior

Third Day

Configuration & Resources

- Configuration database
- Resource database
- Setting and getting configuration
- Hierarchical configuration

Transaction-Level Modeling

- TLM overview
- TLM interfaces
- Communication types
  - Blocking transport
  - Non-blocking transport
  - Analysis communication
- Ports / Export / Imps
  - Uni-directional
  - Bi-directional
  - Broadcast
  - TLM FIFOs

Exercise : Connect driver and sequencer using TLM
Exercise : Use analysis port between monitor and scoreboard
Exercise : Implement FIFO-based communication
Exercise : Create Coverpoints, Bins (including illegal bins)
Exercise : Add cross coverage
Exercise : Run simulation and analyze coverage results

Fourth Day

Sequences and Sequencer

- Sequence overview
- UVM Sequence item
- Sequencer
  - Arbitration
  - Communication with drivers
- Sequence Execution
  - Start method
  - Virtual Sequences

UVM Base Classes

- UVM object
- UVL Transaction
- UVM Sequencer item
- UVM Report Object
- Utility Methods

Exercise : Create sequence item and sequence
Exercise : Send transaction to driver
Exercise : Implement virtual sequence

Building a Full UVM Environment

- Testbench architecture
- Connecting Components
- Reusable Agents

Exercise : DUT integration
Exercise : Driver implementation
Exercise : Monitor implementation
Exercise : Scoreboard design
Exercise : Test creation
Exercise : Running simulation

Fifth Day

Command line & Simulation Control

- UVM command-line arguments
- Verbosity control
- Runtime configuration
- Debug options
- Reporting control

Register Abstraction Layer (RAL)

- RAL overview
- Register model
- Register access methods
- Frontdoor vs backdoor
- RAL integration

Advanced Topics

- Virtual interfaces
- Callbacks
- UVM reporting
- Debugging techniques
- Performance optimization

Final Exercise / Mini project

- Build a complete UVM environment for a simple DUT:
  - Driver + Monitor + Scoreboard
  - Sequences
  - Functional checking