UVM (Universal Verification Methodology) For SW Engineers

Tuan Nguyen-viet

What is UVM ?

➔ **DUT**

- A standardized methodology for verifying **digital designs** and **systems-on-chip** (SoCs) in the semiconductor industry.

➔ **Testbench**

- UVM is built on top of the **SystemVerilog** language
  - and provides a framework for creating modular, reusable testbench components
    - that can be easily integrated into the **design verification process**.
- It also includes a set of guidelines and best practices for developing testbenches,
  - as well as a methodology for
    - running simulations
    - and analyzing results.

➔ UVM has become the de facto standard for design verification in the semiconductor industry,

  - and is widely used by **chip designers** and **verification engineers**
    - to verify the correctness and functionality of their designs. [1]

➔ UVM offers significant benefits in terms of reusable and scalable testbenches. [2]

What is UVM ? (2): SystemVerilog Language Classes

- The **Accellera Systems Initiative** worked with **leaders in SoC** and **EDA** to develop **UVM**.

- This API / methodology is meant for building functional testbenches for SoCs.

- UVM being constructed in SystemVerilog
  - is supported by simulators from all vendors.

- SystemVerilog is a language just like Verilog and has its own constructs, syntax and features,
  - but UVM is a framework of SystemVerilog classes
    - from which fully functional testbenches can be built.

- There is only one prerequisite to learn UVM,
  - and that is SystemVerilog
    - because it is the foundation for the tower that is UVM.

What does UVM contain ?

UVM contains a set of **pre-defined** classes and methods

- that enable users to create modular, reusable testbench components
  - for verifying digital designs and systems-on-chip (SoCs).

Some of the **key** components of UVM include:

1. **Testbench Components**:
   - UVM provides a set of base classes that can be extended to create testbench components,
     - such as drivers, monitors, scoreboards, and agents.

2. **Transactions**: Transactions are used to model the communication between the design-under-test (**DUT**) and the testbench.
   - UVM provides a transaction class that can be extended to create transaction objects that carry information between the **DUT** and the testbench.

3. **Phases**: UVM defines a set of simulation phases that enable users to control the order in which testbench components are created, initialized, and executed.

What does UVM contain ? (2)

Some of the **key** components of UVM include:

4. **Messaging and Reporting**: UVM provides a messaging and reporting infrastructure
   - that enables users to output information about the simulation,
     - such as warnings, errors, and debug information.

5. **Configuration**: UVM provides a configuration database that allows users to store and retrieve *configuration information* for testbench components.

6. **Functional Coverage**: UVM provides a mechanism for tracking functional coverage, which is used to ensure that the design has been thoroughly tested.

7. **Register Abstraction Layer**: UVM provides a register abstraction layer (**RAL**) that simplifies the process of creating and accessing *register maps*.

References

- [1] https://www.chipverify.com/tutorials/uvm
- [2] https://verificationacademy.com/topics/uvm-universal-verification-methodology/

Thank You