

Distinction between Driver Sequences and Test Sequences in UVM

**Raj Kapoor Singh**

ASIC Engineer | PCIe | CXL | Trainer | bazinga-vlsi.com



July 24, 2023

In the realm of hardware verification, the Universal Verification Methodology (UVM) has emerged as a widely used standard to ensure the correctness and functionality of complex digital designs. Within the UVM framework, sequences play a crucial role in generating stimulus and orchestrating test scenarios. However, UVM sequences can be broadly categorized into two types: Driver Sequences and Test Sequences. Let us delve into the distinctions between these two sequence types, their purposes, and how they contribute to the overall verification process.

Driver Sequences - Protocol-Specific Stimulus Generation:

Driver sequences in UVM are designed to generate protocol-specific stimuli for the Design Under Test (DUT) by interacting directly with the driver. The driver is responsible for translating the high-level transactions produced by the sequences into appropriate protocol-specific signals to be sent to the DUT. Driver sequences are closely associated with the DUT's interface and focus on low-level details, adhering to the protocol rules and ensuring proper communication with the hardware.

Key Characteristics of Driver Sequences:

- Low-level and protocol-specific.
- Interact directly with the driver.
- Responsible for generating individual transactions or small sequences of transactions.
- Translate high-level transactions into protocol-specific signals for the DUT.

Test Sequences - High-Level Test Orchestration:

In contrast to driver sequences, test sequences in UVM operate at a higher level of abstraction. They encapsulate entire test scenarios or use cases to be verified and orchestrate the overall flow of the testbench. Test sequences are more generic and abstracted from the protocol details, enabling them to be reused across different tests and test scenarios. They coordinate the interactions between various components in the testbench, such as drivers, monitors, scoreboards, and coverage collectors, to verify the DUT's behavior in diverse test scenarios.

Key Characteristics of Test Sequences:

- High-level and abstracted from protocol details.
- Orchestrate the overall flow of the testbench.
- Coordinate interactions between multiple components in the testbench.
- Represent larger test scenarios involving multiple transactions and DUT interactions.

Utilizing Driver and Test Sequences in UVM Verification:

In UVM verification environments, the effective utilization of both driver and test sequences is essential to ensure thorough testing of the DUT. Driver sequences focus on generating stimulus at the protocol level, facilitating communication with the DUT and testing specific protocol functionalities. They are often used in conjunction with the protocol-specific checks implemented in monitors to validate the correctness of the DUT's responses.

Test sequences, on the other hand, provide a higher-level perspective on the DUT's behavior. They are responsible for executing complete test scenarios, which may involve sequences of transactions and responses. By orchestrating these complex test scenarios, test sequences can effectively capture the DUT's performance under various real-world conditions and edge cases.

Moreover, test sequences can incorporate functional coverage and assertions to track the progress of the test and ensure that the DUT is behaving as expected. The coverage information generated by test sequences helps verification engineers identify untested portions of the DUT and aids in achieving comprehensive test coverage.

The distinction between driver sequences and test sequences in UVM verification is critical for successful and efficient hardware verification. Driver sequences, with their protocol-specific focus, generate stimulus at the hardware interface level, enabling accurate communication with the DUT. They are fundamental for verifying specific protocol functionalities and interactions with the DUT.

On the other hand, test sequences provide high-level orchestration of test scenarios, making it possible to verify the DUT's functionality in diverse situations. By utilizing both driver and test sequences effectively, verification engineers can build robust and comprehensive UVM testbenches that ensure the accuracy and reliability of modern digital designs. The synergy between driver sequences and test sequences is key to achieving efficient and thorough verification, ultimately leading to the successful deployment of complex digital systems.

Comments

👍 44 · 1 comment



Like



Comment



Share

Add a comment...



Most recent ▾



Srinivasan Venkataramanan · 1st

CEO & Founder AsFigo

1y ...

By any chance is this AI created article? Sounds like that to me, maybe the model that wrote this article will need more real world UVM training dataset?

Like · 🗨️ 7 | Reply



Raj Kapoor Singh


ASIC Engineer | PCIe | CXL | Trainer | bazinga-vlsi.com



Ansuman, Tejaswini and 14 others you know followed

✓ Following

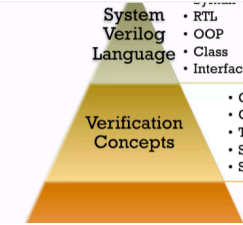
More articles for you



Published on LinkedIn

EDA - Universal Verification Methodology

Prateek Khanna



The pyramid is divided into three horizontal layers. The top layer is labeled 'System Verilog Language' and lists: • RTL, • OOP, • Class, • Interface. The middle layer is labeled 'Verification Concepts' and lists: • Constrained Random, • Coverage Driven, • Transaction Level, • Sequences, • Scoreboards. The bottom layer is labeled 'Verification Concepts' and lists: • Base Classes, • Use Cases.

Configuration object in UVM test bench:
For top level and agent's level/UVC:

Ajazul Haque

5



The diagram shows the UVM architecture. At the top are 'sequence 1' and 'sequence 2'. Below them is the 'environment' block containing 'coverage collector', 'virtual sequencer', and 'scoreboard'. The 'agent' block contains 'coverage collector', 'virtual sequencer', and 'reorder'. The 'DUT' (Device Under Test) is at the bottom, connected to 'signals' and 'input/output'. A 'checker' block is also shown on the right, connected to 'signals' and 'input/output'.

Comprehensive Verification Methods for
RISC-V Designs By www.vlsijobseekers.com

MANJUNATHA VIJAYA