Building UVM Testbenches for High Quality Serial Design IPs

Modelling real world scenarios

Kilaru Vamsikrishna, Verification Architect, Cadence Design Systems
Sushrut B Veerapur, Verification Director/Architect, Cadence Design Systems

Problem Statement/Introduction

Serial Design IPs are expected to be built for peak performance under less-than-ideal operating conditions

In a directed test case based Testbench, stimulus is generally contained to driving one or more primary data path interfaces, a register or control interface, and some miscellaneous interfaces.

To generate stimulus targeting all the corners of the DUT, multiple virtual sequences need to execute concurrently.

In a traditional approach traffic sequence running as primary sequence in foreground must be changed to one of the background sequences to execute multiple scenarios randomly.

Robust verification stressing the Design IP is a minimum requirement
- Mimicking real world traffic
- Mimicking real world feature stress cases
- Mimicking real world errors, low power, resets

Proposed Methodology/Advantages

Constraint Random "One-Test" Testbench Architecture
- Base test can create all possible test scenarios; built for true system-level verification
- Randomization is built into the testbench

In a single test, combinations of errors, resets, low power and register tests will be covered.

Some example real time scenarios which can happen randomly
- Resets ⇔ Errors ⇔ Registers ⇔ Low Power ⇔ Errors ⇔ Resets ⇔Registers
- Register access with traffic and functional test
- On the fly reset with traffic
- Good traffic mixed with error traffic

All scenarios which need to be intermixed with each other should start as a background sequence

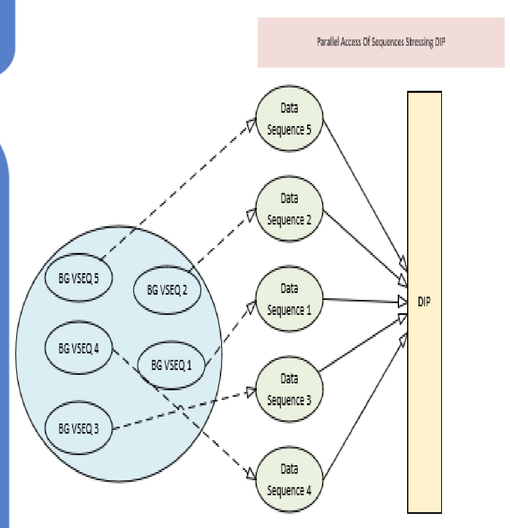
Implementation Details/Diagram

The selection of the background (BG) sequence is controlled by Test Config/Scenario config

- Scenario class, act as control object to synchronize all the layers of TB into ONE common platform
- Scenario class prevents the mutually exclusive scenarios and provides the fine control and randomizes the legal scenarios

Every background sequence is a virtual sequence which starts the actual data sequence and drives the sequence item on the interface

- Sequence is always active till the operation is completed
- Runs in the background and does not block any other sequences (called in fork join_none)
- Do not raise and drop objections
- Handshake between each BG sequence and rest of the TB
  - Every BG sequence initiates the cause and checks the effect of it
  - Updates the status when in InProgress to poll for end of the test
  - Encapsulate the sequence controls
    - Test ⇔ Main Sequence ⇔ Virtual Sequence ⇔ Layered Sequence ⇔ Actual Data Sequence
  - One common configuration object shared across TB along with hierarchical configurations
  - Background sequence starts on a null sequencer within the main/virtual sequence



Implementation Details/Flow Chart

Testbench Requirements

On the Fly Reset Handling
- Every component and Sequence in the testbench is reset aware
- Resets are always monitored and takes the reset action based on the type of reset(soft, hard)

Fully random and fine control sequences
- Scenario configuration controls all the BG sequences
- Command line args package provides the desired control to sequences through scenario and config objects

Base scoreboard
- APIs to turn off and turn on dynamically
- APIs to flush the scoreboard queues and reset scoreboard
- APIs to query the status

End of Test criteria
- Virtual sequence waits for every BG Sequence to be completed
- All the scoreboard Q to be empty
- Depends on the implementation specific and protocol requirements
- Guard every thread with a watch dog timer to avoid hang scenarios

Results Table

This methodology is implemented to verify one of industry's most Complex Serial Design IP

Owing primarily to the methodology, multiple dependent cross feature, and corner case scenarios issues were exposed in the RTL

Full randomization of legal scenarios with fine control is a key aspect of the methodology

This methodology driven verification established that real work stress scenarios can be well exercised in functional simulation of the design

Conclusion

Though not impossible but it is quite inefficient to plan and create all combination of intermixed scenarios

Efficient strategy would be to be push the problem to Testbench
- To exercise full legal space with more random combinations
- Layering sequences smartly that spawn multiple background threads

Fine control knobs are mandatory to provide user control to select scenarios

REFERENCES