Class-Based Verification of AXI4-Lite Protocol

Team Members:
Sarath Jampani
Venkata Vyshnavi Julakanti
Nithesh Kamireddy
Satyam Sharma

1. Introduction

The AXI4-Lite protocol is a streamlined version of AXI4, designed for low-bandwidth, simple memory-mapped communication. It supports independent read and write address and data channels, allowing for concurrent operations. This makes it ideal for verifying control registers and simple peripherals.

This report outlines a class-based verification approach using SystemVerilog and object-oriented concepts to construct a modular, reusable, and scalable testbench environment for an AXI4-Lite slave interface.

2. Verification Architecture Overview

The testbench is built using a layered architecture with two independent agents:

- **Write Agent**
- **Read Agent**

Each agent consists of the following components:

- **Transaction Class:** Defines fields such as address, data, strobe for write and address for read, used to describe individual operations.
- **Generator:** Randomizes transaction objects and pushes them to respective mailboxes.
- **Driver:** Fetches transactions from the mailbox and applies them to the DUT using a virtual interface.
- **Monitor:** Observes DUT signal changes and packages results into transaction objects, sending them to the scoreboard.

3. Interface and Virtualization

A SystemVerilog `interface` is used to bundle AXI signals and connect DUT and testbench. The interface includes `modports` to separate read and write channels, enabling concurrent access. Drivers and monitors use virtual interfaces to communicate with the DUT indirectly, improving reusability and modularity.

4. Scoreboard and Reference Model

The scoreboard is central to functional checking. It receives monitored transactions from both agents and compares DUT responses to the expected values generated by a golden reference model. This reference model simulates memory behavior:

- **On Write:** Updates memory at the specified address with the given data and strobe.
- **On Read:** Returns the value from the memory model and compares it with the DUT output.

Scoreboard comparisons ensure functional correctness across edge cases and protocol scenarios.

5. Simulation Output Highlights

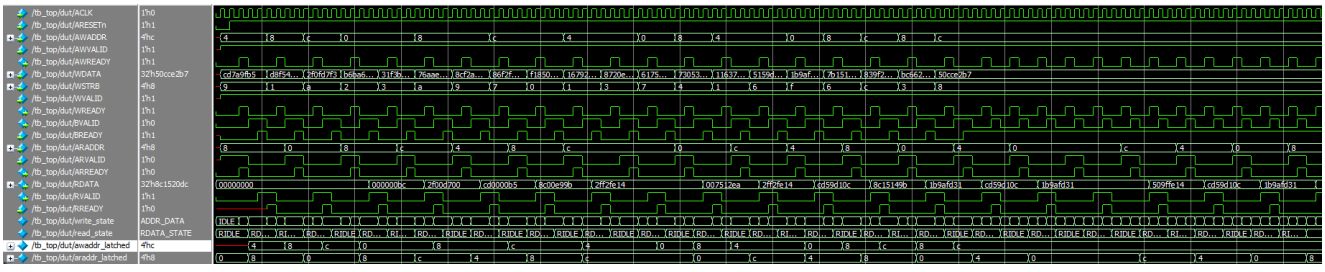
Write Transactions:

```
# write_transcation : Addr - 4 , strobe - 9 , data - cd7a9fb5
# write_transcation : Addr - 0 , strobe - f , data - 1b9afd31
# write_transcation : Addr - 8 , strobe - 3 , data - bc6620dc
# write_transcation : Addr - c , strobe - 8 , data - 50cce2b7
```

Read Transactions and Scoreboard Results:

```
# READ DRIVER: ARADDR = 0
# Scoreboard PASS: Addr=0x0 | Data=0x1b9afd31 | Expected=0x1b9afd31
# READ DRIVER: ARADDR = 8
# Scoreboard PASS: Addr=0x8 | Data=0x8c1520dc | Expected=0x8c1520dc
# READ DRIVER: ARADDR = c
# Scoreboard PASS: Addr=0xc | Data=0x509ffe14 | Expected=0x509ffe14
```

Waveform Snapshot:



Note: The waveform highlights concurrent read and write operations with signal transitions on valid, ready, address, and data lines.

6. Key Verification Advantages

- **Concurrent Validation:** Separate agents for read/write allow for testing concurrent channel usage.
- **Randomized Stimuli:** CRV via generators ensures thorough scenario coverage.
- **Scalable Architecture:** Agents and components can be reused for other protocols or enhanced designs.
- **Self-Checking:** The scoreboard automates output verification against a reference model.

7. Conclusion

The class-based testbench successfully verified the AXI4-Lite slave interface using randomized, self-checking principles. By employing separate agents, virtual interfaces, and a memory-based reference model, the environment achieved complete validation with clean scoreboarding. This methodology promotes reuse, maintainability, and extensibility for more complex verification projects in the future.