Fundamentals of Pre-Silicon Validation || Winter -2024

Implementation and Verification of Asynchronous FIFO using both Class based and UVM methodologies.

VERIFICATION TEST PLAN

Project Team:
Jaswanth Pallappa, jaswanth@pdx.edu
Brunda Marpadaga, brunda@pdx.edu
Maha Lakshmi Potabattuni, mahalak@pdx.edu
Akhila Veedula, akhilav@pdx.edu

Introduction

Objective of the verification:

The main objective of this verification plan is to thoroughly verify the design functionality and correctness through various verification methods and implementing error correction mechanisms and checking for errors.

Division of tasks:

Team members and tasks allocation:

• **Brunda**: She will focus on implementing and testing the core functionality of the asynchronous FIFO design. This includes refining the write functionality with two write idle cycles idle cycles according to specifications, designing and testing the FIFO memory, and ensuring the correct functionality of the generator and monitor modules. Additionally, she will contribute to writing the verification plan document and providing support in other areas as needed.

• **Jaswanth:** He will be responsible for implementing and verifying the read functionality of the asynchronous FIFO design, ensuring one read idle cycle as per requirements. He will thoroughly test the design with various test cases, write the Design Specification document, and develop the driver and scoreboard functionality. He will also contribute to testing other modules and aiding as required.

• **Maha Lakshmi**: She will lead the integration efforts by implementing the design through interfaces and ensuring proper connectivity with the top module. Additionally, she will calculate and document the depth of the FIFO, write the scoreboard, and verify the percentage thoroughly. She will also work on creating the overall environment, tests, and testbench top module. She will collaborate with other team members to ensure seamless integration and functionality.

• **Akhila Veedula:** Akhila will join the team to assist in various tasks related to the asynchronous FIFO design verification. Her responsibilities will include contributing to the implementation and testing of both read and write functionalities, supporting the development of test cases, and assisting in documentation tasks. Akhila will work closely with coverage and assertion aspects of the verification plan ensuring the overall development of the project.

VERIFICATION TEST PLAN:

Functional Verification:

- Efficient testing of fundamental features such as write/read operations, overflow/underflow scenarios, and proper control signal handling is part of the functional verification process for an asynchronous FIFO.
- Extra care is taken to ensure correct synchronization and handle any possible metastability problems by considering the asynchronous behavior between clock domains. To verify performance and resilience, test cases include a variety of scenarios, such as boundary conditions and stress testing.
- The correct operation of the FIFO is confirmed, and verification environments are set up with the aid of simulation tools.

Corner Case Testing:

Test Case Scenarios:

Test Full: This test case involves writing data to all available locations within the FIFO until it becomes full. The verification process includes monitoring the FIFO's full flag or status to confirm that it accurately reflects when the FIFO is indeed full. Additionally, the behavior of the FIFO when attempting to write data when it is full should be observed. It should either block further writes, generate an error, or employ some other predefined behavior.

Test Empty: In this test case, the FIFO is initially empty, and data is read from it until it becomes empty again. Similar to the "Test Full" case, the empty flag or status of the FIFO needs to be monitored to ensure it correctly indicates when the FIFO is empty. Also, the behavior of the FIFO when attempting to read from it when it is empty should be observed, such as blocking reads, generating an error, or employing a specific behavior.

Test Full Error: This test case verifies the behavior of the FIFO when attempting to write data to it while it is already full. The expected behavior here is that the FIFO should either raise an error, assert a full flag, or employ some other mechanism to indicate that a write operation cannot be performed due to the FIFO being full.

Test Empty Error: Similar to the "Test Full Error" case, this test case verifies the behavior of the FIFO when attempting to read data from it while it is empty. The expected behavior is that the FIFO should raise an error, assert an empty flag, or use some other mechanism to indicate that a read operation cannot be performed due to the FIFO being empty.

Test Concurrent Write-Read: This test case involves concurrently writing and reading data to/from the FIFO. It verifies the concurrent operation of both read and write ports of the FIFO and ensures that there are no race conditions or other synchronization issues. The behavior of the FIFO under simultaneous read and write operations should be observed to ensure correct operation.

Coverage Metrics:

- Functional coverage, code coverage, and assertion coverage are the three types of coverage metrics for an Asynchronous FIFO.

- Functional coverage monitors how well test scenarios capture all the necessary behavior of the FIFO, while code coverage counts the proportion of code lines, branches, and conditions that are tested by the verification tests, guaranteeing thorough testing of the design.
- Assertion coverage assesses how well assertions capture design properties and identify violations. These metrics help determine how thorough the verification effort was, where more testing might be needed, and how confident one can be in the accuracy and resilience of the FIFO implementation.

Verification Environment:

We create a UVM (Universal Verification Methodology) based verification environment for an Asynchronous FIFO involves several steps to ensure thorough testing and validation of the design.

UVM Testbench Architecture:

- Define the overall testbench architecture following the UVM methodology, including components such as agents, sequences, drivers, monitors, and scoreboards.
- Organize the testbench hierarchy to facilitate modularization, scalability, and reusability.

Agent Configuration:

- Implement separate agents for the FIFO's input and output interfaces, encapsulating the functionality of the driver, monitor, and sequencer for each interface.
- Configure the agents to interface with the FIFO design, handling data transactions, protocol checks, and synchronization between clock domains.

Sequences and Sequencers:

- Develop sequences to generate stimulus for write and read operations, covering various scenarios such as empty, full, overflow, and underflow conditions.
- Implement sequencers to control the generation and scheduling of sequence items, ensuring proper synchronization and coordination with the rest of the testbench.

Driver and Monitor:

- Create drivers responsible for driving stimulus to the FIFO's input interface, including data, control signals, and timing constraints.
- Develop monitors to capture and analyze transactions on the FIFO's output interface, verifying data integrity, protocol compliance, and timing constraints.

Scoreboard:

- Implement a scoreboard to compare the expected and observed behavior of the FIFO, ensuring correctness and completeness of test results.
- Verify data consistency, FIFO occupancy, and control signal interactions between the input and output interfaces.

Stimulus Generation:

- Inputs are generated using randomization and constraints.

REFERENCES:

- **https://github.com/teekamkhandelwal/asynchronous_fifo/blob/main/r_pointer_epty.v**

- **http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf**
-