**VERIFICATION PLAN**

**Fundamentals of Pre-Silicon Validation Spring -2024**

**Asynchronous FIFO**

**Rafath Achugatla, Divya Sri Ayluri, Nandini Maddela,Pooja Satpute**

**Date: 04/22/2024**

Contents

[1 Introduction 2](#_Toc164761780)

[1.1 Objective of the Verification Plan 2](#_Toc164761781)

[1.2 Functional Intent 2](#_Toc164761782)

[1.3 High Level Block Diagram 2](#_Toc164761783)

[1.4 Design Description 2](#_Toc164761784)

[1.5 Specifications of the Design 3](#_Toc164761785)

[2 Verification Levels 3](#_Toc164761786)

[2.1 Module Hierarchy 3](#_Toc164761787)

[2.2 Controllability and Observability 3](#_Toc164761788)

[2.3 Interface Signals 3](#_Toc164761789)

[3 Required Tools 5](#_Toc164761790)

[4 Risks and Dependencies 5](#_Toc164761791)

[5 Functions to be verified 8](#_Toc164761792)

[6 Tests and Methods 10](#_Toc164761793)

[7 Resource Requirments 12](#_Toc164761794)

[8 References 13](#_Toc164761795)

## 1 Introduction

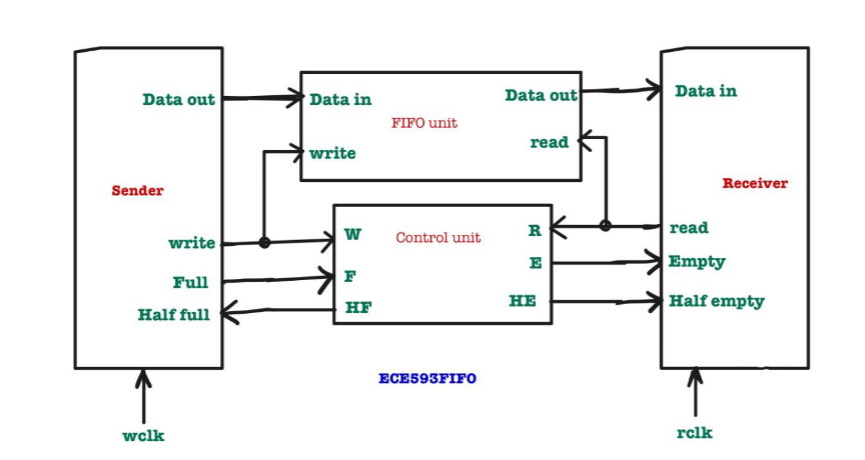
## Objective of the Verification Plan

The objective of this verification plan is to ensure the functional correctness and performance compliance of the FIFO design. Specifically, the plan aims to verify a FIFO with a depth of 333, read idle cycles of 1, and write idle cycles of 2.

## Functional Intent

Asynchronous FIFO’s main purpose is to safely pass data from one clock domain to another asynchronous clock domain.

## 1.3 High Level Block Diagram



## 1.4 Design Description

An asynchronous FIFO design consists of data storage, control logic, and read and write pointers. Write operations store data at the write pointer position, while read operations retrieve data from the read pointer position. Control logic manages FIFO state, generating status signals indicating full, empty, half full, and half empty conditions. Data integrity is maintained through synchronization techniques, crucial for reliable data transfer.

## 1.5 Specifications of the Design

* Sender Clk Frequency(Mhz) = 500 Mhz
* Write Idle cycles = 2
* Write Burst size = 1024
* Receiver Clk Frequency(Mhz) = 225 Mhz
* Read Idle Cycles = 1
* Calculated minimum depth of the FIFO is 333.

# 2 Verification Levels

## 2.1 Module Hierarchy

We are verifying the FIFO design at the block level as it encapsulates the complete functionality of the FIFO.

## 2.2 Controllability and Observability

Controllability and Observability are achieved through dedicated input and output ports of the FIFO module, allowing effective stimulus application and result monitoring.

## 2.3 Interface Signals

**FIFO Module Interface:**

**Definition:** The primary interface of the FIFO module, including input and output ports.

**Specifications:**

winc: Write enable signal. wclk:

Write clock signal.

wrst\_n: Active-low write reset signal.

rinc: Read enable signal.

rclk: Read clock signal.

rrst\_n: Active-low read reset signal.

wdata: Input databus for write operations.

rdata: Output data bus for read operations.

wfull: Output signal indicating FIFO full condition.

rempty: Output signal indicating FIFO empty condition.

**Memory Interface (FIFO mem):**

**Definition:** The interface between the FIFO module and the memory subsystem (FIFO mem).

**Specifications:**

winc: Write enable signal.

wfull: Input signal indicating FIFO full condition.

wclk: Write clock signal.

waddr: Write address bus.

raddr: Read address bus.

wdata: Input data bus for write operations.

rdata: Output data bus for read operations.

**Read Pointer Empty Module Interface (rptr\_empty):**

**Definition:** The interface of the module handles read pointer and empty condition.

**Specifications:**

rinc: Read enable signal.

rclk: Read clock signal.

rrst\_n: Active-low read reset signal.

rq2\_wptr: Input read pointer with write side.

rempty : Output signal indicating FIFO empty condition.

raddr: Output read address.

**Write Pointer Full Module Interface (wptr\_full):**

**Definition:** The interface of the module handles write pointer and full condition.

**Specifications:**

winc: Write enable signal.

wfull: Output signal indicating FIFO full condition.

wclk: Write clock signal.

wq2\_rptr: Input read pointer with write side.

waddr: Output write.

address. wptr: Output write pointer.

**Synchronization Modules (sync\_r2w, sync\_w2r):**

**Definition:** Modules ensuring synchronization between read and write pointers.

**Specifications:**

wclk, wrst\_n: Write clock and reset signals.

rptr: Read pointer.

rq2\_wptr, wq2\_rptr: Synchronized read and write pointers.

# 3 Required Tools

**3.1 Software and Hardware Tools**

Questa sim.

# 4 Risks and Dependencies

**Incomplete Specifications:**

**Risk:** Incomplete or ambiguous specifications may lead to misinterpretations during verification.

**Contingency/Mitigation:** Regularly communicate with design and specification teams to clarify any ambiguities. Document assumptions made during verification.

**Concurrency Issues:**

**Risk:** Concurrency-related problems may arise during read and write operations.

**Contingency/Mitigation:** Thoroughly test scenarios involving concurrent read and write operations to identify and address any concurrency issues.

**Performance Bottlenecks:**

**Risk:** Performance issues, such as slow read or write operations, may impact the overall functionality.

**Contingency/Mitigation:** Implement performance-related tests to ensure compliance with specifications. Optimize design if performance bottlenecks are identified.

**Integration Challenges:**

**Risk:** Integration issues may arise when connecting the FIFO design with other system components.

**Contingency/Mitigation:** Conduct integration testing with other modules early in the verification process. Collaborate with other teams to address interface compatibility.

**Assertion Failures:**

**Risk:** Assertions may fail to capture specific corner cases or may trigger false positives.

**Contingency/Mitigation:** Regularly review and update assertions. Use multiple assertion types to cover various aspects of the design.

**Incomplete Test Coverage:**

**Risk:** Insufficient test coverage may lead to undetected bugs or corner cases.

**Contingency/Mitigation:** Develop a comprehensive test plan with a variety of scenarios to achieve high test coverage. Utilize code coverage tools to identify areas that need additional testing.

**Misalignment with Design Changes:**

**Risk:** Changes in the design may not be accurately reflected in the testbench or test cases.

**Contingency/Mitigation:** Establish a robust change management process. Regularly synchronize the testbench and test cases with the latest design specifications.

**Resource Constraints:**

**Risk:** Insufficient computing resources may impact the efficiency of simulations.

**Contingency/Mitigation:** Optimize testbenches for efficiency. Use parallel simulation options if available. Upgrade hardware resources if necessary.

**Model-Reality Mismatch:**

**Risk:** The model may deviate from the real-world behavior, leading to incorrect verification results.

**Contingency/Mitigation:** Regularly validate the model against real-world scenarios. Update the model based on feedback from silicon testing.

**Incomplete Documentation:**

**Risk:** Lack of comprehensive documentation may hinder understanding and troubleshooting.

**Contingency/Mitigation:** Maintain up-to-date documentation. It includes assumptions, known issues, and resolutions for future reference.

**Ambiguous Error Handling:**

**Risk:** Unclear error handling mechanisms may make it challenging to identify and debug issues.

**Contingency/Mitigation**: Clearly define error handling procedures. Implement detailed logging and reporting mechanisms.

**Inadequate Regression Testing:**

**Risk:** Changes to the design or testbench may introduce regressions that go undetected.

**Contingency/Mitigation:** Implement a comprehensive regression testing suite. Verify the integrity of existing functionalities after each update.

**Unrealistic Test Scenarios:**

**Risk:** Test scenarios may not accurately represent real-world usage.

**Contingency/Mitigation:** Collaborate with domain experts to create realistic test scenarios. Incorporate feedback from system architects.

**Human Error:**

**Risk:** Mistakes in testbench development, test case creation, or result analysis.

**Contingency/Mitigation:** Implement thorough code reviews, use automation where possible, and encourage a culture of careful verification practices.

# 5 Functions to be verified

**5.1 Functions from specifications and implementation**

**Write Operation:**

Function: Verify that data can be successfully written into the FIFO.

Description: Ensure that the FIFO accepts data when the write enable signal is asserted. Check if the data is correctly stored in the memory.

**Read Operation:**

Function: Verify that data can be successfully read from the FIFO.

Description: Ensure that the FIFO provides valid data when the read enable signal is asserted. Check if the data read matches the expected values.

**Write and Read Concurrency:**

Function: Verify simultaneous write and read operations.

Description: Test scenarios where write and read operations occur concurrently. Ensure that the FIFO handles simultaneous read and write requests without data corruption.

**FIFO Full Condition:**

Function: Verify the FIFO full condition.

Description: Write data into the FIFO until it reaches its maximum depth. Verify that the FIFO signals a full condition correctly.

**FIFO Empty Condition:**

Function: Verify the FIFO empty condition.

Description: Read data from the FIFO until it becomes empty. Verify that the FIFO signals an empty condition correctly.

**Idle Write Cycles:**

Function: Verify the behavior during idle write cycles.

Description: Confirm that the FIFO remains stable during idle write cycles (when no data is being written). Check for any unintended side effects during idle write periods.

**Idle Read Cycles:**

Function: Verify the behavior during idle read cycles.

Description: Confirm that the FIFO remains stable during idle read cycles (when no data is being read). Check for any unintended side effects during idle read periods.

**Write Error Handling:**

Function: Verify error handling during write operations.

Description: Test scenarios where the FIFO receives write requests beyond its capacity. Ensure that appropriate error handling mechanisms are in place.

**Read Error Handling:**

Function: Verify error handling during read operations.

Description: Test scenarios where read requests are made when the FIFO is empty. Ensure that appropriate error handling mechanisms are in place.

**Reset Operation:**

Function: Verify the reset functionality.

Description: Assert the reset signal and verify that the FIFO resets to its initial state, clearing all stored the data.

**Sequential Write and Read:**

Function: Verify sequential write and read operations.

Description: Perform a series of sequential write and read operations to ensure proper data flow through the FIFO.

**Asynchronous Write and Read:**

Function: Verify asynchronous write and read operations.

Description: Introduce variations in write and read timing to ensure that the FIFO can handle asynchronous operations.

**FIFO Half Empty Condition:**

Function: Verify the FIFO half empty condition.

Description: Write data into the FIFO until it reaches its half of the depth. Verify that the FIFO signals a half empty condition correctly.

# 6 Tests and Methods

**6.1 Testing methods to be used: Black/White/Gray Box**

**Black Box Testing:** Functional verification based on specifications.

**White Box Testing:** Code coverage analysis and assertion-based verification.

**Gray Box Testing:** Scenario-based testing for corner cases.

**Advantages and Disadvantages**

**Black Box Testing:** PRO - High-level coverage. CON - Limited visibility into internals.

**White Box Testing:** PRO - In-depth analysis. CON - May miss system-level issues.

**Gray Box Testing**: PRO - Comprehensive testing. CON - Increased simulation time.

**6.2 Verification Strategy: ((Dynamic Simulation, Formal Simulation, Emulation etc.)**

**Describe why you chose the strategy?**

Verification Strategy: Dynamic Simulation.

Reasoning: Dynamic simulation balances accuracy and efficiency for FIFO verification.

**6.3 Test Scenarios**

|  |  |
| --- | --- |
| **Name** | **Test Case Description** |
| Basic 0,1 test for every bit | * Generate transactions with different values for each bit of wdata. * Ensure that each bin for wdata\_bit0 through wdata\_bit7 is hit. |
| Rinc check 0,1 | * Generate transactions with different values for rinc. * Ensure that each bin for rinc\_bin is hit. |
| Winc check 0,1 | * Generate transactions with different values for winc. * Ensure that each bin for rinc\_winc is hit. |
| wrst\_n | * Generate transactions with different values for wrst\_n. * Ensure that each bin for rinc\_wrst\_n is hit. |
| rrst\_n | * Generate transactions with different values for rrst\_n. * Ensure that each bin for rinc\_rrst\_n is hit. |
| **Name** | **Test Case Description** |
| wfull | * Generate transactions to fill the FIFO. * Generate transactions to empty the FIFO. * Ensure that both bins for wfull\_bin are hit. |
| rempty | * Generate transactions to fill the FIFO. * Generate transactions to empty the FIFO. * Ensure that both bins for rempty\_bin are hit. |
| Sequential patters | Test with specific patterns that may exercise specific functionality or corner cases in your design. |
| Wdata check one hot bits | * Generate transactions with different one-hot encoded values for wdata. * Ensure that each bin for wdata\_onehot is hit. |
| Edge Cases:Wdata check for ff and 00 | * Generate transactions with border values (0 and 255) for wdata. * Ensure that both bins for wdata\_border are hit. |
| Stress Testing: Back to back continuous testcases | Give back to back values for the wdata in continuous format. |
| Stress Testing: Back to back same values | Give same value back to back and check the functionality. |
| Randomization | Give randomized data to the wdata and check the output. |

# 7 Resource Requirments

**7.1 Team members and who is doing what and expertise.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Task** | **Responsible Person** | **Status** | **Comments** |
| 1 | **Design and implementation** | Pooja & Rafath | In Progress |  |
| 2 | **Basic Testbench** | Nandini & Divya | In Progress | The testbench plan has been written and testcases |
| 3 | **Final academic paper** | Divya, Nandini, Pooja & Rafath | In Progress | Final Report of the Project. |
| 4 | **Design specification document** | Pooja & Rafath | Completed | Depth calculation modules understanding, and plan has been made for the design and modules has been splitted. |
| 5 | **Verification plan document** | Divya & Nandini | Completed | Gives the overview of the verification test plan. |
| 6 | **The sunburst paper review** | Divya, Nandini, Pooja & Rafath | Completed | The sunburst paper has been reviewed and design implementation method has been decided accordingly. |
| 7 | **Power point slides** | Divya, Nandini, Pooja & Rafath | In Progress | Preparing the slides for the presentation. |

# 8 References

1. S. Cummings, "FIFOs: Fast, predictable, and deep," in Proceedings of SNUG, 2002. [Online]. Available:

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

2. S. Cummings, "FIFOs: Fast, predictable, and deep (Part II)," in Proceedings of SNUG, 2002. [Online]. Available:

<http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdf>.

3. Putta Satish, "FIFO Depth Calculation Made Easy," [Online]. Available: <https://hardwaregeeksblog.files.wordpress.com/2016/12/fifodepthcalculationmadeeasy2.pdf>.

4. A. Author et al., "Title of the Paper," in Proceedings of the Conference, 2015, pp. 123-456. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/7237325>.

5. M. Last Name et al., "Designing Asynchronous FIFO," [Online]. Available: <https://d1wqtxts1xzle7.cloudfront.net/56108360/EC109-libre.pdf>.

6. A. Author et al., "Title of the Paper," in Proceedings of the Conference, 2011, pp. 789-012. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6041338>

7. Author, "Title of the Video," [Online]. Available: <https://www.youtube.com/watch?v=UNoCDY3pFh0>

8. Open AI, "Chat GPT," [Online].