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

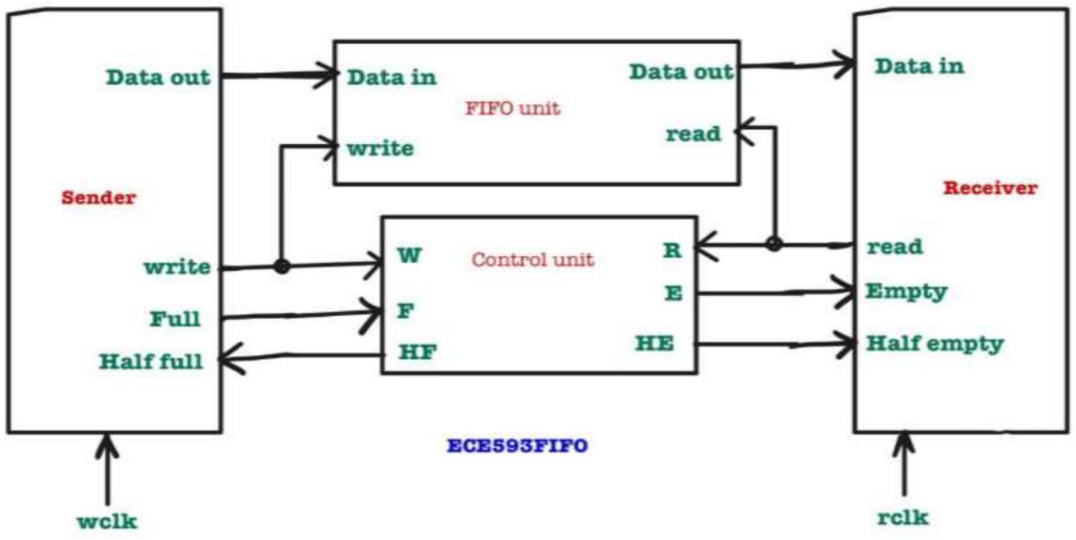
1.1 Objective of the Verification Plan

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.

1.2 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 (Asynch_fifo_interface):

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

Specifications:

w_inc: Write increment signal.

w_clk: Write clock signal.

w_addr: Address for write.

w_rst: Write reset.

r_inc: Read increment signal.

r_addr: Address for read.

r_half_empty: Read half empty signal.

w_half_full: Write half full signal.

w_data: Write data.

r_clk: Read clock.

r_rst: Read reset.

r_data: Read data

w_full: Output signal indicating FIFO full condition.

r_empty: Output signal indicating FIFO empty condition.

Memory Module (fifo_mem):

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

Specifications:

w_inc: Write increment signal.

w_full: Input signal indicating FIFO full condition.

w_clk: Write Clock signal.

w_addr: Write address.

r_addr: Read address.

w_data: Write data.

r_data: Read data.

Read Pointer Empty Module (rptr_empty):

Definition: The module handles read pointer and empty condition.

Specifications:

r_inc: Read increment signal.

r_clk: Read clock signal.

r_rst: Active-low read reset signal.

r2w_ptr: Read to write pointer.

r_empty: Read empty.

r_addr: Read address.

r_ptr: Read pointer.

Write Pointer Full Module Interface (wptr_full):

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

Specifications:

w_inc: Write increment signal.

w_clk: Write clock signal.

w_rst: Write reset signal.

w2r_ptr: Write to read pointer.

w_full: Output signal indicating FIFO full condition.

w_ptr: write pointers.

Synchronization Modules (Synchronizer):

Definition: Modules ensuring synchronization between read and write pointers.

Specifications:

w_clk: Write clock signals.

w2r_ptr: Write to read pointer.

r2w_ptr: Read to write pointer.

w_rst: Write reset signal.

w_ptr: write pointers.

r_ptr: Read pointer.

r_rst: Active-low read reset signal.

Half Full Module (half_full):

Definition: The modules handles write pointer and half full condition.

Specifications:

r_ptr: Read pointer.

w_ptr: Write pointer.

w_half_full: Write half full signal.

Half Empty Module (half_empty):

Definition: The modules handles write pointer and half empty condition.

Specifications:

r_ptr: Read pointer.

w_ptr: Write pointer.

r_half_empty: Read half empty signal.

3 Required Tools

3.1 Software and Hardware Tools

Questa sim.

3.2 Version control:

Git hub: https://github.com/DivyasriAyluri/team_11_Async_FIFO

4 Functions to be verified

4.1 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.

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.

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.

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.

5 Tests and Methods

5.1 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.

5.2 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.

5.3 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.
r_inc check 0,1	• Generate transactions with different values for rinc. • Ensure that each bin for rinc_bin is hit.
w_inc check 0,1	• Generate transactions with different values for winc. • Ensure that each bin for rinc_winc is hit.
w_rst	• Generate transactions with different values for wrst_n. • Ensure that each bin for rinc_wrst_n is hit.
r_rst	• Generate transactions with different values for rrst_n. • Ensure that each bin for rinc_rrst_n is hit.
w_full	• Generate transactions to fill the FIFO. • Generate transactions to empty the FIFO. • Ensure that both bins for wfull_bin are hit.
r_empty	• Generate transactions to fill the FIFO. • Generate transactions to empty the FIFO. • Ensure that both bins for rempty_bin are hit.

6 Resource Requirments

6.1 Team members and who is doing what and expertise.

No	Task	Responsible Person	Status	Comments
1	**Design and implementation**	Pooja & Rafath	Completed	Reviewed the sunburst papers and implemented the design accordingly.

2	**Basic Testbench**	Nandini & Divya	Completed	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.

7 Schedule

- **Week 1**(04/16/24 – 04/24/24): The sunburst paper has been reviewed and design implementation method has been decided. Design specification and verification plan documents are written. Implemented design and verified it with testbench.
- **Week 2:** Develop Class based testbench and testbench components. Verify it with randomized burst of data. Update verification plan document according to the changes made.
- **Week3:** Completed Class based testbench. Create code-coverage and functional coverage documents.

- **Week4:** Develop UVM testbench. Utilize UVM_MESSAGING and UVM_LOGGING mechanisms to create and log the reports and data.
- **Week5:** Complete the UVM architecture, UVM environment and UVM testbench. Create a few scenarios of bug injection and verify it. Finalize documents, paper and presentations.

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].