

VERIFICATION TEST PLAN

Fundamentals of Pre-Silicon Validation
Winter -2024

Project Name: Asynchronous FIFO

Members: Daniyal Ahmad, Fardeen Wasey,
Adeel Ahmed

Date: 02/11/2024

Acknowledgement:

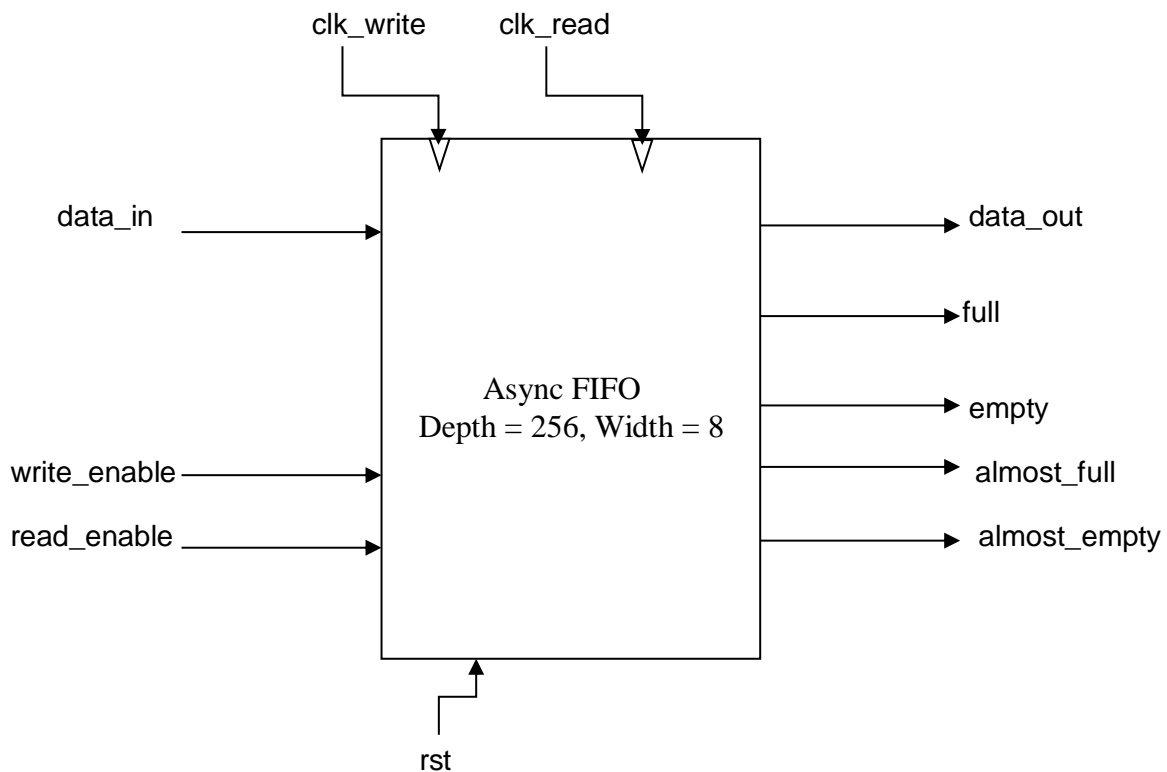
1 Table of Contents

2	Introduction:	4
2.1	Objective of the verification plan.....	Error! Bookmark not defined.
2.2	Top Level block diagram.....	Error! Bookmark not defined.
2.3	Specifications for the design.....	Error! Bookmark not defined.
3	Verification Requirements	5
3.1	Verification Levels.....	Error! Bookmark not defined.
3.1.1	What hierarchy level are you verifying and why?	Error! Bookmark not defined.
3.1.2	How is the controllability and observability at the level you are verifying?	Error! Bookmark not defined.
3.1.3	Are the interfaces and specifications clearly defined at the level you are verifying. List them.	5
4	Required Tools.....	6
4.1	List of required software and hardware toolsets needed.	6
4.2	Directory structure of your runs, what computer resources you will be using. ..	Error! Bookmark not defined.
5	Risks and Dependencies.....	6
5.1	List all the critical threats or any known risks. List contingency and mitigation plans.	6
6	Functions to be Verified.....	6
6.1	Functions from specification and implementation.....	Error! Bookmark not defined.
6.1.1	List of functions that will be verified. Description of each function	Error! Bookmark not defined.
6.1.2	List of functions that will not be verified. Description of each function and why it will not be verified.	6
6.1.3	List of critical functions and non-critical functions for tapeout.....	7
7	Tests and Methods	8
7.1.1	Testing methods to be used: Black/White/Gray Box.	8
7.1.2	State the PROs and CONs for each and why you selected the method for this DUV....	Error! Bookmark not defined.
7.1.3	Testbench Architecture; Component used (list and describe Drivers, Monitors, scoreboards, checkers etc.).....	Error! Bookmark not defined.

7.1.4	Verification Strategy: (Dynamic Simulation, Formal Simulation, Emulation etc.) Describe why you chose the strategy.	9
7.1.5	What is your driving methodology?	9
7.1.6	What will be your checking methodology?	10
7.1.7	Testcase Scenarios (Matrix)	10
8	Coverage Requirements.....	11
8.1.2	Assertions.....	11
9	Resources requirements	11
9.1	Team members and who is doing what and expertise Error! Bookmark not defined.	
10	Schedule	12
10.1	Create a table with plan of completion. You can use the milestones as a guide to fill this	12
11	References Uses / Citations/Acknowledgements.....	12

2 Introduction:

- 2.1 This document describes hardware architecture and Implementation off the FIFO block. It basically to understand and validate the internal hardware implementation of the block, along with its configuration, integration and use within FIFO. It also provides guidance and information on implementation specific issues for functional verification as appropriate.



- 2.2 A FIFO (First In First Out) serves as a buffer for data transfer between two clock domains operating asynchronously or within the same clock domain but with varying throughputs. FIFOs are versatile solutions suitable for various applications. Different types of FIFOs exist, with synchronous and asynchronous FIFOs being particularly noteworthy. The choice between them hinges on the specific application. Synchronous FIFOs are generally preferred because they provide fully synchronized flags, unlike asynchronous FIFOs. Various architectures can be employed to design FIFOs, but the most widely accepted and reliable one is the static memory-based FIFO. In contrast, fall-through FIFOs lack benefits and have notable drawbacks compared to FIFOs with static memory.

3 Verification Requirements

- 3.1 There are basically different levels of verification.

- 3.1.1 The verification is High level Verification. In which we've made a connection between sequence and driver to generate burst and driving them through driver is tested in this milestone. Also, we're printing statements in our driver whenever we generate a sequence_item from a sequence it is printing burst details.

- 3.1.2 Interfacing signals:

For this version, we are using interfacing signals shown below: -

```
Interface fifo_if(input clk_write, clk_read);
```

```
logic rst;
```

```
logic write_enable, read_enable;
```

```
logic full, empty;
```

```
logic [7:0]data_in;
```

```
logic [7:0]data_out;
```

```
logic almost_empty;
```

logic almost_full;

4 Required Tools

- 4.1 For this Verification process we are using Questa-sim to check the functionality of the design.
- 4.2 We are using VSCode which is mapped to our git repository, from that we can push and pull changes made by the team members.

5 Risks and Dependencies // will be covered in coming milestones.

- 5.1 List all the critical threats or any known risks. List contingency and mitigation plans.

Integration issues with external blocks

Risks: - A FIFO Block may not integrate smoothly with other system components, leading to unexpected behavior or performance issues.

Mitigation: - Begin developing integration tests at the early stages of the design process. Utilize mock components to mimic interaction and detect potential integration problems ahead of time.

Contingency: - Should integration challenges arise, dedicate resources to focused debugging efforts and contemplate adjusting the interface logic to enhance compatibility.

Note: - Rest all Risks, mitigation and contingency will be covered in coming milestones like incomplete coverage etc.

6 Functions to be Verified. // will be covered in coming milestones.

- 6.1 In this milestone we've made a transaction between Sequence and Driver to generate a burst from sequence.
- 6.1.1 List of functions that will not be verified. Description of each function and why it will not be verified.

- ***Enhanced Diagnostic Capabilities:***

Overview: These are on-chip diagnostic tools designed for in-depth error investigation, not for use in regular operations.

Reason for exclusion: Although beneficial for diagnostics in the field or during manufacturing evaluations, these capabilities might not affect the main operations of the FIFO. Therefore, the verification of these features could be deferred until the completion of the primary functional and integration testing.

- ***Full range of operational Corner cases:***

Overview: These involve extreme scenarios unlikely to be encountered during regular usage, for instance, swift transitions between completely full and empty states under different temperature and voltage conditions.

Reason for exclusion: Due to limitations in resources and time, verification efforts are likely to prioritize scenarios that are more likely to occur. While valuable, testing for these extreme edge cases may only be conducted if they are considered crucial following a risk evaluation.

6.1.2 List of critical functions and non-critical functions for tapeout

Critical Functions:-

- Data integrity.
- Clock domain crossing.
- Reset behavior.
- W/R synchronization.

Non- Critical Functions: -

- Almost Full/Almost Empty indicators.
- Power management features.
- Diagnostic test interface.

Strategies for tape out: -

- Focusing on Critical Functions.
- Resource allocation.

- Risk assessment for non-critical functions.

7 Tests and Methods

7.1.1 Testing methods to be used: White Box

Black Box

Pros: Function-focused, straightforward, and unaffected by internal details.

Cons: May not detect some design faults; limited coverage of internal pathways and issues.

White Box:

Pros: Optimal test cases, early identification of internal problems, and comprehensive code coverage.

Cons: Time-consuming and requiring in-depth knowledge of internal implementation.

Gray Box:

Pros: more thorough than black-box, balanced approach, partly internal knowledge used.

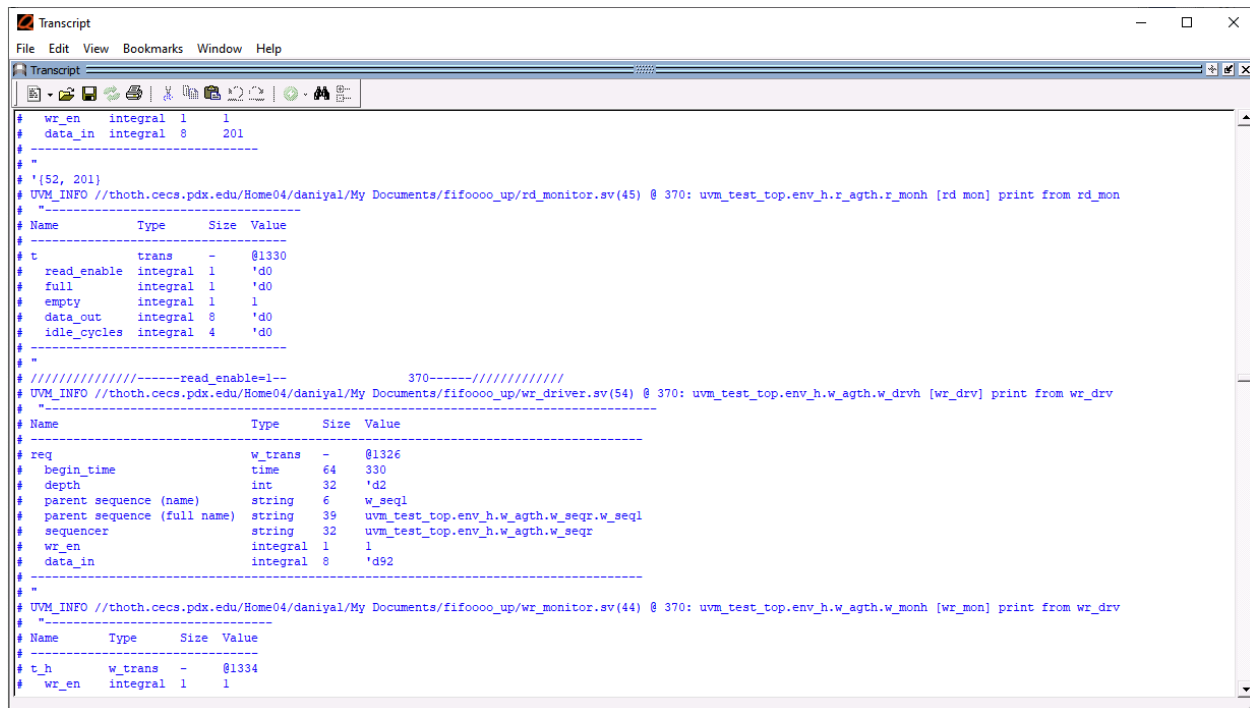
Cons: It might not obtain complete code coverage and depends on the correctness and availability of incomplete information.

We simply chose white box because we have in-depth knowledge of the internal implementation and wish to guarantee complete code coverage.

7.1.2 To test the entire functionality of driver and sequence, we've created a UVM environment with few test scenarios like:-

1. **Read enable is high, write enable low and checking status of full, empty and dataout signals.**
2. **Read enable is high, write enable is high.**
3. **Read enable is high, write enable is low.**
4. **Reset condition.**
5. **Full/empty condition.**
6. **Almost Full/empty condition.**

NOTE: This snippet is just Portion of Output and not the entire output. To see the entire transactions, you must simulate the design in Questa, and you must compile only package.sv, top.sv and the design which is fifo_if.sv as we've included everything in the package so it will compile each file parallelly and then simulate it. You will be able to see the entire random transactions.



```

Transcript
File Edit View Bookmarks Window Help
Transcript
# wr_en integral 1 1
# data_in integral 8 201
#
#
# {52, 201}
# UVM_INFO //thoth.cecs.pdx.edu/Home04/daniyal/My Documents/fifo0000_up/rd_monitor.sv(45) @ 370: uvm_test_top.env_h.r_agth.r_monh [rd_mon] print from rd_mon
#
#-----
# Name      Type      Size  Value
#-----
# t         trans     -      @1330
# read_enable integral 1  'd0
# full      integral 1  'd0
# empty     integral 1  1
# data_out  integral 8  'd0
# idle_cycles integral 4  'd0
#-----
#
#
# //-----read_enable=1----- 370-----//
# UVM_INFO //thoth.cecs.pdx.edu/Home04/daniyal/My Documents/fifo0000_up/wr_driver.sv(54) @ 370: uvm_test_top.env_h.w_agth.w_drvh [wr_drv] print from wr_drv
#
#-----
# Name      Type      Size  Value
#-----
# req       w_trans    -      @1326
# begin_time time        64      330
# depth     int         32      'd2
# parent sequence (name) string    6      w_seq1
# parent sequence (full name) string    39      uvm_test_top.env_h.w_agth.w_seqr.w_seq1
# sequencer string      32      uvm_test_top.env_h.w_agth.w_seqr
# wr_en     integral 1  1
# data_in   integral 8  'd92
#-----
#
# UVM_INFO //thoth.cecs.pdx.edu/Home04/daniyal/My Documents/fifo0000_up/wr_monitor.sv(44) @ 370: uvm_test_top.env_h.w_agth.w_monh [wr_mon] print from wr_mon
#
#-----
# Name      Type      Size  Value
#-----
# t_h       w_trans    -      @1334
# wr_en     integral 1  1

```

7.1.4 Verification Strategy: (Dynamic Simulation, Formal Simulation, Emulation etc.) Describe why you chose the strategy.

We've used Dynamic Simulation strategy.

7.1.5 What is your driving methodology?

We've used UVM methodology, and mechanism we've implemented to ensure that the sequence items (representing bursts of transactions) are accurately transmitted to the DUT and correctly logged for which we've made a connection between Sequence and Driver for burst Transaction.

7.1.6 What will be your checking methodology?

Our checking methodology is designed to ensure the FIFO's response to sequence bursts is as expected, focusing on the integrity and order of data, as well as the FIFO's control signals (e.g. full_empty, almost_full, almost_empty signals).

Scoreboard: - we've implemented a scoreboard in our testbench. Which basically compares the expected outcomes (based on the sequence items sent to the FIFO) against the actual outputs observed by the monitor.

Data Integrity and order:- The scoreboard checks for data integrity and orders, ensuring that each item in a bursts sequence is correctly stored and retrieved from the FIFO.

7.1.6.1 From specification, from implementation, from context, from architecture etc

7.1.7 Testcase Scenarios (Matrix)

7.1.7.1 Basic Tests

Test Name / Number	Test Description/ Features
1.1.1Read_and_write_test	This test will check basic read and write operation.
1.1.2Reset_test	This test will checks for reset.
1.1.3Basic_seq_transaction_test	This test will validate basic sequence to driver communication and correct FIFO operation for simple write and read transactions. Ensure data integrity and correct signal processing.

7.1.7.2 Complex Tests

Test Name / Number	Test Description/ Features
1.2.1Concurrent_R/W_transaction_test	Concurrent events (R+W) Conditions: fifo_full/fifo_empty/always_full/always empty etc.
1.2.2 randomized_transaction_test	This test will Assess the resilience of the FIFO and driver during stress situations by conducting randomized sequence transactions, which include testing of boundary conditions and implementing random resets.

7.1.7.3 Regression Tests (Must pass every time)

Test Name / Number	Test Description/Features
1.3.1 Clock_domain_crossing_test	This test will check the clock domain by focusing on data integrity across clock borders, ensure proper data transmission between various clock domains through driver-sequence interactions.
1.3.2	

7.1.7.4 Any special or corner cases testcases

Test Name / Number	Test Description
1.4.1 data_integrity_check_test	To guarantee strong FIFO functioning, carry out extensive data integrity checks for scenarios requiring complicated sequence transactions, encompassing a range of data patterns and operating circumstances.
1.4.2	

8 Coverage Requirements // will be covered in coming milestones.

8.1.1.1 Describe Code and Functional Coverage goals for the DUV

8.1.1.2 Formulate conditions of how you will achieve the goals. Explain the Covergroups and Coverpoints and your selection of bins.

8.1.2 Assertions

8.1.2.1 Describe the assertions that you are planning to use and how it will help you improve the overall coverage and functional aspects of the design.

9 Resources requirements

9.1 Daniyal Ahmad: - Worked on module top, package, sequence_items, agents and driver, agent and scoreboard for the checking strategies.

Fardeen Wasey: - Worked on monitor, env, sequencer, sequence and have made connections between driver and sequence.

Adeel Ahmed: - Worked on RTL and modifies it according to the specs also debugged the entire tb.

10 Schedule

10.1 Create a table with a plan of completion. You can use milestones as a guide to

Milestone	Task Description	Deadline
Milestone 1	In this milestone we've tested a basic functionality of a FIFO.	03/02/2024
Milestone 2	In this milestone we've developed a testbench environment in UVM and the whole infrastructure, to ensure the proper communication between the sequence and driver, to generate bursts from the sequence and drive them.	11/02/2024
Coverage Analysis	Assess code and functional coverage reports	DD/MM/YYYY
Debugging stage	Debus TB and collaborating with RTL Designer.	DD/MM/YYYY
Verification Report	Final report with results	DD/MM/YYYY
Sign-off	Final check with DV Manager and sign-off	DD/MM/YYYY

11 References Uses / Citations/Acknowledgements

1. http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdfLinks to an external site.
2. http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdfLinks to an external site.
3. <https://hardwaregeeksblog.files.wordpress.com/2016/12/fifodepthcalculationmadeeasy2.pdf>Links to an external site.
4. <https://github.com/raysalemi/uvmprimer>Links to an external site.
5. <https://www.chipverify.com/verification/constraint-random-verification>
6. <https://www.chipverify.com/tutorials/uvm>

NOTE: This document is in its basic version and will be updated accordingly at each milestone and will be made as a proper report.