

Assignment 2

Synchronous FIFO Verification using Layered Testbench

Version 1.1

Information contained within this document is for the sole readership of the recipient, without authorization of distribution to individuals and / or corporations without prior notification and approval.

Document History

The changes and versions of the document are outlined below:

Version	State / Changes	Date	Author
1.0	Initial Draft	August, 2024	Qamar Moavia
1.1	Modifications	Feb, 2025	Qamar Moavia

Table of Contents

Objectives	4
Tools	4
Instructions for Assignment	4
DUT Specification (Synchronous FIFO)	5
Introduction	5
Key Features	5
Functional Description	5
Interface Signals	6
Operations	6
Reset Operation	6
Write Operation	6
Read Operation	6
Flag Descriptions	7
Full Flag (full)	7
Empty Flag (empty)	7
Configurable Parameters	7
Verification Tasks	8
Task1: Creating Verification Plan for FIFO	8
Task 2: FIFO Verification	8

Objectives

This assignment will enable you to,

- Apply the concepts learned through labs on a more complex fifo design.
- Understand DUT behaviour through the specification document
- Write Verification plan of any DUT based on its specification document.
- Setup a layered testbench for verification of any Design.
- Write Diverse test cases to verify all the features of the design mentioned in the Specification Document.

Tools

- SystemVerilog
- Synopsys VCS

Instructions for Assignment

The required files for this assignment can be found on the

```
share_folder/CX-301-DesignVerification/Assignments/A2
```

Submit Verification plan and all the layered testbench files. Make sure all the components and classes are in separate files.

DUT Specification (Synchronous FIFO)

Read the specification first and then follow the instructions to fully verify the provided FIFO design.

Introduction

A **Synchronous FIFO** (First-In, First-Out) memory is a type of buffer used for temporary data storage while transferring data between two systems or processes operating at the same clock frequency. This document specifies the design, functionality, and verification process for a synchronous FIFO design.

Key Features

- **Data Width:** Configurable (8-bit, 16-bit, 32-bit, etc.)
- **FIFO Depth:** Configurable (16, 32, 64, 128, etc.)
- **Synchronous Operation:** All operations are synchronized with the clock signal.
- **Flags:** Full, Empty.
- **Reset:** Asynchronous reset to initialize the FIFO.

Functional Description

The FIFO memory is structured as a circular buffer consisting of multiple registers. Data is written to the FIFO via the write interface and read through the read interface. All operations are synchronized with the rising edge of the clock signal.

Interface Signals

Signal Name	Direction	Width	Description
clk	Input	1	Clock signal
rst_n	Input	1	Asynchronous reset (active low)
wr_en	Input	1	Write enable
rd_en	Input	1	Read enable
data_in	Input	DATA_WIDTH	Data input is parametrized
data_out	Output	DATA_WIDTH	Data output is parametrized
full	Output	1	Full flag (indicates FIFO is full)
empty	Output	1	Empty flag (indicates FIFO is empty)

Operations

Reset Operation

FIFO is asynchronously reset when **rst_n** transitions from high to low. After reset, the FIFO is empty, indicated by **empty** flag being high. **data_out** can hold any value post-reset.

Write Operation

Data is written into the FIFO on the rising edge of the clock when **wr_en** is asserted high and the FIFO is not full (full is low).

Read Operation

Data is read from the FIFO on the rising edge of the clock when **rd_en** is asserted high and the FIFO is not empty (empty is low).

Note: Both Write and Read Operation can be performed on FIFO at the same time.

Flag Descriptions

Full Flag (full)

The full flag is asserted (set to high) when the FIFO is full. Once this flag is set, FIFO write operation will be disabled. The flag will only be deserialized (set to low) when data is read from the FIFO through fifo read operation or the reset operation occurs.

Empty Flag (empty)

The empty flag is asserted (set to high) when the FIFO is empty or if the reset operation is performed on the FIFO. Once this flag is set, FIFO read operation will be disabled. The flag will only be deserialized (set to low) only when data is written to the FIFO through fifo write operation.

Configurable Parameters

Parameter	Description	Default Value
DATA_WIDTH	Width of the data bus	8
DEPTH	Depth of the FIFO	8

Verification Tasks

Task1: Creating Verification Plan for FIFO

1. Understand the FIFO functionality first through specifications mentioned above.
2. Using [this](#) verification plan as a template, create a detailed verification plan by stating objectives, listing features to verify, outlining specific test cases, and specifying whether the stimulus will be random or directed.

Example Test Cases for Full Flag

- Fill the FIFO to its maximum capacity and check if the full flag is asserted.
- Add one more item to the full FIFO and ensure no data is written and the full flag remains asserted.
- Read one item from the full FIFO and verify the full flag is de-asserted.

Stimulus: Directed control signals(`rst_n`, `wr_en`, `rd_en`) with random data.

3. Add coverage goals in the verification plan by defining what needs to be covered (e.g., all possible states, transitions, boundary conditions), create coverage bins by categorizing different scenarios and conditions that need to be tested.

Note: Your coverage goals should be defined based on the FIFO design specification, not the RTL implementation. This ensures that your testbench verifies compliance with the specification rather than being written for a specific RTL implementation. Your testbench should detect and fail any FIFO design that does not adhere to this specification document. We will validate your testbench against multiple FIFO designs, each with different types of errors. Your testbench must be capable of identifying these bugs.

Task 2: FIFO Verification

Set up the Environment

1. Follow verification plan to create a Layered Testbench Environment to Verify the FIFO Design.

Test fifo with different test cases

2. Execute the verification plan by creating test cases mentioned in the plan. As your environment is already set, you will only be modifying the Generator code in this subtask.
3. Run the simulation with the provided design file **fifo.sv**.
4. Debug as needed until you are happy that design is verified.
5. In case of any error, clearly mention that in the comment section of the verification plan. Additionally, attach the waveform and provide a detailed error description.
6. Finally add the Complete Coverage report in the verification plan.

Good Luck 😊