CHIPXPRT

Module ID : CX-301
Design Verification
Instructor : Dr. Abid Rafique

Assignment 1

Synchronous FIFO Verification

Version 1.1

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

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 SystemVerilog Tasks and Functions for reusability and modularity.
- 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

```
shared_folder/CX-301-DesignVerification/Assignments/A1
```

```
./A2/
      ├── top.sv
      ├── fifo.sv
      ├── fifo_test.sv
      ├── fifo_interface.sv
```

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 deserted (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 deserted (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 FIFO Interface

1. Create SystemVerilog Interface in the file `fifo_interface.sv`.
2. Write SystemVerilog Tasks to access interface signals for various operations, including but not limited to `fifo_write` and fifo_read operations.
3. Make sure you follow the timing correctly for driving the inputs to the DUT or capturing the outputs of the DUT.

Task 2: Test FIFO with Diverse Test Cases

1. In the fifo_test.sv create a `fifo_test` module and write test cases to verify the fifo functionality.
2. Note that you have to test data correctness as well as the correctness of the `empty` and `full` flags.
3. Verify the FIFO with at least three different parameter configurations (e.g., different DEPTH and DATA_WIDTH values).
4. Run the simulation with the provided design file **fifo.sv**.
5. Debug as needed until you are happy that design is verified.

Good Luck 🙂