Final Project: ECE593, Spring 2024

Implementation and Verification of Asynchronous FIFO

DUT Specifications:

FIFO are often used to safely pass data from one clock domain to another asynchronous clock domain. An asynchronous FIFO refers to FIFO design where data values are written to a FIFO buffer from one clock domain and the data value are read from the same FIFO buffer from another clock domain, where two clock domains are asynchronous to each other.

The write pointer always points to the next word to be written. As soon as the first data is written to the FIFO, the write pointer increments, the empty flag is cleared. The read pointer always points to the current FIFO word to be read.

The FIFO is empty when the read and write pointers are both equal.

FIFO Depth Calculation:

Double the frequency of write and half of the read

Sender Clock Frequency = 120MHz

Number of idle cycles between two successive writes = 3

Receiver Clock Frequency = 50MHz

Number of idle cycles between two successive reads = 2

Write Burst = 1024

Given, Sender Clock Frequency < Receiver Clock Frequency

The number of idle cycles between two successive writes is 3 clock cycles, which means after writing one data, write module is waiting for 3 clock cycles to initiate the next write, meaning every **4 clock cycles** one data is written.

The number of idle cycles between two successive reads is 2 clock cycles, which means after reading one data, read module is waiting for 3 clock cycles to initiate next read, meaning every **3 clock cycles** on data is read.

Time required to write one data item = 4 * (1/120MHz) = 33.33 ns.

Time required to write the data in the burst = 34,129.92 = 34,129ns.

Time required to read one data item = 3 * (1/50MHz) = 60ns

So, for every 60ns read module is going to read one data item in the burst.

No of data items can be read in a period of 34,129ns = (34,129/60) = 568.81 = 568 items.

Remaining no of bytes to be stored in the FIFO = 1024 – 568 = 456

The minimum depth of the FIFO should be 456. Write clock timing = 33.33ns, ready clock timing = 60ns.

TESTING SCENARIOS:

1) **Basic Functionality Test:**
    - Write a single data word into the FIFO and read it back to verify basic read/write operations.
    - Check for correct data alignment, the data read should match the data written.
2) **Full Memory Write and Read Test:**
    - Fill the FIFO to capacity and verify that all data can be read out correctly.
    - Ensure 'wfull' signal is asserted when the FIFO is full.
    - Attempt to write beyond the FIFO depth and confirm no data corruption occurs.
3) **Write Increment and Read Test:**
    - Perform sequential write operations until FIFO is full.
    - Read back all the data sequentially to ensure order and data integrity.
    - Ensure 'rempty' signal is asserted when the FIFO is empty

GIT LINK : https://github.com/krishnaachyuth/AsynchronousFIFO

Roles & Responsibilities

S.No.	Task	Responsible person	Comments
1	High Level Design Specification (HLDS)	Achyuth, Sathwik, Harsha, Amrutha	Design spec documentation
2	Design spec calculation and plan	Achyuth, Sathwik, Harsha, Amrutha	Depth calculation modules understanding and plan has been made for the design and modules have been divided
3	FIFO Memory module	Achyuth	System Verilog code has been implemented for FIFO mem module
4	Read pointer Empty module	Sathwik	System Verilog code has been implemented & uploaded to GitHub.
5	Synchronous Write to Read module	Harsha	System Verilog code has been implemented & uploaded to GitHub.
6	Synchronous Read to Write module	Amrutha	System Verilog code has been implemented & uploaded to GitHub.
7	Write pointer Full module	Achyuth, Sathwik, Harsha, Amrutha	System Verilog code has been implemented & uploaded to GitHub.
8	Basic Testbench	Achyuth, Sathwik, Harsha, Amrutha	System Verilog code has been implemented & uploaded to GitHub.