FINAL PROJECT

ECE593 - Fund of Pre-Silicon Validation

(Spring 2024)

IMPLEMENTATION AND VERIFICATION OF ASYNCHRONOUS FIFO

MILESTONE - 1

By Team 8

Dileepkumar Pulluru - 923600886

Karthika Munnuri - 933279910

Navaneeth Kumar Gadde - 938717155

Sri Sai Sumanth Yadalapalli - 966493852

Design Specification:

FIFO is a buffer that stores data in a way that data stored first comes out of the buffer first. As the System on chip involves multiple IPs operating at different speeds. Generally, Asynchronous FIFO is used when the write operation is faster than the read operation. Therefore, they need to be synchronized. Otherwise, it may lead to metastability conditions. This will affect the operation of the chip. To overcome this problem Asynchronous FIFOs are used.

An asynchronous FIFO is a type of FIFO (First-In, First-Out) design commonly utilized to securely transfer data between two clock domains that operate asynchronously to each other. In this setup, data is written into a FIFO buffer from one clock domain and read from the same buffer in another clock domain.

In this design:

- The write pointer indicates the next data word to be written into the FIFO. When the first data is written, the write pointer increments, and the empty flag is cleared.

- The read pointer indicates the current word in the FIFO available for reading.

The FIFO is considered empty when both the read and write pointers are at the same position.

FIFO Calculation:

Write Clock Frequency (Sender) : 120MHZ

Read Clock Frequency (Receiver) : 60MHZ

No of write idle clock cycles = 3

No of read idle clock cycles = 2

Burst Size = 1024

Write Clock Frequency > Read Clock Frequency

No. of idle cycles between successive writes is 3 clock cycle. It means that after writing one data, the Producer block is waiting for 3 clock cycles to initiate the next write. So, it can be understood that for 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, the read module is waiting for 3 clock cycles to initiate the 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, 600 – 568 = 32

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

Verification Plan

Test cases

Test Case	Test Features
Write and Read	Write a single data word into the FIFO and read it back to verify basic read/write operations.
FIFO Memory Array check	Check Read and Write into Array at module level
FIFO Memory Full Flag Detection	Checks to see if an asserted full flag
Write Pointer increment	Write Pointer must increment
Read Pointer increment	Read Pointer must increment
Reset Operation	Reset operation from Top module must propagate through the entire design

Roles & Responsibilities

S.No.	Role	Responsible person	Comments
1	High Level Design Specification (HLDS)	Dileep, Navaneeth,Karthika,Sumanth	Design specification Documentation, implementation of read pointer empty and write pointer write full modules.
2	Depth calculations, module analysis, and plan implementation for the design.	Navaneeth	Performed depth computations. Enabling top main connections, involved in the implementation of some part of write only full module & uploaded to GitHub.
3	Data structure coding in accordance with the design specification	Sumanth	Implementation of the FIFO memory, write full modules as per the design requirements & uploaded to GitHub.
4	Enabling the design coding and implementation of initial testbench	Dileep	Implementation of synchronous read to write & write to read modules & uploaded to GitHub.
5	Implementation of the testbench pipelining. Any changes to GIT and updating the version control system of the design code.	Karthika	Implementation of testbench, involved in half part of read pointer empty and write pointer write full codes & uploaded to GitHub.