Verification of Synchronous FIFO using Cocotb

Mihir Rana
Department of Electronics and Communication Engineering
Institute of Technology Nirma University
Ahmedabad, India
mihirrana620@gmail.com

***Abstract*—Verification is essential to ensuring that the design accurately reflects its specifications and is bug-free. Verification helps to avoid surprises later so that a product can launch on schedule, with excellent quality, and at a reasonable price. When transferring data between processor blocks, First-In First-Out (FIFO) memory structures are frequently utilised as a buffer. Data transmission across modules with various, even unrelated clock frequencies is becoming an increasing requirement for high performance and high complexity digital systems. This project gives a comprehensive explanation of the driving factors and design choices that went into creating a reliable and scalable FIFO architecture. The suggested architecture makes use of an effective memory array structure and may be further altered to function in situations where there are many clock cycles of delay between the FIFO, the data producer, and the data consumer. In this paper the main objective is to design Synchronous Fifo and verify the functionality using Cocotb. The RTL code of the Synchronous Fifo is written using verilog.**

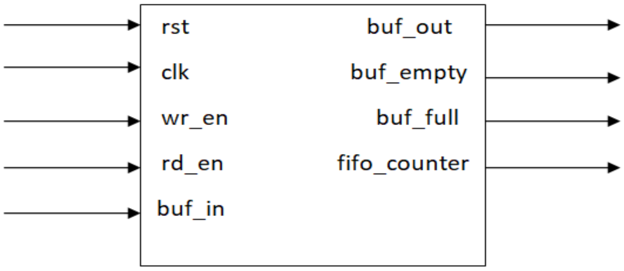
I. Description

First-in, first-out (FIFO) is a strategy used in computer programming to manage programme work requests from queues or stacks so that the oldest request is addressed first. When storing data from one clock domain and supplying it to another clock domain upon request in hardware, it is either an array of flops or a read-write memory that follows the first in first out principle. Read or output logic refers to the clock domain that reads data from the FIFO, whereas write or input logic refers to the clock domain that sends data to the FIFO.

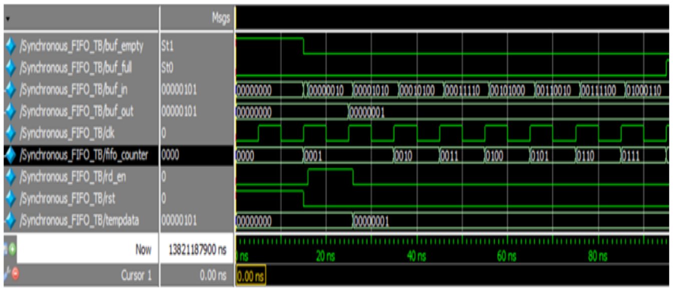
To pass multi-bit data words reliably, FIFOs are utilised in designs transferring data between source and destination sides located in the same clock domain or controlling the flow of data from one clock domain to another. The FIFO is referred to be synchronous if the read and write clock domains are controlled by the same clock signal, and ASYNCHRONOUS if the read and write clock domains are controlled by asynchronous clock signals.

Data should never be written in a full state, and data should never be read in an empty state since doing so might result in data loss or the production of irrelevant data. When referring to a FIFO design, the term ”synchronous FIFO” refers to a FIFO where data values are sequentially written into a memory array using a clock signal and sequentially retrieved out of the memory array using the same clock signal. Since there is no clock domain crossover, the production of empty and full flags in synchronous FIFO is simple. A user can even create programmed half empty and partial full flags, which are required in many applications, in light of this fact.

II. Design of Synchronous Fifo



III. Waveform



	Msgs
/Synchronous_FIFO_TB/buf_empty	St1
/Synchronous_FIFO_TB/buf_full	St0
/Synchronous_FIFO_TB/buf_in	00000101
/Synchronous_FIFO_TB/buf_out	00000101
/Synchronous_FIFO_TB/clk	0
/Synchronous_FIFO_TB/fifo_counter	0000
/Synchronous_FIFO_TB/rd_en	0
/Synchronous_FIFO_TB/rst	0
/Synchronous_FIFO_TB/tempdata	00000101
Now	13821187900 ns
Cursor 1	0.00 ns

Fig. 1. Reference Circuit

References

[1] https://github.com/mihirrana620/Synchronous-FIFO-NIELIT-IITM-CTB-Verification-Hackathon