DESIGN SPECIFIACTION REPORT

ECE-593: Fundamentals of Pre-Silicon Validation
Maseeh College of Engineering and Computer Science
Winter, 2025

Design and Verification of Asynchronous FIFO using Class Based Verification & UVM

Team 12:

Aravindh Nanjaiya Latha (PSU ID:963672501)
Rishi Gunda (PSU ID: 919656792)
Rohit Bonigala (PSU ID:952156645)
Shreya Umesh Shetty (PSU ID:926818354)

Date: 28 May 2025

GitHub Repository:

https://github.com/aravindh-nanjaiya-latha/Team12_Asynchronous-FIFO_S25_ECE593

Project Name	Design and Verification of Asynchronous FIFO using Class Based Verification & UVM
Location	Portland
Start Date	15 April 2025
Estimated Finish Date	23 May 2025
Completed Date	28 May 2025

Prepared by: Team Number 12	
Prepared for: Prof. Venkatesh Patil	
Team Member Name	Email
Aravindh Nanjaiya Latha	aravindh@pdx.edu
Rishi Gunda	rgunda@pdx.edu
Rohit Bonigala	rohit@pdx.edu
Shreya Umesh Shetty	shreyau@pdx.edu

Introduction:

An Asynchronous FIFO (First-In, First-Out) buffer addresses this problem by allowing data to be written and read using independent clock signals. Compared to Synchronous FIFOs, which are controlled by a single clock domain. Asynchronous FIFOs offer a flexible method of data transfer without data loss or corruption across systems operating at varying rates.

An asynchronous FIFO's main benefit is that it serves as a buffer, preventing overflow and facilitating seamless communication between the sender and receiver modules. This kind of complex system, which includes data integrity, synchronization, and metastability handling difficulties, is challenging to design. When signals move between asynchronous clock domains, they can cause undefinable logic states, which is known as metastability. Using strategies like synchronizers, gray-coded points, and appropriate clock domain crossover protocols are some workarounds.

Design Features:

To facilitate data transfer with overflow and underflow avoidance, the architecture has a memory buffer, write pointer, read pointer, and control logic. While the read pointer points to the present location from which data must be read, the write pointer records the location in memory where the subsequent data piece is to be written. Because FIFO operates in two distinct clock domains—the sender writing frequency is 300 MHz, and the receiver reading frequency is 150 MHz—a synchronization mechanism is necessary to prevent metastability.

When initialization or reset occurs, the FIFO is cleared since both the read and write pointers are reset or initialized to zero. Data is written to the location indicated by the write pointer during writing, and the pointer is incremented upon successful write. Likewise, after every read, the read pointer is increased. When both pointers are the same, the FIFO is maintained empty, and the empty flag is set to prevent pointless reads. The FIFO is filled and the full flag is set to stop additional writes when the write pointer finishes and comes into contact with the read pointer.

Differentiating between the full and empty states—both of which happen when the read and write pointers are equal—is one of the primary challenges in asynchronous FIFO design. In order to fix this, they give each pointer an extra bit that toggles as the FIFO turns around. The FIFO has wrapped around and is full when the write pointer's extra bit is different from the read pointer's. The FIFO is empty when the two pointers and the extra bit are equal.

The architecture also includes the logic for handling overflow and underflow scenarios. To prevent data corruption, the following write operations are skipped when the FIFO is full. To avoid reading trash data, reading operations are halted when the FIFO is empty. The read pointer is made to always refer to the subsequent valid piece of data in order to maximize the FIFO, which lowers the number of clock cycles needed to read the data.

The Asynchronous FIFO architecture mainly prioritizes efficient memory use, suitable synchronization, and robust pointer management to enable reliable data transfer over several time domains.

Specifications:

- Producer CLK frequency: 500 MHz
- Consumer CLK frequency: 450.45 MHz
- Maximum write burst size: 512
- Duty cycle: %

- No. of idle cycles between successive writes: 0 (same as original)
- No. of idle cycles between successive reads: 2 (same as original)

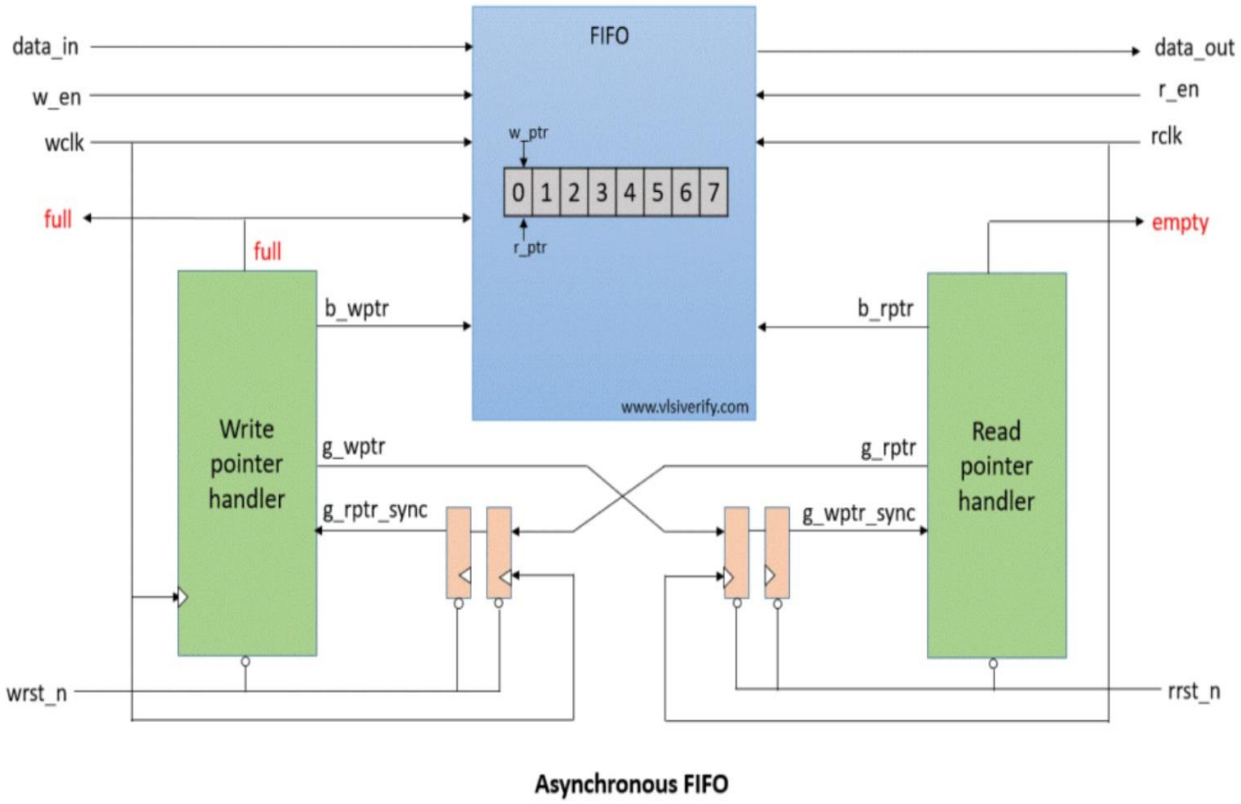
Important Signals/Flags:

Signal	Direction	Description
full	Output	Full flag
write_error	Output	Write error
read_error	Output	Read error
half_full	Output	Stack Memory Half Full
empty	Output	FIFO empty Flag

Design Signals:

Signals	Description
wclk	Write clock
wrst_n	Write reset
rclk	Read clock
rrst_n	Read reset
w_en	Write enable
r_en	Read enable
data_in	Data input for FIFO
data_out	Data output
g_wptr_sync	Gray Write pointer
g_rptr_sync	Gray Read Pointer
b_wptr	Black write pointer
b_rptr	Black read pointer
g_wptr	Gray write pointer
g_rptr	Gray read pointer

Block Diagram :



Asynchronous FIFO

References/Citations:

1. S. Cummings, "FIFOs: Fast, predictable, and deep (Part II)," in Proceedings of SNUG, 2002.

http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdf

2.ASIC-World FIFO Examples

http://www.asic-world.com/examples/verilog/fifo.html

3. VLSI Verify

https://vlsiverify.com/verilog/verilog-codes/asynchronous-fifo/