DESIGN SPEC DOCUMENT

ECE-593: Fundamentals of Pre-Silicon Validation

Maseeh College of Engineering and Computer Science

Winter, 2025

Project Name: Asynchronous FIFO Design and Verification using UVM

Members: Uma Mahesh Bestha, Thanmai Neelampalli, Sathwik Bandi, Bhanu Sai Sidhardha Tummala

Date: 01/25/2025

Project Name	Asynchronous FIFO Design and Verification using UVM
Location	Portland, Oregon
Start Date	01/18/2025
Estimated Finish Date	03/06/2025
Completed Date	

Prepared by: Team 11	
Prepared for: Prof. Venkatesh Patil	
Team Member Name	**Email**
Uma Mahesh Bestha	bestha@pdx.edu
Thanmai Neelampalli	thanmain@pdx.edu
Sathwik Bandi	sathwikb@pdx.edu
Bhanu Sai Sidhardha Tummala	bhanusai@pdx.edu

Design Features:
Clock Domain Crossing: The asynchronous FIFO facilitates data transfer between different clock domains, allowing write and read operations to occur at different frequencies. This ensures synchronization of data flow across different clock domain systems.
Distinct Operating Frequencies: The FIFO operates at a frequency of 240 MHz, while the consumer operates at a higher frequency of 400 MHz. This design allows for the production and consumption of data at different rates.
Maximum Write Burst Size: The FIFO imposes a maximum write burst size of 512, which influences how data is managed within the buffer.
Idle Cycles: The design incorporates two idle cycles between consecutive reads, while there are no idle cycles between writes. This affects the overall flow of data.
Data Transfer Timing for a write cycle: The producer writes data at each clock cycle, taking approximately 4.1667 nanoseconds to write one data item.

Data Transfer Timing for a read cycle: The consumer reads data every three clock cycles, requiring about 7.5 nanoseconds to read one data item.

Buffer Management: The calculations indicate that in a period of 2134 nanoseconds, the consumer can read 284 items, necessitating storage for 228 additional items in the FIFO.

FIFO Depth and Width: The design can accommodate a FIFO depth of either 256 or a custom size of 228, with a width of 8 bits.

Project Description:

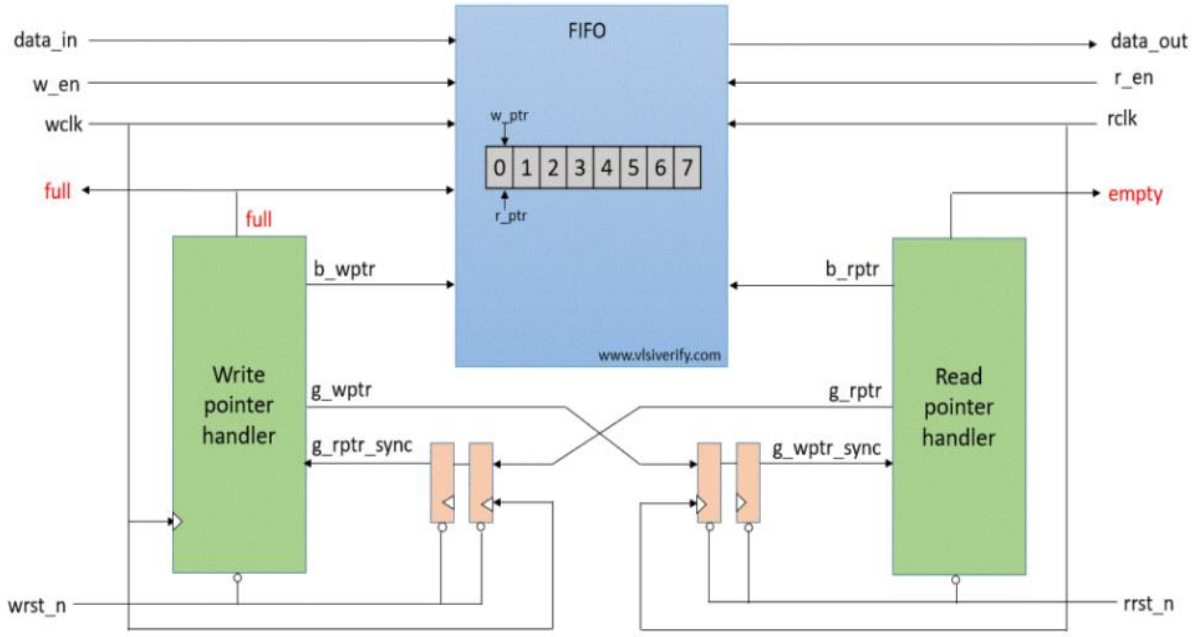
The project titled **"Asynchronous FIFO Design and Verification using UVM"** focuses on the development of an asynchronous First-In-First-Out (FIFO) buffer that facilitates data transfer between different clock domains, allowing for synchronization of data flow across systems operating at distinct frequencies. The design operates at a frequency of 240 MHz for the producer and 400 MHz for the consumer, enabling efficient communication despite differing data production and consumption rates. Key specifications include a maximum write burst size of 512, with a structured approach to data writing and reading that incorporates idle cycles to manage the buffer effectively. The calculations demonstrate the need for a FIFO depth of either 256 or a custom design of 228, with a width of 8 bits, ensuring optimal performance in handling data across varying clock signals.

Important Signals/Flags

Write Enable (WE): Indicates when data can be written to the FIFO.
Read Enable (RE): Indicates when data can be read from the FIFO.
Full Flag: Signals that the FIFO is full and cannot accept more data.
Empty Flag: Signals that the FIFO is empty and there is no data to read.
Data Valid Flag: Indicates that the data present in the FIFO is valid and can be read.
Overflow Flag: Indicates that an attempt to write data has occurred when the FIFO is full.
Underflow Flag: Indicates that an attempt to read data has occurred when the FIFO is empty.

Design Signals
Write Enable (WE): This signal indicates when data can be written into the FIFO. It is activated during the write operation. **Read Enable (RE):** This signal indicates when data can be read from the FIFO. It is activated during the read operation. **Data Input (DIN):** This is the data line where the input data is fed into the FIFO during the write operation. **Data Output (DOUT):** This is the data line where the output data is read from the FIFO during the read operation. **Full Flag (FULL):** This signal indicates whether the FIFO is full. It prevents further write operations when FIFO reaches its maximum capacity. **Empty Flag (EMPTY):** This signal indicates whether the FIFO is empty. It prevents read operations when there is no data available in the FIFO. **Clock Signals:** Since the FIFO operates across different clock domains, there will be separate clock signals for the producer and consumer. These signals synchronize the write and read operations. **Reset Signal (RESET):** This signal is used to initialize the FIFO, clearing all data and resetting the status flags.

Block Diagram
The following block diagram represents basic functionality of Asynchronous FIFO



Asynchronous FIFO

Calculations needed for DUT:

The steps included in the calculation are:

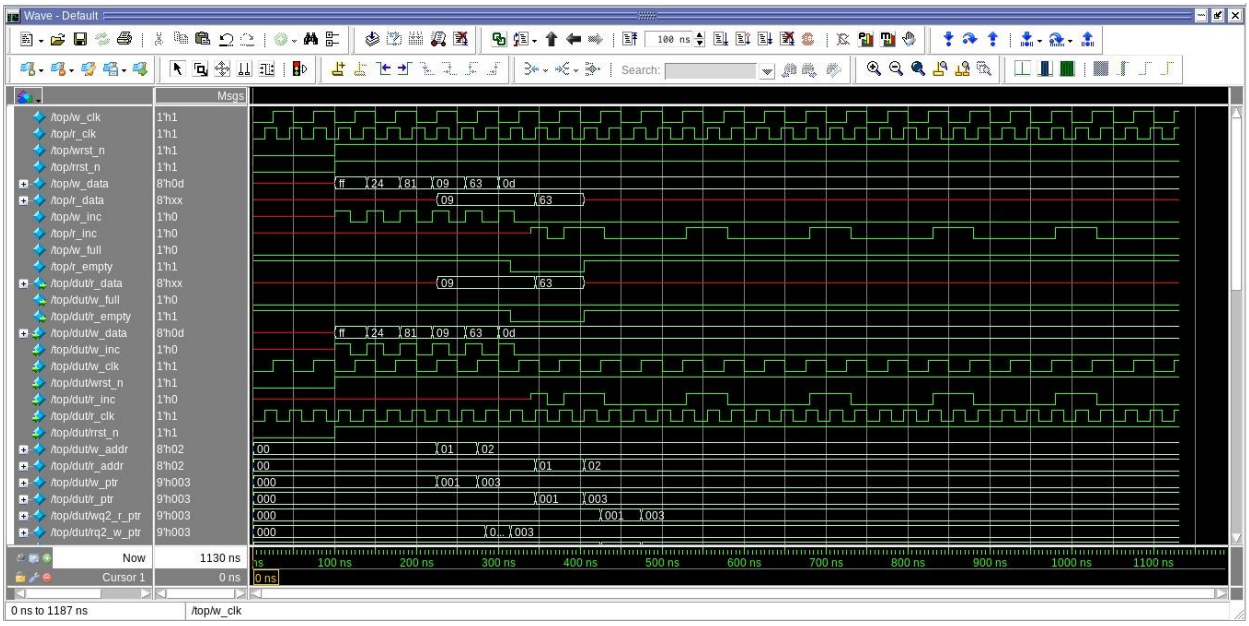
- The producer writes data at each clock cycle
- The consumer reads data every three clock cycles.
- The time needed to write one data item is 1/240 MHz = 4.1667 nanoseconds.
- The time required to write all data items is 512 * 4.1667 = 2134 nanoseconds.
- The time needed to read one data item is 3/400 MHz = 7.5 nanoseconds.
- The number of data items that can be read in a period of 2134 nanoseconds are
- 2134/7.5 = 284 items.
- Therefore, the number of data items that need to be stored is 512 – 284 = 228.

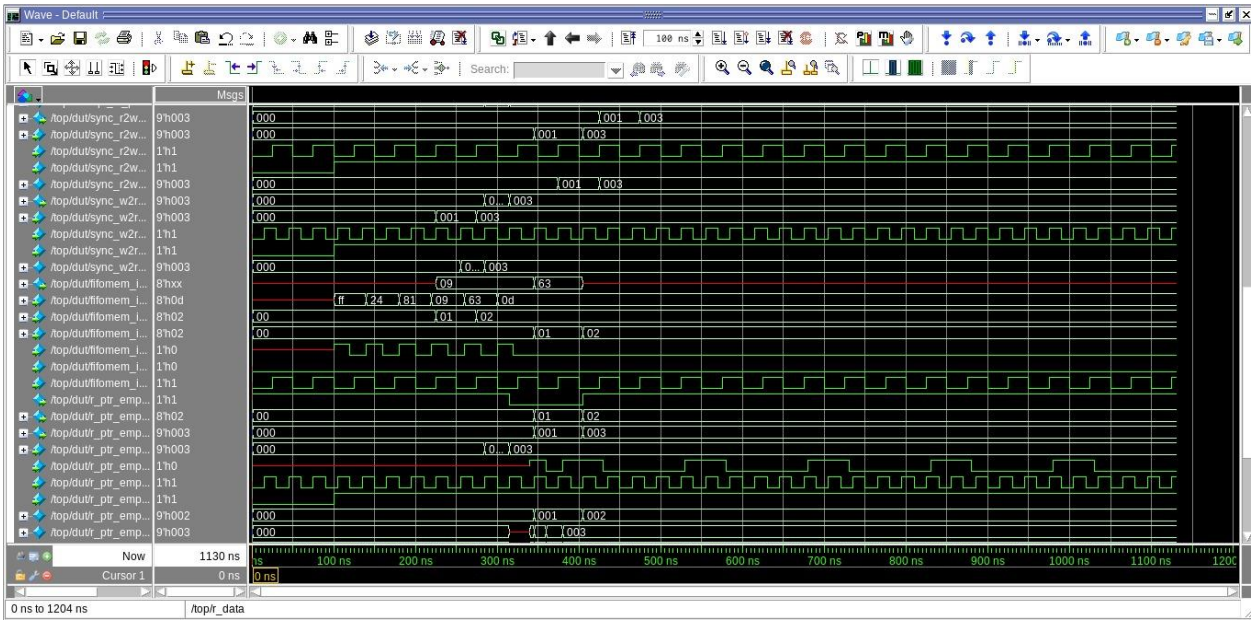
Hence, the logarithm of 228 with a base of 2 yields 8 bits.

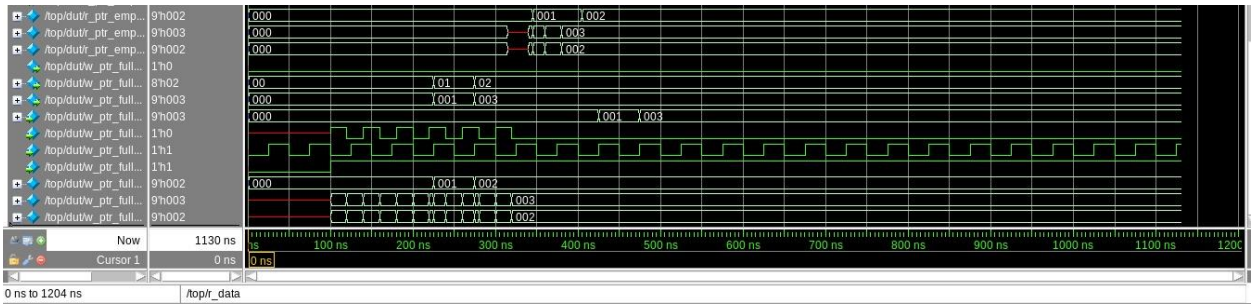
Therefore, we can opt for a FIFO with a depth of either 256 or design one with a size of 228. The width of the FIFO is 8 bits.

References/Citations:
Prof. Venkatesh Patil lecture slides of ECE 571: Introduction to SystemVerilog
Prof. Venkatesh Patil lecture slides of ECE-593: Fundamentals of Pre-Silicon Validation
https://vlsiverify.com/verilog/verilog-codes/asynchronous-fifo/
XILINX Asynchronous FIFO V3.0 Design specification document

Waveforms:




Wave - Default
Msgs
/top/dut/sync_r2w... 9'h003
/top/dut/sync_r2w... 9'h003
/top/dut/sync_r2w... 1'h1
/top/dut/sync_r2w... 1'h1
/top/dut/sync_r2w... 9'h003
/top/dut/sync_w2r... 9'h003
/top/dut/sync_w2r... 9'h003
/top/dut/sync_w2r... 1'h1
/top/dut/sync_w2r... 1'h1
/top/dut/sync_w2r... 9'h003
/top/dut/fifomem_i... 8'hxx
/top/dut/fifomem_i... 8'h0d
/top/dut/fifomem_i... 8'h02
/top/dut/fifomem_i... 8'h02
/top/dut/fifomem_i... 1'h0
/top/dut/fifomem_i... 1'h0
/top/dut/fifomem_i... 1'h1
/top/dut/r_ptr_emp... 1'h1
/top/dut/r_ptr_emp... 8'h02
/top/dut/r_ptr_emp... 9'h003
/top/dut/r_ptr_emp... 9'h003
/top/dut/r_ptr_emp... 1'h0
/top/dut/r_ptr_emp... 1'h1
/top/dut/r_ptr_emp... 1'h1
/top/dut/r_ptr_emp... 9'h002
/top/dut/r_ptr_emp... 9'h003
Now 1130 ns
Cursor 1 0 ns
100 ns 200 ns 300 ns 400 ns 500 ns 600 ns 700 ns 800 ns 900 ns 1000 ns 1100 ns
0 ns to 1204 ns
/top/r_data


/top/dut/r_ptr_emp... 9'h002
/top/dut/r_ptr_emp... 9'h003
/top/dut/r_ptr_emp... 9'h002
/top/dut/w_ptr_full... 1'h0
/top/dut/w_ptr_full... 8'h02
/top/dut/w_ptr_full... 9'h003
/top/dut/w_ptr_full... 9'h003
/top/dut/w_ptr_full... 1'h0
/top/dut/w_ptr_full... 1'h1
/top/dut/w_ptr_full... 1'h1
/top/dut/w_ptr_full... 9'h002
/top/dut/w_ptr_full... 9'h003
/top/dut/w_ptr_full... 9'h002
Now 1130 ns
Cursor 1 0 ns
100 ns 200 ns 300 ns 400 ns 500 ns 600 ns 700 ns 800 ns 900 ns 1000 ns 1100 ns
0 ns to 1204 ns
/top/r_data