Asynchronous FIFO: Design Specification

Session 1 Group 4

Nick Allmeyer, Alexander Maso, Ahliah Nordstrom

Version 1.1 - April 22, 2024

Features

- Separate modules for READ control, WRITE control, Memory, and for each clock synchronizer
- Pointer management using binary counter, *later implementation of Gray Code to come*
- Simplified binary counter to minimize synchronization issues between clock domains initially
- Sender clock frequency of 80 Mhz for rapid data transmission
- Receiver clock frequency of 50 Mhz for optimized processing capabilities
- Sender data burst size of 120 data items for efficient bulk processing
- Zero idle cycles for both reading and writing allows for maximized throughput
- High data rate with minimal latency by achieving 12.5 ns for every WRITE operation and 20 ns for every READ operation
- Maintains a 8 : 5 write to read ratio to optimize buffer utilization and prevent bottlenecks
- Minimum FIFO depth of 45
- FIFO depth, Addr width and Data width are parameterized
- Initial implementation has depth of 64, addr width of 6 (+1 for ptrs) and data width of 8

Functional Description

The asynchronous FIFO acts as a buffer to manage data flow between two parts of a system operating on different clock domains. The FIFO ensures data integrity and timing without the need for clock synchronization between sender and receiver blocks, operating at 80 Mhz and 50 Mhz respectively.

A memory buffer with a depth of at least 45 items is used to store data elements. Operations are consistent and predictable for both READ and WRITE, as they are fully synchronous to their respective clocks. Data is written to the FIFO at a rate of 12.5 ns per operation and is determined by the rising edge of the WRITE clock. Data is read at a rate of 20 ns per operation and is determined by the rising edge of the READ clock. The ratio of WRITEs to READs is 8:5, simplified down from 120:75. This ratio defines the operational balance between input and output operations so that the FIFO does not overflow/underflow under normal operating conditions.

A binary counter is used to track READ and WRITE pointers that then indicate the next positions for READ and WRITE operations. These pointers will be initialized to zero upon a system reset and then increment as data is written and read to and from the FIFO. If an invalid READ operation is performed, due to an empty FIFO, the RD_ERR signal will be asserted. If an invalid WRITE operation is performed, due to a full FIFO, the WR_ERR signal will be asserted. The HALF_EMPTY and HALF_FULL flags are used as a way to indicate nearing their respective states. The FULL flag is generated in the write-clock domain when no more data can be written to the FIFO without overwriting existing data. The EMPTY flag is generated in the read-clock domain when there are no data items to read from the FIFO and both pointers are equal.

Synchronization and Timing Issues

Synchronization of the READ and WRITE pointers between different clock domains is a common source of error. To mitigate this and the risk of metastability, each pointer is passed through a series of dual flip-flop stages and serves as a synchronization mechanism.

Behavior of Status Signals

An asynchronous reset will render the FIFO empty with no valid data. This clearing signal instantaneously resets all synchronizing registers, WCLK, FULL, HALF_FULL and RCLK but sets the EMPTY and HALF_EMPTY flag status since the FIFO is indeed empty/half empty. Although this may seem abrupt, this does not affect the FIFO since all of its contents are reinitialized. Reset signals are negated in alignment with the rising edge of their respective clock domains.

The FULL signal is activated when the FIFO buffer reaches its maximum capacity that is indicated by the WRITE pointer being just one position away from the READ pointer. This FULL signal prevents any additional data writes that could lead to data overflow. The EMPTY signal indicates that the FIFO is lacking any data and the W_PTR and R_PTR are equal. There are edge cases where the FULL and EMPTY signals could be simultaneously asserted so an additional logic bit can be used to differentiate between these states. This FULL/EMPTY bit toggles upon each wrap around of the pointers to buffer.

Removing the FULL status is intentionally delayed afterwards so that even after the R_PTR increments and the FIFO is no longer full, the logic retains the FULL asserted for two additional rising edges of the WCLK. This latency results in synchronization of the incremented R_PTR into the write domain and thus giving the FIFO sufficient time to clear space before accepting new data. Similarly, when removing the EMPTY status, the FIFO will hold the EMPTY flag high for an additional two rising edges of the RCLK following a WRITE. This synchronizes the incremented W_PTR into the read domain.

The HALF_FULL signal is asserted when the data reaches or exceeds half of the FIFOs capacity; when the count of items in the FIFO is at least half the buffer size. The HALF_EMPTY signal is asserted when the content of the FIFO falls below half.

FIFO Calculations

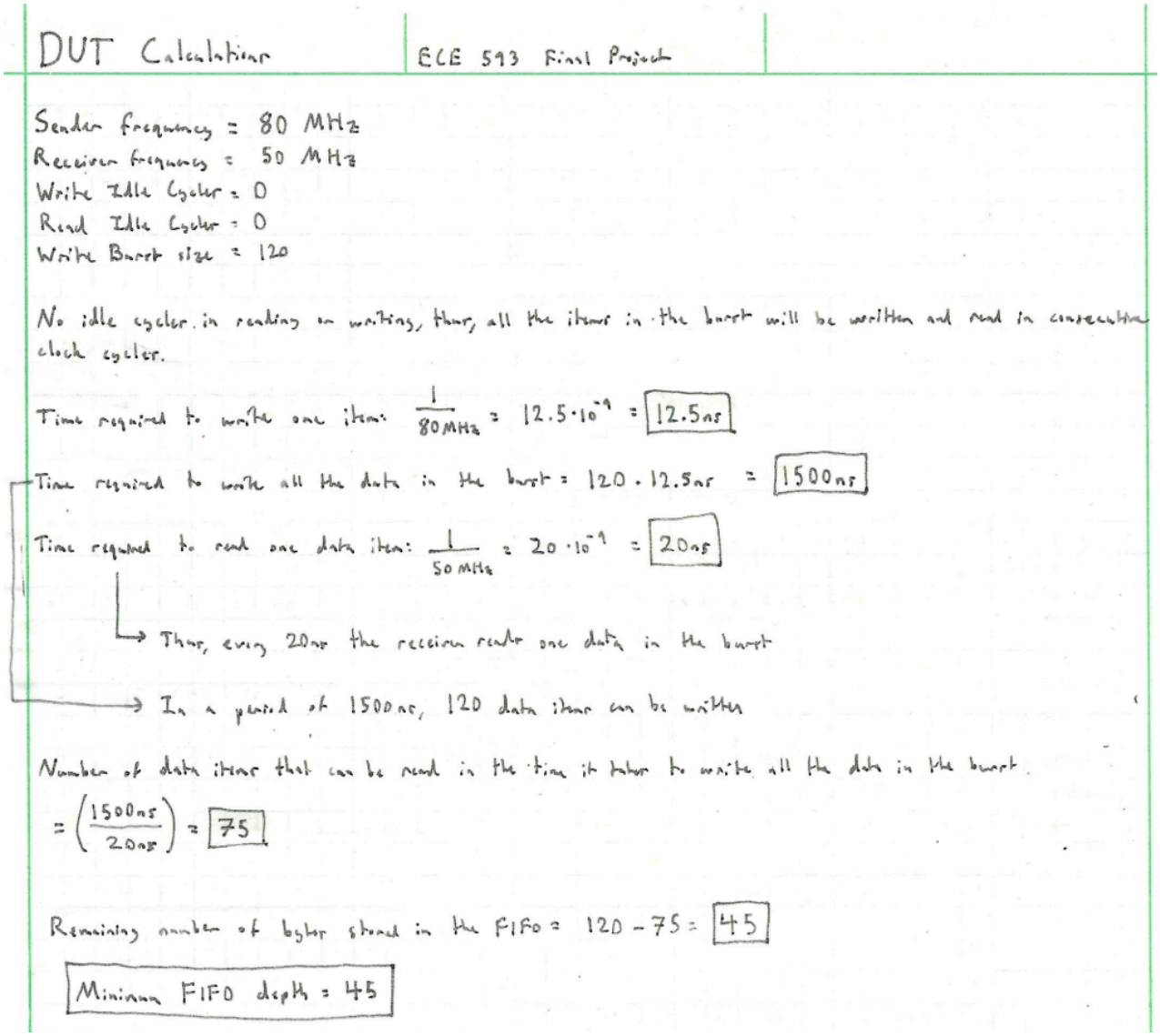

Figure 1: FIFO team-based specification calculations for asynchronous FIFO

References

[1] “Crossing clock domains with an Asynchronous FIFO,” zipcpu.com. https://zipcpu.com/blog/2018/07/06/afifo.html. [Accessed April 20, 2024]

[2] C. Cummings, “Simulation and Synthesis Techniques for Asynchronous FIFO Design,” *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers.* March 2002, Vol. 281.

[3] R. Salemi, “The UVM Primer: An Introduction to the Universal Verification Methodology”. Boston: Boston Light Press, 2013. Accessed: Apr. 1, 2024. [Online].

[3] B. Wile and J. Gross and W. Roesner, “Comprehensive Functional Verification: The Complete Industry Cycle”. San Francisco, CA, USA: Morgan Kaufmann, 2005. Accessed: Apr. 1, 2024. [Online].

Document Revision History

Version	Date	Description
v 1.1	4/22/2024	Milestone 1 deliverable

Appendices

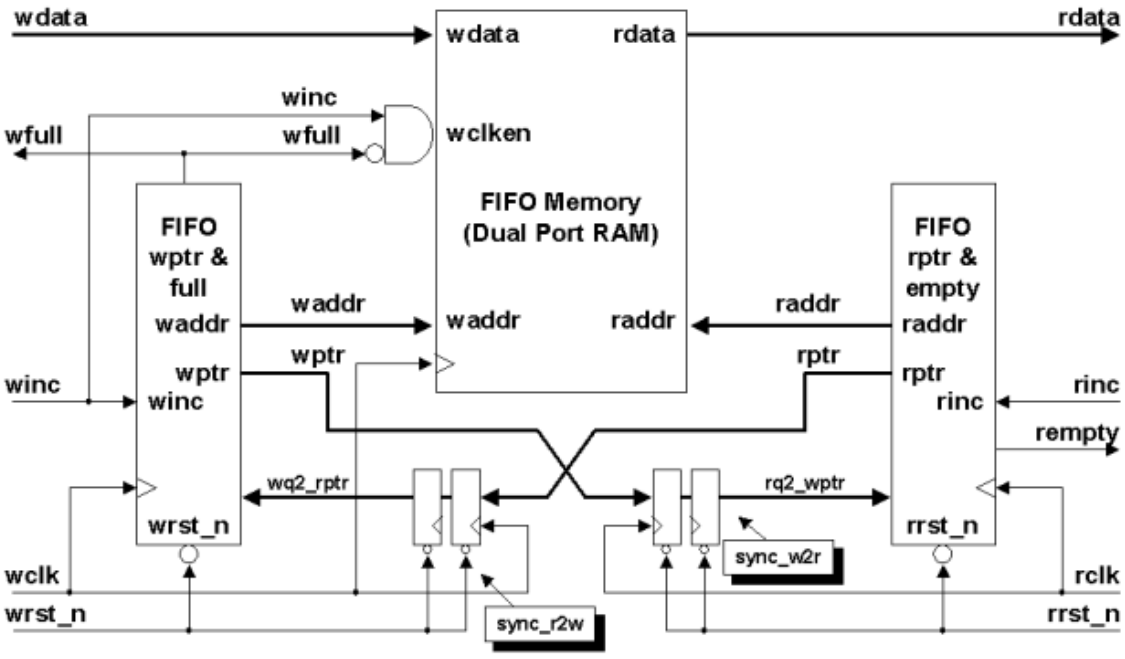

Figure 2: Block diagram for asynchronous FIFO from C. Cummings FIFO1

Signal	Direction	Description
WR_EN	Input	Data to be written into the FIFO.
WINC	Input	Control signal to write data into FIFO.
WCLK	Input	Write domain clock signal.
RINC	Input	Control signal to read data from FIFO.
RCLK	Input	Read domain clock signal.
RST_N	Input	Active-low asynchronous reset for FIFO.
WCLKEN	Input	Optional signal to enable writing on wclk rising edge.
RCLKEN	Input	Optional signal to enable reading on rclk rising edge.
RD_EN	Output	Data read from the FIFO.
FULL	Output	Indicates FIFO is full.
EMPTY	Output	Indicates FIFO is empty.
HALF_FULL	Output	Optional signal indicating FIFO is approaching full.
HALF_EMPTY	Output	Optional signal indicating FIFO is approaching empty.
WR_ACK	Output	Optional handshake signal for successful write.
WR_ERR	Output	Optional handshake signal for failed write.
RD_ACK	Output	Optional handshake signal for successful read.
RD_ERR	Output	Optional handshake signal for failed read.

Table 1: Core signal pinout for asynchronous FIFO