Design Specification Document

ECE-593 Fundamentals of Pre-Silicon Validation

Maseeh College of Engineering and Computer Science

Winter 2025

Project name- Design and Verification of Asynchronous FIFO
Members- Aadityasingh, Moulya, Rakshith, Subramanya
Date – 31 January 2025

Project Description

The Asynchronous FIFO (First-In-First-Out) memory module is designed to facilitate seamless data transfer between two clock domains that operate at different frequencies. Unlike a Synchronous FIFO, which operates under a single clock, an Asynchronous FIFO ensures reliable data exchange between modules that do not share the same clock, eliminating data loss and metastability issues.

Key Applications:

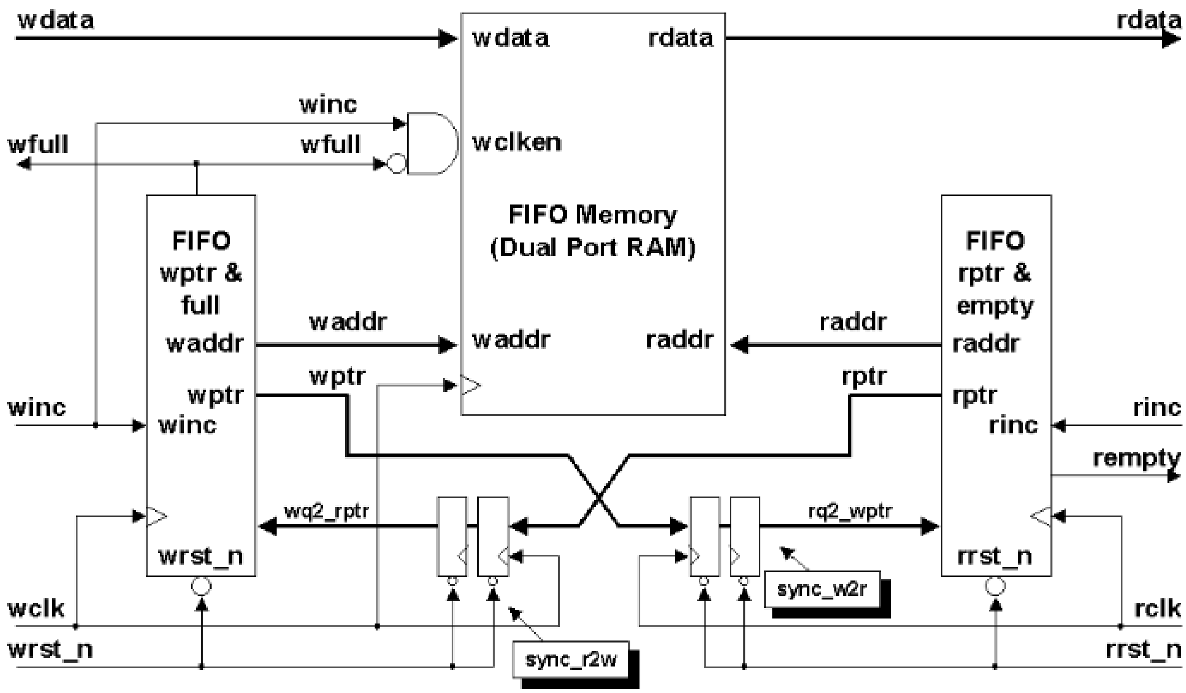
- Clock Domain Crossing (CDC): Essential in SoCs (System-on-Chips) and FPGA-based designs for transferring data between different clock regions.
- Processor-to-Peripheral Communication: Used in microprocessor-based systems where the CPU and peripherals operate at different speeds.
- Network and Multimedia Processing: Ideal for buffering high-speed data streams between fast and slow processing blocks.
- Data Flow Control in High-Speed Systems: Prevents data overflow or underflow when interfacing with sensors, memory units, or external interfaces.

Challenges Addressed:

- Metastability: By using Gray-coded pointers, the FIFO prevents metastability when transferring pointers between different clock domains.
- Data Integrity: Ensures that data is written and read correctly without loss, even with varying producer and consumer speeds.
- Efficient Flow Control: Implements status flags (FULL, EMPTY) to prevent erroneous writes or reads.

Design Features

The Asynchronous FIFO (First-In-First-Out) is a specialized memory buffer designed to facilitate seamless data transfer between two clock domains operating at different frequencies. Unlike synchronous FIFOs, which rely on a single clock, an asynchronous FIFO ensures reliable communication between independent clock regions by using dual clock domains, Gray-coded pointers, and synchronization mechanisms to prevent metastability. It consists of key sub-modules, including write and read pointer handlers, read-to-write and write-to-read synchronizers, and a FIFO memory block, all working together to maintain data integrity and efficient flow control. This design is crucial in SoCs, processors, and high-speed communication systems, where bridging different clock domains is essential for system performance and stability.



The Asynchronous FIFO consists of several sub-modules, each responsible for handling specific functions in the data transfer process. The top-level Asynchronous FIFO module includes the following sub-modules:

- FIFO Write Pointer
- FIFO Read Pointer
- Read-to-Write Synchronizer ()
- Write-to-Read Synchronizer
- FIFO Memory

FIFO Depth Calculation

Transmitter Clock (clk1) = 80MHz
Receiver Clock (clk2) = 50MHz
Maximum burst size = 120
Time required to write one data item = 1/80MHZ = 12.5 nSec.
Time required to write all data in the burst = 120* 12.5nSec = 1500nSec
Time required to read one data item = 1/50MHz = 20nSec.
Total reads in burst period = 1500ns/20nSec = 75
Required FIFO depth = 120 – 75 = 45
So, the minimum depth of FIFO should be 45(rounded to 64).

Sub modules Description

1. FIFO Write Pointer

Functionality:

- Manages the write address pointer (wptr) in the FIFO memory.
  - Ensures proper sequencing of writes by tracking where the next write operation should occur.
  - Converts the binary write pointer to a Gray code representation to prevent metastability when crossing clock domains.
  - The write pointer is synchronized to the read clock domain using the SYNCHRONIZER_W2R module.

Key Operations:

- Write Addressing: The pointer (wptr) increments when a write occurs.
- Gray Code Conversion: Ensures only one-bit transitions at a time, reducing metastability risks.
- FIFO Full Detection: The FIFO is full when the next write pointer matches the synchronized read pointer (rptr_s).

Key Signals:

Signal	**Description**
wclk	Write clock domain signal.
wrst	Active-low write reset.
wptr	Write pointer (binary & Gray code).
wptr_s	Synchronized write pointer in the read domain.
waddr	Write address for FIFO memory.
wFull	FIFO full flag.

FIFO Read Pointer

Functionality:

- Manages the read address pointer (rptr) in the FIFO memory.
- Ensures proper sequencing of reads by tracking where the next read operation should occur.
- Converts the binary read pointer to Gray code for synchronization with the write clock domain.
- The read pointer is synchronized to the write clock domain using the SYNCHRONIZER_R2W module.

Key Operations:

- Read Addressing: The pointer (rptr) increments when a read occurs.
- Gray Code Conversion: Prevents glitches and metastability in asynchronous clock transitions.
- FIFO Empty Detection: The FIFO is empty when the next read pointer matches the synchronized write pointer (wptr_s).
- Idle Cycles Handling: Enforces timing constraints for read bursts.

Key Signals –

Signal	Description
rclk	Read clock domain signal.
rrst	Active-low read reset.
rptr	Read pointer (binary & Gray code).
rptr_s	Synchronized read pointer in the write domain.
raddr	Read address for FIFO memory.
rEmpty	FIFO empty flag.

Read-to-Write Synchronizer

Functionality:

- Transfers the read pointer (rptr) from the read clock domain (rclk) to the write clock domain (wclk).
- Ensures the write logic knows when the FIFO is empty to prevent unnecessary writes.
- Implements a two-stage flip-flop synchronizer to reduce metastability risks.

Key Operations:

- Captures rptr in the read domain.
- Passes it through two D-flip-flops clocked by wclk for reliable synchronization.
- Outputs a synchronized read pointer (rptr_s) to be used in FIFO full detection.

Key Signals:

Signal	Signal
rclk	Read clock input.
wclk	Write clock input.
rptr	Read pointer in binary.
rptr_s	Synchronized read pointer in the write clock domain.

Write-to-Read Synchronizer

Functionality:

- Transfers the write pointer (wptr) from the write clock domain (wclk) to the read clock domain (rclk).
- Ensures the read logic knows when the FIFO is full to prevent invalid reads.
- Implements a two-stage flip-flop synchronizer to prevent metastability.

Key Operations:

- Captures wptr in the write domain.
- Passes it through two D-flip-flops clocked by rclk for reliable synchronization.
- Outputs a synchronized write pointer (wptr_s) to be used in FIFO empty detection.

Key Signals:

Signal	Description
wclk	Write clock input.
rclk	Read clock input.
wptr	Write pointer in binary.
wptr_s	Synchronized write pointer in the read clock domain.

FIFO Memory

Functionality:

- Stores data written by the producer (write clock domain).
- Allows reading by the consumer (read clock domain) at a different speed.
- Uses a dual-port RAM architecture to allow simultaneous read and write operations.

Key Operations:

- Write Operation:
    - wData is written to the memory location specified by waddr.
    - Write is enabled when winc is high and wFull is low.
- Read Operation:
    - Data is retrieved from raddr when rinc is high and rEmpty is low.
    - Read data is available at rData.
- Depth Management:
    - FIFO depth is calculated based on clock speeds and burst sizes to prevent data loss.

Key Signals:

Signal	Description
wclk	Write clock input.
rclk	Read clock input.
wData	Data written to FIFO.
rData	Data read from FIFO.
wAddr	Write address input.
rAddr	Read address input.
winc	Write enable signal.
rinc	Read enable signal.

Important Signals and Flags

Signal/Flag	Description
wFull	FIFO Full Flag – Set when FIFO reaches its maximum capacity, preventing additional writes.
rEmpty	FIFO Empty Flag – Set when FIFO has no available data for reading.
winc	Write Increment (Enable Signal) – Initiates a write operation if the FIFO is not full.
rinc	Read Increment (Enable Signal) – Initiates a read operation if the FIFO is not empty.
wData[7:0]	Write Data Bus – Data to be written into FIFO, 8-bit wide.
rData[7:0]	Read Data Bus – Data retrieved from FIFO memory.
wrst	Write Reset (Active Low) – Resets write-side logic, clearing write pointers.
rrst	Read Reset (Active Low) – Resets read-side logic, clearing read pointers.

FIFO Status Indicators

- FIFO Full (wFull): Prevents writes when the FIFO memory is completely occupied.
- FIFO Empty (rEmpty): Prevents reads when no data is available in the buffer.
- Almost Full & Almost Empty flags can be implemented to provide early warnings.

Design Signals

Signal	Description
wclk	Write Clock: Governs the writing of data into FIFO.
rclk	Read Clock: Governs the reading of data from FIFO.
wAddr[5:0]	Write Address: Specifies FIFO memory location where the next data write will occur.
rAddr[5:0]	Read Address: Specifies FIFO memory location from which the next data will be retrieved.
wPtr[5:0]	Write Pointer: Indicates the current write position in FIFO memory.
rPtr[5:0]	Read Pointer: Indicates the current read position in FIFO memory.
wPtr_s[5:0]	Synchronized Write Pointer: Used in the read domain to determine the FIFO empty condition.
rPtr_s[5:0]	Synchronized Read Pointer: Used in the write domain to determine the FIFO full condition.

References/Citations –

1. S. Cummings, "FIFOs: Fast, predictable, and deep," in Proceedings of SNUG, 2002. [Online]. Available: http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf.
2. S. Cummings, "FIFOs: Fast, predictable, and deep (Part II)," in Proceedings of SNUG, 2002. [Online]. Available: http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdf.
3. Putta Satish, "FIFO Depth Calculation Made Easy," [Online]. Available: https://hardwaregeeksblog.files.wordpress.com/2016/12/fifodepthcalculationmadeeasy2.pdf.
4. A. Author et al., "Title of the Paper," in Proceedings of the Conference, 2015, pp. 123-456. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7237325.
5. M. Last Name et al., "Designing Asynchronous FIFO," [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/56108360/EC109-libre.pdf.
6. Open AI, "Chat GPT," [Online]