Synchronous FIFO Design and Explanation

Ayush Yadav

26-11-2024

Contents

1 Overview

A **Synchronous FIFO (First In, First Out)** is a specialized buffer used in digital circuits to temporarily store data. It ensures that the data written first is read first, maintaining the order in which it was received. The FIFO operates synchronously, meaning that both read and write operations are driven by the same clock signal, which simplifies timing and coordination in digital designs. It is particularly useful in communication systems and digital processing where there is a need to decouple the rates of input and output data.

2 Components and Their Roles

2.1 Source

The source represents the data producer, often a sensor, processing unit, or data generator, that sends data to the FIFO. The data is typically transferred in chunks or individual words.

- **W_data (Write Data):** This is the actual data that is sent from the source to be written into the FIFO. It is typically a multi-bit data word. - **Full Signal:** This signal indicates whether the FIFO is full. If the FIFO is full, the source will be prevented from writing further data until space is available.

2.2 Write Enable Logic

Write Enable logic ensures that data is written to the FIFO only when certain conditions are met. It controls the process of writing data into the FIFO memory.

- **W_en (Write Enable):** This is a control signal that determines when data can be written into the FIFO. If `W_en` is high and the FIFO is not full, data can be written. If the FIFO is full, `W_en` will be blocked. - **Logic Gate:** The write enable logic ensures that the FIFO accepts data only when it is not full. A logic gate checks that `W_en` is high and that the FIFO is not in a full state before allowing data to be written.

2.3 Read Enable Logic

Read Enable logic ensures that data can be read from the FIFO under the correct conditions.

- **R_en (Read Enable):** This is the control signal that allows data to be read from the FIFO. Data can only be read when `R_en` is high, and the FIFO is not empty. - **Logic Gate:** Similar to write enable, the read enable logic verifies that the FIFO is not empty before permitting a read operation.

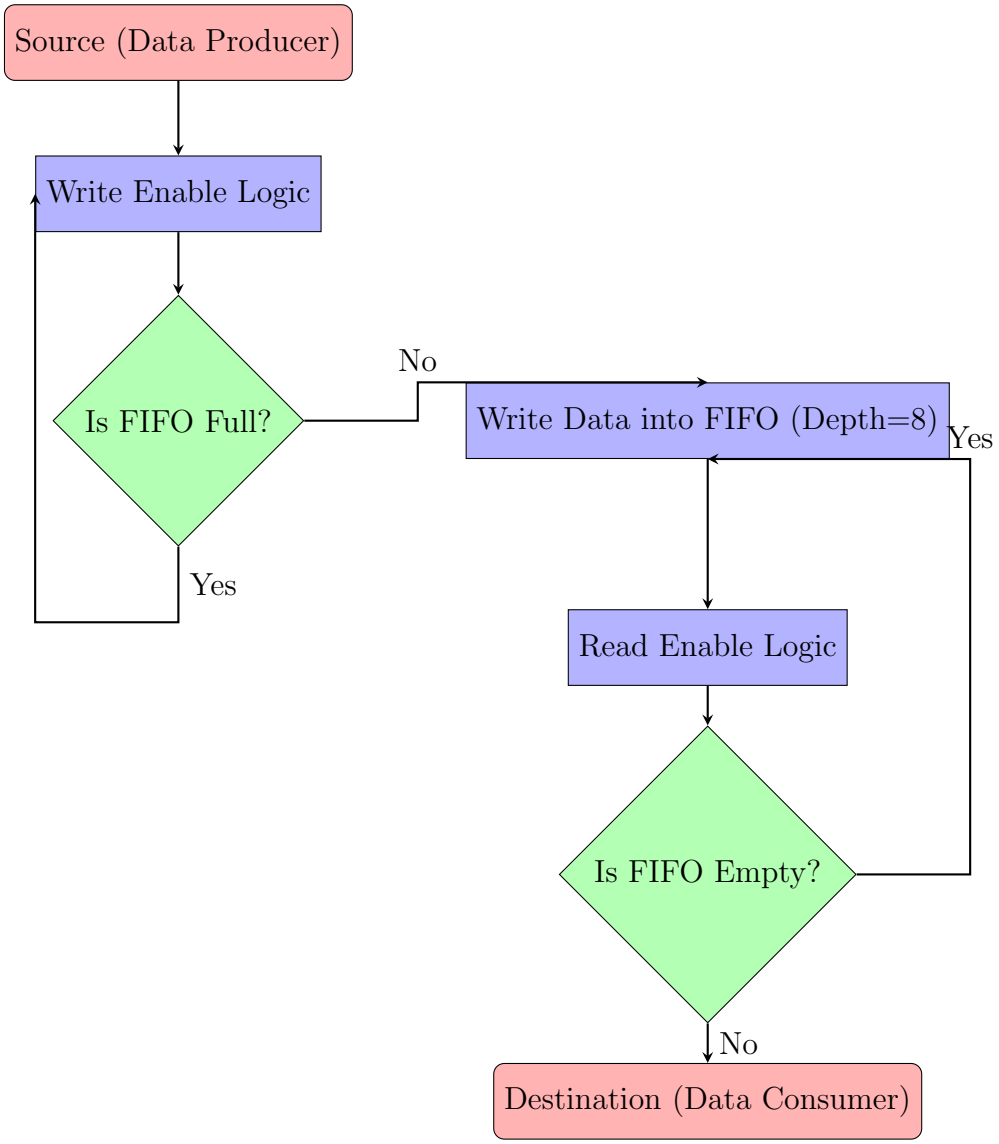
2.4 Destination

The destination represents the data consumer, such as a processing unit or output system, which retrieves data from the FIFO.

- **R_data (Read Data):** This is the data read from the FIFO and sent to the destination. It is typically the same format as the input data, processed or used by the destination. - **Empty Signal:** This signal indicates whether the FIFO is empty. If the

FIFO is empty, the destination cannot read any more data until new data is written into the FIFO.

3 FIFO Block Diagram



4 Detailed Logic of the FIFO Design in the Flowchart

4.1 Start with the Source (Data Producer):

Data originates from the source (data producer) and is ready to be written into the FIFO. This data could represent anything from sensor readings, control signals, or processed information. The source initiates the writing process when the FIFO has space for new data.

4.2 Write Enable Logic:

The write enable logic ensures that data is written into the FIFO only when certain conditions are met.

- Data can only be written to the FIFO if the `Full` signal is low, indicating there is space in the FIFO. - If the FIFO is full, the system will not allow any new data to be written until space becomes available (after data is read out). - Once data is written, the write pointer `Wptr` is incremented, and the data is stored in the FIFO.

4.3 FIFO Memory:

The FIFO stores the data in sequential order in memory.

- The memory is organized as a set of data slots, where each slot can hold one unit of data (e.g., 8-bit word). In this design, the FIFO has a depth of 8, meaning it can store up to 8 data units at any given time. - **Wptr** (Write Pointer) and **Rptr** (Read Pointer) keep track of where data is written and read. The `Wptr` increments each time data is written, and the `Rptr` increments each time data is read.

4.4 Read Enable Logic:

The read enable logic allows the destination system to retrieve data from the FIFO.

- Data can only be read from the FIFO if the `Empty` signal is low, meaning there is data available to read. - If the FIFO is empty, no data can be read until new data is written into the FIFO. - When data is read, the `Rptr` is incremented, and the data is removed from the FIFO.

4.5 Destination (Data Consumer):

The destination system is where the read data is sent. The data is processed or used by the consumer, which could be a digital processor, a communication module, or any other system that requires the data. The data is read in the order it was written, ensuring the FIFO's First In, First Out principle.

4.6 Full and Empty Conditions:

The FIFO operates based on the full and empty conditions, which determine whether data can be written or read.

- **Full Condition:** The FIFO is considered full when the write pointer `Wptr` and the read pointer `Rptr` indicate that there is no more space left in the FIFO. In a system with a depth of 8, the FIFO is full when `Wptr` and `Rptr` are offset by 8 slots (i.e., the FIFO has no available space for new data).

- **Empty Condition:** The FIFO is empty when the `Wptr` and `Rptr` pointers are the same, indicating that no data is available to be read from the FIFO.

5 Write and Read Operations

5.1 Writing to FIFO

- When `W_en` is high and the FIFO is not full, the data is written from the source to the FIFO. - The write pointer `Wptr` is incremented, ensuring that data is written to the next available slot.

5.2 Reading from FIFO

- When `R_en` is high and the FIFO is not empty, the data is read from the FIFO and sent to the destination. - The read pointer `Rptr` is incremented, ensuring the next data word can be read in sequence.

6 Applications of Synchronous FIFO

Synchronous FIFOs have a wide range of applications in digital systems, particularly when there is a need for reliable, ordered data transfer between different system components. Some common applications include:

- **Communication Protocols:** FIFOs are used in UART, SPI, and I2C communication protocols to temporarily store incoming and outgoing data.
- **Data Buffers:** FIFOs are used as buffers to store data between high-speed and low-speed processes in hardware/software interfaces.
- **Video and Audio Streaming:** FIFOs are used to buffer data in multimedia systems to handle different data rates between sources and consumers.
- **Inter-process Communication:** FIFOs are used in operating systems for communication between different software processes, ensuring data is passed in the correct order.