Design and Implementation of High-Speed Asynchronous FIFO

Xinran Jia *

School of Microelectronics, Xidian University, Shaanxi, 710126, China

* Corresponding Author Email: 22009100641@stu.xidian.edu.cn

Abstract. Asynchronous FIFO is a crucial module for adapting phase differences and frequency drift between different clock domains. With the rapid development of integrated circuit technology and the increasing transistor density on chips of the same area, design systems are evolving toward large-scale SOC. This evolution has raised the challenge of handling multiple clock domains in large-scale integrated circuits. This research presents an excellent solution using asynchronous FIFO (First in First Out) circuits, which can effectively transmit data under different clock domains, showing promising prospects in image processing and interface applications. The FIFO design employs gray code for read/write address encoding, successfully avoiding metastability. The paper emphasizes the design approach for empty/full state detection and implements the design using Verilog. Simulation results demonstrate high system reliability, strong interference resistance, and good scalability through modular design. In modern integrated circuit systems, different modules often operate under different clock domains, and efficient, accurate data transmission between these clock domains has been a technical challenge. Asynchronous FIFO circuits enable synchronized data transmission between clock signals of different frequencies and phases, effectively solving this problem.

Keywords: High-speed asynchronous FIFO, Modular design, Simulation verification.

1. Introduction

With the rapid development of integrated circuit technology, this research tends to adopt circuits with higher integration and precision in design and application. IC design has entered the SoC era, and while pursuing higher integration, more asynchronous data is generated. Implementing high-precision transmission of these asynchronous data across different clock domains has become a necessary step in design. Asynchronous FIFO design precisely meets the need for effective data transfer between different clock systems [1-3]. FIFO primarily implements 8-bit data access at once, outputs empty/full status signals at any time, provides easily readable FIFO status, and effectively prevents glitches during operation, ensuring efficient and stable data transmission. In circuits, asynchronous FIFO typically serves as a rate-matching data buffer, achieving rate matching for data transmission between fast processors and slow peripherals [6]. In military requirements and industrial production applications, asynchronous FIFO is mainly used in fields related to data storage and transfer, such as image processing, long-distance communication, storage systems, and digital signal processing [6]. With continuous development in FIFO chip-related fields across various industries, the application prospects of asynchronous FIFO will become increasingly promising. Due to the rapid development of microelectronic technology, new-generation FIFO chips have increasingly larger capacity, smaller size, and faster speed. The U.S. company IDT has launched a FIFO series with operating speeds up to 225MHz, voltage as low as 2.5V, and the industry's largest data throughput with up to 9Mbit density under various configurations. The latest product in the FIFO field is IDT's multi-queue FIFO memory series, which integrates high-speed queue logic and embedded FIFO memory cores to form a block structure. Data write and read speeds can reach 200MHz, with storage time of only 3.6ns, and depth and queue expansion can be achieved through connection of up to eight devices [4]. Currently, most of China's high-end storage devices rely on imports, domestic FIFO circuits have not yet been systematized, and there is a huge gap in military demand [6].

This project's advanced high-speed asynchronous FIFO aims to address common issues such as glitches and metastability in FIFO design and implementation, with the goal of designing a highly reliable asynchronous FIFO circuit with high-speed processing capabilities. The research results of this project have broad application prospects in China's current market. Based on these achievements, through further development and integration, it is expected to provide an effective solution to China's predicament of large-scale reliance on imported storage devices for military equipment.

2. Structure of Asynchronous FIFO

An asynchronous FIFO typically consists of storage units, write address module, read address module, and empty/full state generation module. Due to the asynchronous nature of read and write pointers, asynchronous FIFO design is generally divided into read clock and write clock blocks.

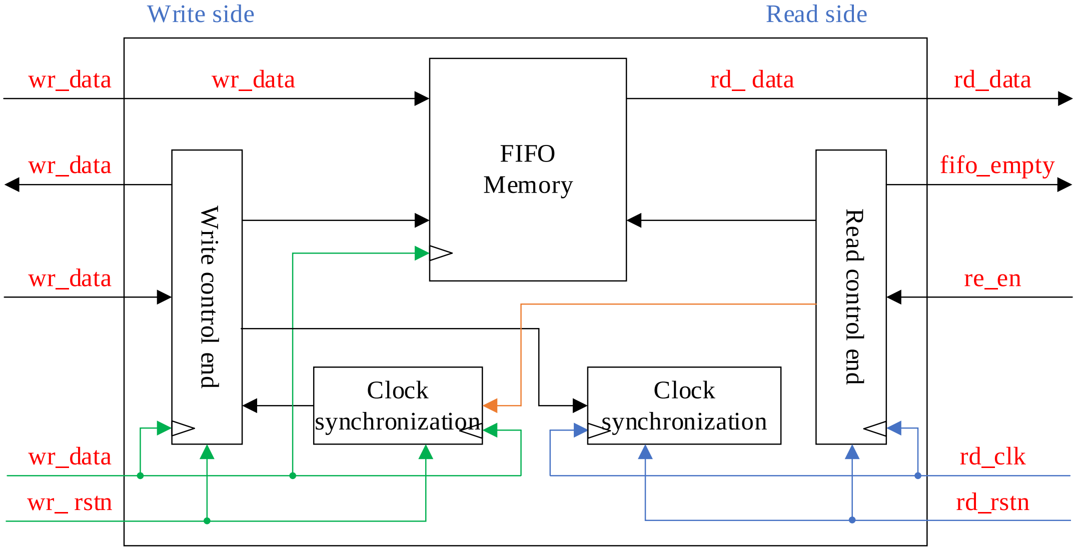

Figure 1. FIFO Structure Diagram

As shown in Figure 1, read and write regions operate independently around the asynchronous FIFO's storage state, generating read/write addresses, read/write pointers, and read/write enable signals respectively. In the empty/full state generation module, empty/full flags are generated by comparing the Gray-coded read and write addresses. When the write clock's rising edge arrives and write enable is active, the asynchronous FIFO writes data into the storage unit. When the read clock's rising edge arrives and read enable is active, data is output in a first-in-first-out order, with fifo_empty and fifo_full signals indicating the storage unit's empty/full status.

3. Working Principle of Asynchronous FIFO

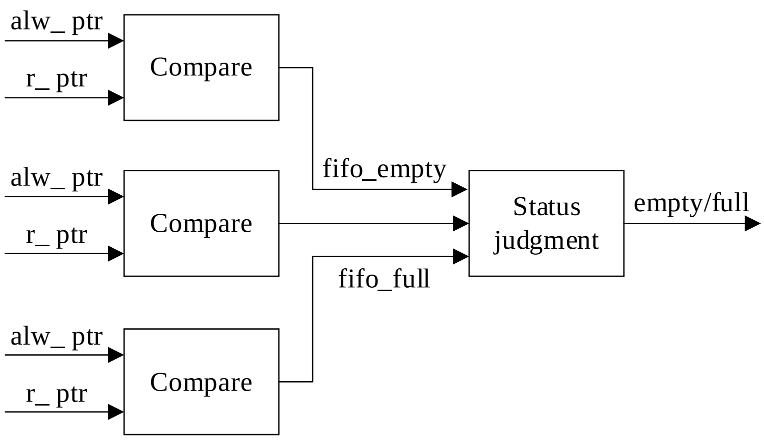

Figure 2. Empty/Full Status Determination Diagram [5]

Asynchronous FIFO (First in First Out) memory, or first-in-first-out queue, primarily consists of write control end, read control end, storage module, and corresponding write and read clocks. The read/write control ends determine whether data can be written or read, with the key conditions being valid write enable and non-full FIFO for write operations, and valid read enable and non-empty FIFO for read operations. Therefore, two enable signals and empty/full determination signals are connected to the control end, along with clock and reset signals. Asynchronous FIFO is a dual-port device where data enters through one port and exits through another. In design, pointers are typically used to manage data, with write addresses stored in the write pointer and read addresses stored in the read pointer.

Figure 2 demonstrates empty/full status determination through pointer comparison. The empty/full flags of asynchronous FIFO are derived from comparing read/write address pointer values, preventing overflow when full and under-reading when empty. In empty state, read and write addresses are identical. Taking state 0010 as an example, its gray code is 0011, and in empty state, both binary and gray codes are identical; in full state, the write pointer has completed one more cycle than the read pointer. To better distinguish states, address width is expanded from 3 bits to 4 bits. For instance, state 1010's Gray code is 1111, while 0010's gray code is 0011; comparison shows opposite high two bits and identical low two bits, indicating full status. Due to the flexibility of asynchronous FIFO design, flags such as half-full, almost-full, and almost-empty can be added as needed.

High-speed asynchronous FIFO significantly outperforms similar storage circuits in reliability, speed capability, and flexibility, holding an irreplaceable position as the optimal solution for data transmission in asynchronous sequential logic circuits with current technology. In terms of reliability, asynchronous FIFO implements empty/full signal output and can appropriately add thresholds as needed, outputting corresponding warning signals or half-full/almost-full signals, greatly enhancing programmability. Internally, using gray code for comparison eliminates glitches and metastability phenomena associated with binary code, further improving circuit stability. This design not only enhances reliability but also improves system speed through gray code comparison-based empty/full determination standards. Performance indicators show excellent high-speed processing capability - simulation results using full-custom circuit design based on UMC 28 nm standard CMOS process demonstrate a maximum operating frequency of 666.6 MHz with average power consumption of 7.1 mW at 1 V standard voltage. The outstanding high-speed processing capability meets high-speed data transmission and processing requirements [3]. In programming design, asynchronous FIFO achieves higher programmability through adding near-empty/full warning thresholds and status bits. This programmability enables adaptation to various application scenarios and requirements, improving flexibility and applicability. High-speed asynchronous FIFO provides more efficient, flexible data processing solutions through enhanced reliability, high-speed processing capability, and improved programmability. Compared to synchronous FIFO, asynchronous FIFO circuits don't require clock signals for reading/writing data, offering greater flexibility and fault tolerance. Since asynchronous FIFO is unaffected by clock skew during read/write operations and doesn't require strict timing consistency, it's suitable for systems with various operating frequencies. This characteristic makes asynchronous FIFO more flexible in design and application, adaptable to different operating frequencies and environmental requirements.

4. Challenges in Asynchronous FIFO Implementation

4.1. Metastability

During FIFO write and read operations, due to the lack of regular logical relationships between the two clocks in asynchronous circuits, when sampling in one clock domain, it's possible to encounter metastability where the sampled value falls between 0 and 1 [5] [6]. If metastability propagates to subsequent combinational logic circuits, it can cause FIFO's basic functionality to fail, specifically its ability to accurately transfer data. Therefore, resolving metastability issues is urgent in implementing asynchronous signal synchronization.

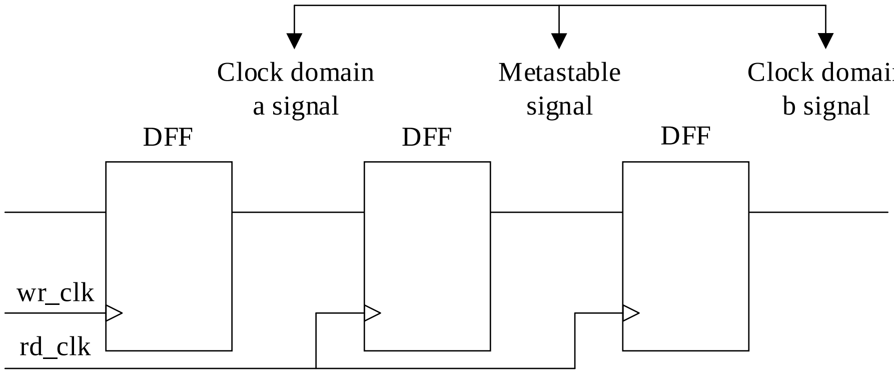

Figure 3. Synchronization Module Circuit Diagram

As shown in Figure 3, metastability will transition to a stable state after a certain time, typically within one clock cycle. Therefore, in design, a synchronizer using two cascaded registers is generally employed, implementing a two-cycle delay of gray code in the code to eliminate metastability.

4.2. Glitch Phenomena

Another challenging problem accompanying asynchronous signals is glitch phenomena. When multiple bits change simultaneously at the input signal, glitches are inevitable due to impossible perfect timing alignment of different bit positions in FIFO encoding. This can result in incorrect glitch signals being sampled by the other clock, causing synchronization errors in asynchronous signals [7][8]. The occurrence of glitches not only affects overall FIFO performance but also generates indeterminate states and other errors during FIFO operation, preventing effective data transmission. Such errors can make FIFO unable to determine empty/full signals or output correct data.

The FIFO circuit designed in this research ensures that circuit signals are latched by registers before synchronization and guarantees that only one-bit changes in each encoding transition, effectively reducing or even eliminating glitch issues during FIFO operation. Gray code's characteristic of single-bit changes perfectly meets the design requirements, making it the widely adopted encoding method in this design and most asynchronous FIFO designs [9] [10].

5. Simulation Verification

In the field of Electronic Design Automation (EDA), the design and implementation of asynchronous FIFO is a relatively mature technology. Based on the above analysis, this paper designed an 8×8 high-speed asynchronous FIFO and used MODELSIM for code debugging, simulation, and functionality demonstration, as shown in Figure 2.

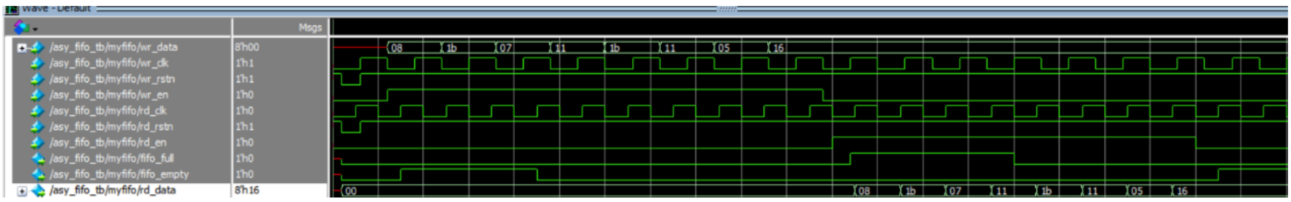

Figure 4. Simulation Waveform Diagram

In Figure 4, wr_clk, wr_en, fifo_full, wr_data, rd_clk, rd_en, fifo_empty, rd_data, wr_rstn, and rd_rstn represent write clock signal, write enable signal, full signal, write pointer, read clock signal, read enable, empty signal, read pointer, and two reset signals respectively. Figure 2 shows the corresponding waveforms of each signal.

After the fifo_empty signal is pulled high, data begins to enter FIFO, and after a time delay when the fifo_full signal is pulled high, data begins to output. It can be observed that both input and output signals are 8-bit data, and their contents are consistent, achieving stable and effective data storage and retrieval.

This research completed the design and implementation of FIFO, creatively setting independent read/write clocks, reset, and enable signals on top of basic FIFO functionality. This allows clearer observation of FIFO operation, facilitates error correction during research, modifications during use, and easier error handling, while effectively reducing time needed for future innovative rewrites and function additions. This paper proposed an asynchronous FIFO design using gray code for address encoding and implemented circuit design using Verilog language. The circuit software simulation has been verified, proving it can resolve asynchronous-generated errors while increasing application flexibility, providing a basic implementation method for FIFO design and implementation.

This paper addressed metastability, glitches, and empty/full state determination in FIFO design and implementation, achieving simulation at 50MHz using Verilog language, with results closely matching expectations and meeting basic design requirements. Current improvement directions and research goals include optimizing FIFO performance degradation caused by "near empty/full state" determination and addressing increased register circuit area due to gray code word length. FIFO chips' programming flexibility and high-speed characteristics are increasingly showing advantages in high-speed data acquisition, processing, transmission, and multi-processor systems. There are broad development prospects and inherent development momentum in both domestic and international high-end hardware fields and military requirements [4].

6. Conclusion

This research provided a method for implementing FIFO circuits using Verilog language, employing gray code for address encoding, effectively solving metastability and glitch issues while outputting empty/full state signals, enhancing FIFO's utility. The research still has issues such as lack of hardware implementation. FIFO is mainly used in military data storage design. Asynchronous FIFO's advantages are demonstrated in remote communication, image processing, large-capacity storage systems, digital signal processing, and printing systems. Its unique functionality and continuously optimized performance make FIFO chips' application prospects increasingly promising. Rapid microelectronic technology development has reduced FIFO chip size while increasing capacity, with performance continuously improving through technological advancement, including increased read/write speeds, reduced power consumption, and enhanced stability. In emerging fields, FIFO aligns well with development needs. With the rapid development of Internet of Things (IoT), autonomous driving, and Artificial Intelligence (AI), high-speed asynchronous FIFO will play important roles: as data buffers in IoT devices storing sensor data, processing real-time data from multiple sensors in autonomous driving systems ensuring safe vehicle operation, and supporting real-time inference and training of deep learning models in AI applications. In market competition and military aspects, the global asynchronous FIFO memory market is dominated by several foreign manufacturers. As FIFO applications broaden, competition will intensify. To prevent foreign technical blockades and expedite domestic high-performance asynchronous FIFO development, this project's research has significant implications for strengthening military construction and developing China's electronic information field.

References

[1] Shashank C Pai, Vishwitha A, Rakshath, Sathwik Bhat, Shreya. A REVIEW ON SYNCHRONOUS & ASYNCHRONOUS FIFO DESIGN. International Journal of Advanced Research in Computer and Communication Engineering, Vol. 12, Issue 6, June 2023.

[2] Verilog, Expert & Cummings, Clifford. (2002). Simulation and Synthesis Techniques for Asynchronous FIFO Design.

[3] Qing-Gong Xu, Zi Li, Jia Zhao, et al. Design and Implementation of High-Speed Asynchronous FIFO. Microcomputer Information, 2009, 25 (20): 153 - 155.

[4] Shi-Wei Xu. Design and Implementation of Asynchronous FIFO. Xidian University, Shaanxi, 2007.

[5] Bo Niu, Hong-Liang Zhao. Design of a High-Reliability High-Speed Programmable Asynchronous FIFO. Electronic Technology Applications, 2019, 45 (07): 36 - 39+43.

[6] P. Wielage, E. J. Marinissen, M. Altheimer and C. Wouters, "Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO," 2007 Design, Automation & Test in Europe Conference & Exhibition, Nice, France, 2007, pp. 1 – 6.

[7] Xian-Zhe Luo, Ren-Zhe Zhang, Da-Peng Fu. Design and Implementation of High-Speed Asynchronous FIFO Memory Based on FPGA. China Science and Technology Information, 2009, (02): 90 - 91+94.

[8] Ramesh, G. (2012). Asynchronous FIFO Design with Gray code Pointer for High Speed AMBA AHB Compliant Memory Controller. IOSR Journal of VLSI and Signal Processing. 1. 32 - 37.

[9] Hong-Bing Xiong, Qi Chen. Design and Implementation of Asynchronous FIFO Based on FPGA. Microcomputer Information, 2006, 22 (17): 216 - 218.

[10] Verilog, Expert & Cummings, Clifford. (2002). Simulation and Synthesis Techniques for Asynchronous FIFO Design.