Dean&Francis
ISSN 2959-6157

Current Status and Prospects of the Design and Implementation of Asynchronous FIFO.

Haoyang Guan[1, *]

[1]College of Electronic and Information Engineering, Hebei University, Baoding, China

*Corresponding author: 1813010107@stu.hrbust.edu.cn

Abstract:

As modern digital system scales expand, multi-clock domain designs are common. Asynchronous FIFO is crucial for cross-clock domain data transmission. This paper studies its design and implementation. Firstly, it explains the importance and purpose, like solving data transmission issues and enhancing system stability. Then, it details basic principles including working principles and solutions for asynchronous clock interaction. For design status, it analyzes common architectures and hardware implementation with optimization strategies. It also points out challenges like reliability and performance-resource balance. Looking ahead, it anticipates future trends towards higher performance, lower power consumption, and stronger adaptability. Innovation directions may include new architectures and algorithms. Research shows asynchronous FIFO is important but needs improvement to meet growing needs. As electronic systems become more complex, the requirements for data processing speed and efficiency are also increasing. This has led to a growing demand for asynchronous FIFO. As data rates increase, asynchronous FIFOs need to be designed to be more efficient

Keywords: Modern digital systems; asynchronous FIFO; basic principles; development status; innovation directions.

1. Introduction

First-In-First-Out (FIFO) is a common buffer structure used in digital circuit design to solve data transmission between different clock domains. In modern digital systems, due to the complexity and diversity of functions, there are often multiple different clock domains. For example, in a communication system, the receiving end and the transmitting end may operate at different clock frequencies. In a system containing a processor and external devices, the processor and external devices may also have their own independent clocks. When data needs to be transmitted between these different clock domains, the prob-

ISSN 2959-6157

lem of clock synchronization will be faced. And if data transmission is directly carried out, it may lead to data loss, errors, or system instability. Asynchronous FIFO provides an effective solution.

The study of asynchronous FIFO is of great significance. From a technical perspective, its purpose lies in mastering the principles and methods of clock asynchronous processing. Through research, it can address the uncertainty in data transmission caused by clock skew, jitter, and metastability, thereby enhancing the stability and reliability of the system. In practical applications, asynchronous FIFO is widely used in fields such as communication, computers, image processing, and embedded systems. For example, in communication systems, it can buffer data between interfaces of different rates; in image processing, it can cache image data to match the speed of processing modules; in embedded systems, it coordinates the asynchronous operations of subsystems[1].

From the perspective of performance optimization, the focus of studying asynchronous FIFO is to find efficient design and implementation methods that reduce resource consumption, increase data transmission rates, reduce latency, and improve storage efficiency, which is crucial for enhancing system performance and competitiveness. From the perspective of innovation and development, it helps to promote related technological innovations. It helps to explore new architectures, algorithms, and technologies. This brings more possibilities and flexibility to the design of digital systems. It helps adapt to changing needs and trends[2].

Asynchronous FIFO is in a key position in modern integrated circuit design. Asynchronous FIFO presents many characteristics: On the one hand, it is widely used in fields such as network communication and digital signal processing. It is a powerful tool for multi-clock domain data transmission. For example, in network equipment, it ensures stable buffering and transmission of data. On the other hand, it has various implementation methods. For example, the traditional pointer synchronization method has a high cost when the pointer has many bits and the data depth is large. The gray code pointer optimization utilizes characteristics to avoid metastability and improve design efficiency. The IP cores of major FPGA manufacturers bring convenience to engineering applications, but there is still room for exploring performance and customization options. In terms of performance, it is focused on improving speed, capacity, reliability, and programmability. For example, optimizing the circuit structure and improving the empty-full judgment algorithm can increase the operating frequency and data throughput. Increasing the near-empty-full warning threshold and status bits can improve programmability. In short, the research and application of asynchronous FIFO have broad prospects and important value[3].

This article supplements the study of asynchronous FIFO, including technical principles, performance optimization, and innovation development. In terms of technical principles, it explores new clock synchronization technologies and optimizations in different application scenarios. In the aspect of performance optimization, it discusses the balance between resource consumption and performance improvement as well as new technical means. In terms of innovation and development, it explores innovations in architecture, algorithms, and implementation technologies, which are instructive for fields such as communications and image processing.

2. Basic Principles of Asynchronous FIFO Section Headings

2.1 Working Principle and Characteristics of FIFO

FIFO (First In First Out), that is, a first-in-first-out memory, plays an important role in digital systems. Its working principle is as follows: FIFO is mainly composed of storage units, write pointers, read pointers, control logic and status flags. During the write operation, data is sequentially written into the storage unit under the control of the write clock. The write pointer increments with each write operation and points to the next available storage location. Once the write pointer loops back to the initial position, it means that the FIFO is full. If data continues to be written, it may cause data overwriting or errors. In the read operation, data is sequentially read out from the storage unit under the control of the read clock. The read pointer increments with each read operation and points to the next data location to be read. When the read pointer is equal to the write pointer, it indicates that the FIFO is empty. At this time, if data continues to be read, valid data cannot be obtained[4].

FIFO has the following significant characteristics: First of all, data sequentiality is one of its important characteristics. It strictly guarantees that data is read out in the order in which it enters. This is crucial in many scenarios that need to maintain data order, such as data acquisition systems and pipeline processing. Secondly, FIFO can play an effective buffering role between modules of different speeds. When a high-speed module transmits data to a low-speed module, FIFO can store the data generated by the high-speed module to avoid data loss, and at the same time allow the low-speed module to read data at its own speed. Furthermore, the FIFO design is relatively simple. Through several control signals such as write enable, read enable, full flag, and empty flag, data storage and reading

can be realized, and it is easy to be integrated into various systems. Finally, FIFO has asynchronous clock domain compatibility and can be used for data transmission between different clock domains. Through appropriate control logic, problems caused by clock differences can be avoided.

2.2 Problems of Asynchronous Clock Domain Interaction and Solutions.

In digital systems, interactions between asynchronous clock domains can lead to a series of issues. One of the most significant problems is metastability. When a signal is transmitted from one clock domain to another, the phase and frequency differences between the clocks may cause the receiving clock domain to be in an uncertain state when sampling the signal, known as a metastable state. Metastability can lead to data errors, system instability, and even system crashes. Additionally, data loss or duplication is a common issue. Without a proper synchronization mechanism between different clock domains, data loss or duplication may occur. For example, data written in one clock domain may not be read in time in another clock domain, leading to data loss; or the same data may be read multiple times due to clock differences during the read process. Moreover, the clocks in different clock domains may have offsets and jitter, which also affect data sampling and transmission.

To address the issues of asynchronous clock domain interaction, the following approaches can be taken. A common method is to use a synchronizer. A synchronizer can synchronize signals from different clock domains, reducing the risk of metastability. Synchronizers are typically constructed by cascading multiple flip-flops. Another method is to use a FIFO for data buffering. FIFO can safely transfer data between different clock domains, avoiding data errors caused by clock differences. With appropriate control logic, the correct writing and reading of data in the FIFO can be ensured. In addition, data transmission can be carried out through handshake signals. The sender and receiver coordinate the transmission of data through handshake signals to ensure the correct reception and processing of data.

For example, a controllable delay asynchronous FIFO circuit can be designed. Based on the existing methods for correctly transferring data across clock domains, the input data is dynamically adjusted and then delayed through the FIFO circuit to eliminate the skew between multiple data sources. To improve the adjustable delay precision and achieve more accurate synchronization between data, a fractional delay is designed. This FIFO can control the circuit delay during data transfer across clock domains. When multiple chips work simultaneously, it can eliminate the skew between data sources, making the output synchronized[5].

3. Current Status of Asynchronous FIFO Design and Implementation

3.1 Common Design Architectures

The traditional asynchronous FIFO design architecture usually uses dual-port RAM as the core storage unit and combines control logics such as read-write pointers and full-empty flag bits. In this architecture, read and write operations are performed in different clock domains respectively, and the state of the FIFO is judged by comparing the read and write pointers. Its advantage lies in that the design is relatively simple and intuitive, easy to understand and implement. Dual-port RAM can perform read and write operations simultaneously, which to a certain extent improves the efficiency of data transmission. However, the traditional architecture also has some drawbacks. When handling high-speed data transmission, there may be a pointer synchronization problem, which leads to misjudgment of the FIFO state. In addition, the utilization rate of storage resources in traditional architecture may not be efficient enough, especially when designing a large-capacity FIFO, this problem is more prominent[6].

In order to overcome the shortcomings of traditional architectures, numerous improved asynchronous FIFO design architectures have emerged. One common improvement is the use of Gray code pointers instead of traditional binary pointers. Since Gray code changes only one bit between adjacent counts, it makes the pointer synchronization between different clock domains more reliable, greatly reducing the possibility of misjudgment [6]. Another improvement is the introduction of multi-clock domain synchronization techniques, which ensure the correct transmission and state judgment of data by inserting synchronization registers between different clock domains. This architecture significantly enhances the stability and reliability of asynchronous FIFO in complex systems. In addition, some improved architectures focus on optimizing the utilization of storage resources. For example, by adopting the method of dynamically allocating storage units and adjusting the storage capacity according to the actual data volume, the efficiency of resource use is effectively improved[7].

The delay controllable asynchronous FIFO circuit has the following characteristics: by adding a delay control module, it achieves control of integer delay and fractional delay[2][4]. Integer delay is achieved by adjusting the difference between the read and write pointers, with the unit being the input data clock cycle, and the adjustment

ISSN 2959-6157

range being 0 to 7 write clock cycles; fractional delay is achieved by adjusting the phase difference between the read clock and the write clock, with the unit being the device clock cycle, and the adjustment range being 0 to (IR - 1) device clock cycles. This design improves the accuracy of delay control, allowing more precise adjustment of data delay to meet the requirements of data rate in the subsequent frequency conversion circuit of the FIFO[5]. In addition, the circuit also has the functions of real-time calculation of delay and monitoring the correctness of the set delay, as well as improving anti-interference ability and operating speed. An update signal is generated only when the interpolation rate or delay changes, thereby reducing unnecessary power consumption and interference.

3.2 Implementation Technologies

Hardware description languages (such as Verilog and VHDL) play a crucial role in the design of asynchronous FIFOs. By using hardware description languages, the logical functions and timing characteristics of asynchronous FIFOs can be precisely described. Designers can utilize the powerful functions of hardware description languages to implement complex control logics and data processing algorithms. At the same time, hardware description languages also have good portability and scalability, enabling asynchronous FIFOs to be implemented on different hardware platforms.

Asynchronous FIFO can be implemented on a variety of hardware platforms, such as FPGAs and ASICs. On the FPGA platform, designers can leverage the programmability and flexibility of FPGAs to quickly implement the design of asynchronous FIFO and perform verification and debugging. FPGAs also have parallel processing capabilities and high-speed data transmission characteristics, making them suitable for high-speed data acquisition and processing systems.

Asynchronous FIFOs can achieve higher performance and lower power consumption on the ASIC platform. ASIC design requires a strict design and verification process to ensure the chip's reliability and stability. However, ASIC design has a higher cost and a longer development cycle, but it is suitable for application scenarios with large-scale production and high-performance requirements [1].

Its architecture is circular, and the FIFO unit includes a transmission control module and a data transmission channel, where data is first paralleled through the data transmission channel and then serially output to the receiving end[8]. The read/write protocol forms read/write pointers through two data tokens generated during initialization, and the pointers are passed clockwise within the FIFO ring, involving two types of handshaking between FIFO stages and within stages, supporting a combined serial and parallel data transfer mode. The data transmission channel encodes the least significant bit of data with dual-rail encoding to ensure high-speed and stable data transmission[9]. The transmission control module is composed of a send control, empty/full control, and receive control modules, which implement data read/write control and the generation and synchronization of empty/full signals. The data transmission channel uses dual-rail encoding to detect the least significant bit of data, ensures communication through handshaking, and after the data is transmitted, it is passed to the receiving end through a serial transmission path. Under different clock frequencies and data transmission conditions, the FIFO can achieve complete and fast data transmission, with a transmission delay of 681ps, a critical path of 696ps, a maximum operating frequency of 1.75GHz for the transmit end, a maximum operating frequency of 1.77GHz for the receiving end, and the maximum throughput of a 3-stage FIFO with data items as the transmission unit is 1.44GDI/s. Under specific conditions, the average dynamic power consumption is 6.45mW, and the average energy consumption per transmission request is 6.45pJ. Compared with other FIFO implementation methods, the FIFO has advantages in terms of total delay, throughput, data transmission security, and design structure. It adopts a combined serial and parallel transmission mode, and data does not need to be passed through the FIFO stage by stage, has low power consumption, a simpler design, separates the transmission control module and data transmission channel, empty/full control from read/write pointers, and can meet different transmission reliability requirements. It features high throughput, low power consumption, strong robustness, and good reusability.

3.3 Optimization Strategies

In the design of asynchronous FIFOs, area optimization is an important consideration. Chip area can be reduced by adopting a streamlined logic structure, optimizing the design of storage units, and making reasonable use of hardware resources. For example, using single-port RAM instead of dual-port RAM can reduce the area overhead of storage units. In addition, the control logic can also be optimized to reduce the number of logic gates and thereby reduce the chip area.

To improve the data transfer speed of asynchronous FIFO, various strategies can be employed. One method is to optimize the synchronization mechanism of read and write pointers to reduce the time overhead of pointer synchronization. Another approach is to use pipelining technology, which divides the writing and reading operations of data

into multiple stages to increase the parallelism of data processing. Additionally, the overall speed of asynchronous FIFO can be improved by optimizing the access time of storage units and the delay of control logic.

In the design of asynchronous FIFO, power consumption reduction is also an important optimization goal[3]. Power consumption can be reduced by using low-power hardware components, optimizing circuit design, and reasonably controlling clock frequencies[6]. For instance, selecting low-power storage units and logic gates can decrease static power consumption[7]. Moreover, power consumption can be reduced by dynamically adjusting clock frequencies and power supply voltages according to the actual workload.

For instance, a circular FIFO can be designed that is applicable to standard cell design. By employing a unique operation protocol and transmission encoding method, the correct operation of the FIFO is regulated by two types of handshakes between FIFO levels, and between the transmission control modules and data transmission channels within each level. The architecture of the FIFO is based on the separation of transmission control logic and data transmission channels, as well as the separation of data synchronization modules and read/write pointers. This FIFO supports a combination of serial and parallel data transfer methods, as well as different width data transfer modes, meeting various requirements for transmission reliability. The minimum transmission delay of three flip-flop delays enhances data transfer efficiency and increases the reusability of the FIFO, avoiding metastability during transmission and achieving complete high-speed data transfer between different clock domains [9].

4. Current Challenges

In complex electronic systems, asynchronous FIFOs face many reliability challenges. First of all, due to the existence of asynchronous clock domains, the cross-clock domain transmission of data may lead to metastability problems, which may cause data errors and affect the stability of the system. Secondly, in a high-noise environment, asynchronous FIFOs may be interfered with, resulting in data loss or incorrect writing. In addition, as the scale of the system continues to expand, asynchronous FIFOs may need to interact with modules in multiple different clock domains, increasing the probability of errors.

On the one hand, modern electronic systems demand increasingly higher performance from asynchronous FIFOs, requiring faster read and write speeds, larger storage capacities, and lower latency. On the other hand, in resource-constrained environments such as embedded systems or portable devices, chip area and power consumption become important considerations. Therefore, how to meet high-performance demands while effectively controlling resource consumption is a significant challenge currently faced in the design of asynchronous FIFOs.

With the continuous development of technology, asynchronous FIFOs are being applied to more and more new scenarios, such as artificial intelligence, the Internet of Things, and high-speed communications. These new application scenarios pose special demands for asynchronous FIFOs. For instance, in the field of artificial intelligence, there is a need for asynchronous FIFOs to handle large-scale parallel data; in the Internet of Things, there is a requirement for asynchronous FIFOs to be low-power and small in size; and in high-speed communications, higher demands are placed on the read and write speeds and reliability of asynchronous FIFOs.

5. Future Prospects of Asynchronous FIFO

With the continuous progress of semiconductor technology, the integration degree of asynchronous FIFOs will become higher and higher, the storage capacity will continue to increase, and the power consumption will be further reduced. Secondly, in order to improve the reliability of the system, asynchronous FIFOs will adopt more advanced error detection and correction technologies. In addition, with the development of fields such as artificial intelligence and big data, asynchronous FIFOs will pay more attention to collaborative design with other hardware modules to meet the needs of complex systems.

In future asynchronous FIFO designs, new storage media such as memristors can be adopted to improve storage density and read-write speed. Combined with the idea of software-defined hardware, asynchronous FIFOs can be flexibly configured according to different application scenarios. New asynchronous clock domain synchronization technologies can also be explored to reduce the occurrence probability of metastability. With the continuous progress of technology, the application fields of asynchronous FIFOs will continue to expand. In addition to traditional electronic system fields, asynchronous FIFOs may also play an important role in emerging fields such as biomedicine and quantum computing.

6. Conclusion

As the scale of digital systems continues to expand, multi-clock domain design has become increasingly common, highlighting the increasing importance of asynchronous FIFO. In many fields such as communication, computers, image processing, and embedded systems, asynchronous FIFO has a wide range of applications. For example, in

ISSN 2959-6157

communication systems, it ensures that data from different clock domains can be stably buffered and transmitted; in image processing, it can cache image data to match the working speeds of different processing modules; in embedded systems, it helps coordinate the asynchronous operations between various subsystems.

In terms of architectural design, traditional asynchronous FIFO design architectures center around dual-port RAM as the core storage unit, combined with read and write pointers and other control logic. However, these designs may encounter pointer synchronization issues during high-speed data transmission and do not make full use of storage resources. Improved architectures adopt Gray code pointers, multi-clock domain synchronization techniques, and other measures to enhance stability and reliability, as well as optimize the utilization of storage resources. In terms of implementation technology, hardware description languages such as Verilog and VHDL play a key role in the design of asynchronous FIFOs and can be implemented on various hardware platforms such as FPGAs and ASICs. In terms of optimization strategies, area optimization can be achieved by streamlining logical structures and optimizing the design of storage units. Speed can be improved by optimizing pointer synchronization mechanisms and adopting pipeline technologies. Power consumption can be reduced by using low-power hardware components and optimizing circuit designs. In terms of reliability, asynchronous clock domains make data transmission across domains susceptible to metastability, high noise interference, and errors in multi-module interactions. In terms of balancing performance and resource consumption, modern electronic systems demand high performance, while resource-constrained environments require controlled resource consumption. New application scenarios have special demands. In the future, asynchronous FIFO technology will have several development trends. It will increase integration, capacity, and collaborative design while reducing power consumption and improving error correction technology. The innovation directions for asynchronous FIFO include new types of storage media, software-defined hardware, and new synchronization technologies. Additionally, its application fields will expand into emerging fields..

References

[1] Qiao Guangxin, Yu Shuiyou, Wang Xuebo. EMIF interface test method based on DSP+FPGA signal processing platform // Tianjin Electronic Industry Association. Proceedings of the 2024 Annual Meeting of Tianjin Electronic Industry Association. Tianjin 712 Communication Broadcasting Co., Ltd.; Aviation Military Representative Office in Tianjin Region of Land Equipment Department;

[2] Fan Ping. Design and development of helicopter vibration monitoring system based on FPGA+DSP architecture. Engineering & Test, 2024, 64(02): 92-95.

[3] Wang Gang, Chen Li, Chen Zhiyi, et al. Receiver design based on domestic FPGA+DSP. Electronics Quality, 2024, 04: 31-35.

[4] Sun Lin. Design and implementation of a three-phase asynchronous motor inverter control system based on FPGA. Dalian Jiaotong University, 2023.

[5] Chen Tingting, Lu Feng, Wan Shuqin, et al. Design of an asynchronous FIFO circuit with controllable delay. Microelectronics, 2022, 52(01): 42-46.

[6] Li Shuo, Lu Zhonghua, Sun Yongze. Distributed parameter optimization scheduling strategy and system design based on Mesos. Information Technology and Applications in Scientific Research, 2019, 10(02): 20-30.

[7] Zhang Xu. Research on performance optimization strategy of multi-axis motion control system with multi-processors. Guangxi University, 2018.

[8] Sun Yan, Tang Shaoju, Luo Hong. Design and implementation of gateway for multimedia sensor networks based on FPGA. Acta Electronica Sinica, 2012, 40(04): 625-631.

[9] Peng Yao, Zhou Duan, Yang Yintang, et al. Design of high-speed circular FIFO. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(03): 488-495.