Design of Asynchronous FIFO with Adjustable Input/Output Bit Width Based on Verilog

Langhe Tian

School of Information Science and Technology, Beijing University Of Technology, Beijing, 100124, China
3267264949@emails.bjut.edu.cn

Abstract. An asynchronous First-In-First-Out (FIFO) is a fundamental data storage and buffering mechanism designed to operate across different clock domains, where the read and write operations are driven by separate, and asynchronous clock sources. This characteristic makes asynchronous FIFOs an essential component for enabling reliable data transmission between circuits with mismatched or independent clocks. The primary focus of this study, implemented in Verilog, is the design, development, and evaluation of an asynchronous FIFO tailored to address challenges in cross-clock domain communication. In addition to supporting the standard FIFO operations, this work introduces an innovative bit-width conversion function. This function allows the FIFO to adapt to varying data requirements by either concatenating or truncating data based on the target application's needs. Such a feature significantly enhances the FIFO's flexibility, enabling it to accommodate diverse data sizes and formats, particularly in complex systems. Furthermore, the modified FIFO is optimized for high-speed data transfers and ensures compatibility between systems that utilize differing data widths, broadening its applicability. These improvements position the asynchronous FIFO as a versatile and efficient solution for advanced cross-clock domain data management in modern digital systems.

Keywords: Asynchronous FIFO, Cross-Clock Domain, Bit-Width Conversion, Data Width Adaptation.

1 Introduction

1.1 Research background

In recent years, the rapid advancement of integrated circuit technology has led to the widespread adoption of systems incorporating multiple clock domains [1]. In current CMOS process designed integrated circuits, it is almost impossible to achieve global clock synchronization due to the impact of timing instability [2]. These clock domains may exhibit either correlated or uncorrelated relationships. When signal transmission occurs between different clock domains, Clock Domain Crossing (CDC) becomes an inevitable challenge [3]. The transmission or storage of digital signals across clock domains, if not properly managed, can lead to metastability state [4]. If the metastable

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_70

state is not handled properly, it may bring risks such as data loss, dislocations, and re aggregation to the circuit, resulting in the loss or error of critical equipment functions. Such as loss of clock check circuit function, abnormal aircraft display images, Avionics Full Duplex Switched Ethernet (AFDX) switch data errors, etc [5]. Consequently, ensuring the efficiency and accuracy of data transmission has emerged as a critical research focus.

Asynchronous FIFO technology, utilizing dual-port Random-Access Memory (RAM) as an intermediate memory buffer, used to implement data buffering and flow control in asynchronous systems, has proven to be an effective solution for cross-clock domain operations [6]. This technology finds extensive applications in various fields, including signal processing, radar systems, and multimedia technology, demonstrating its versatility and importance in modern electronic systems.

1.2 The structure of asynchronous FIFO

A FIFO is a data storage buffer that operates on the principle of sequential data access. Unlike conventional memory systems, FIFO utilizes counters to automatically increment addresses for data read/write operations, which inherently restricts its functionality to sequential access rather than random access [1]. Illustrates the internal architecture of an asynchronous FIFO, which primarily consists of five key components: a RAM module--this component is implemented as a pseudo-dual-port RAM, which allows simultaneous read and write operations using different ports, responsible for data input, storage, and output operations; write control unit--this unit generates the write pointer and full flag signal, which determines whether data can be written to the FIFO. Writing is permitted when the write enable signal is active and the FIFO is not full; read control unit--this unit generates the read pointer and empty flag signal, which determines whether data can be read from the FIFO. Reading is allowed when the read enable signal is active and the FIFO is not empty; two clock synchronization interfaces--these interfaces synchronize the write pointer to the read clock domain for empty flag determination. Similarly, they synchronize the read pointer to the write clock domain for full flag detection. This architecture enables reliable data transfer between different clock domains while maintaining data integrity and preventing overflow or underflow conditions.

1.3 Adjustable input/output bit width design

Modern circuit systems often face scenarios where the input and output data have different bit-widths. To address this challenge, this paper presents a design based on Verilog HDL that extends the functionality of conventional asynchronous FIFO by incorporating bit-width conversion capabilities. The proposed solution enables seamless data transfer by implementing a bit-width adjustment mechanism that aligns input and output bit-widths within the asynchronous FIFO module. This enhanced design not only maintains the essential features of traditional asynchronous FIFO but also provides additional flexibility in handling diverse data formats, making it

particularly suitable for contemporary complex system architectures where data width compatibility is frequently required.

2 Literature Review

2.1 The Specific Application of Asynchronous FIFO

Due to the widespread application of asynchronous FIFO, optimizing its structure and function to improve efficiency and adaptability is essential for meeting diverse field requirements. Tan Chao et al. combined asynchronous FIFO with a 32-bit Microcontroller Series By STMicroelectronics (STM32) and Time-Interleaved Analog-To-Digital Converter (TIADC), applying it to fiber Bragg grating sensors in power systems. By using asynchronous FIFO for data caching, their design ensures high-speed storage and real-time processing of signal data, achieving high sampling rates and low power consumption at a low cost. The system they developed has a storage depth of 16K and a maximum sampling rate of 200 Msamples/s, making it suitable for high-speed data acquisition tasks [7]. However, despite these advantages, the design by Tan et al. faces challenges in practical applications. Data characteristics, such as bit width and volume, can vary significantly. When the input data bit width exceeds the maximum input bit width of the asynchronous FIFO, or when the stored data exceeds the FIFO's depth, data loss and overflow issues may occur. These limitations highlight the need for further research into dynamic adjustment mechanisms for FIFO depth and bit width to enhance its adaptability and reliability in diverse application scenarios.

In addition to its application in large-scale communication systems, asynchronous FIFOs have also found extensive use in data acquisition and transmission. Guoming Sung et al. demonstrated the application of asynchronous FIFOs in Universal Serial Bus (USB) systems, specifically in the context of FPGA-to-FPGA communication [8]. Their work integrated an asynchronous FIFO with a Serial Interface Engine (SIE). This combination facilitated efficient packet transformation and data transmission. This design significantly reduced the delay times associated with data serialization and deserialization, thereby enhancing the overall transmission speed. The asynchronous FIFO used in their study had a capacity of 2KB, which was sufficient for handling simple data transmission tasks. However, in more complex applications, this capacity might be insufficient, leading to overflow issues that could be mitigated by dynamic depth adjustment or enhanced error-handling mechanisms. Future research could explore methods to dynamically adjust FIFO depth or implement more robust error-handling mechanisms to address these challenges.

2.2 Functional Design of Asynchronous FIFO

In addition to studying the changes in requirements of asynchronous FIFOs in specific applications, improving capacity and adaptability is also a key focus of research. Keyu. Liu. Proposed a novel approach to expanding the capacity of asynchronous FIFOs by cascading them with synchronous FIFOs [9]. This design leverages the asynchronous FIFO for initial cross-clock domain data processing, followed by caching and re-

outputting through synchronous FIFOs, effectively increasing FIFO depth while maintaining data integrity across different clock domains. This cascading method offers a relatively simple solution for capacity expansion. However, the study primarily focuses on the basic architecture and simulation verification, lacking detailed exploration of further capacity expansion techniques. Additionally, it does not address power consumption or other practical considerations during hardware implementation. Future work could explore more advanced methods, such as hybrid FIFO designs or low-power hardware architectures, to enhance capacity and optimize power efficiency.

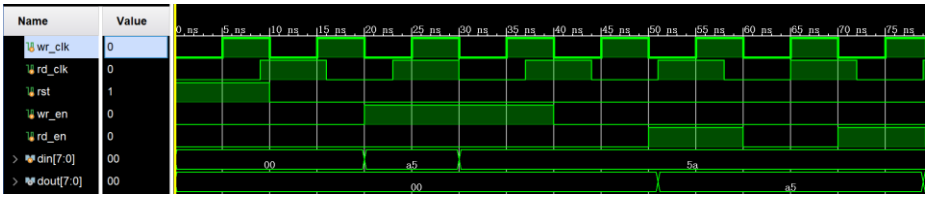
Xiangjie. Li. et al. introduced a novel approach to dynamically adjusting the depth of asynchronous FIFOs by incorporating a top-module in Verilog code design [10]. This method allows for flexibility in FIFO depth adjustment by modifying parameters within the top module, which is particularly useful for accommodating varying data burst sizes and clock frequencies. Additionally, the authors provided an estimation method for calculating the minimum depth of FIFOs when the write clock frequency exceeds the read clock frequency, ensuring efficient data handling and preventing overflow or underflow conditions. However, the depth adjustment relies heavily on static parameter modification, which may limit its adaptability in dynamic real-time applications where runtime adjustments are necessary. Future work could explore more dynamic and flexible methods for depth adjustment to enhance the practicality and efficiency of asynchronous FIFOs in complex systems.

3 Results and Discussion

3.1 Simulation Results and Analysis

Simulate the basic functions of asynchronous FIFO and the input/output bit width conversion function:

Firstly, the test and simulation results of the basic functions of asynchronous FIFO are shown in Fig. 1. Under the conditions of writing clock wr_clk at 100MHz, reading clock rd_clk at 50MHz, and FIFO depth 16, continuously write hexadecimal numbers 12, 34, 56, 57, 5a to 5f, and 60 to 65. It can be seen that after the last bit of data 65 is written, the FIFO is full, full=1. Under the control of the read clock rd_clk and the read enable rd_en, 16 hexadecimal numbers are read out sequentially in the order of input 1 until the last bit 65 is read out, and the FIFO reads empty, empty=1.



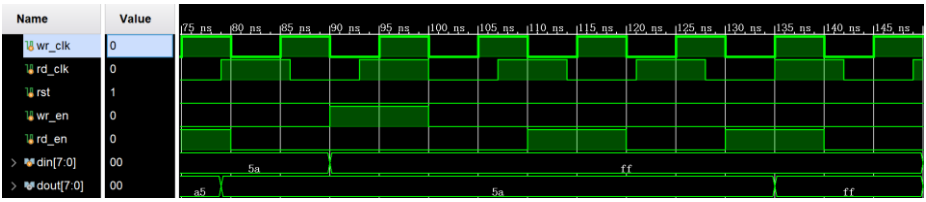


Fig. 1. Asynchronous FIFO Basic Function Test (Photo/Picture credit: Original)

Next is to test the input/output bit width conversion function. Test the data clipping function first. When the input bit width exceeds the output bit width, the FIFO crops the input data to meet the output requirements. As shown in Fig. 2, the input bit width is set to 16 and the output bit width is set to 8. Under the control of rd_clk and rd_den, input two hexadecimal 16 digit numbers 1234 and 5678 in sequence. As can be seen, when rd_den is valid and rd_clk arrives, the FIFO sequentially reads 34, 12, 78, and 56. Then test the data stitching function. When the input data bit width is smaller than the output data bit width, the FIFO will concatenate the input data until the concatenated data bit width is greater than or equal to the output bit width before outputting. As shown in Fig. 3, the input and output data bit widths are configured as 8 and 16, respectively. Under the control of rd_clk and rd_den, input four hexadecimal 8-bit numbers 12, 34, 56, and 78 sequentially. As can be seen, when rd_den is valid and rd_clk arrives, the FIFO sequentially reads 3412 and 7856.

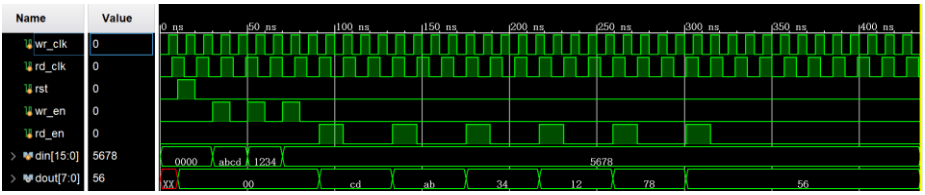


Fig. 2. Data clipping function test (Photo/Picture credit: Original)

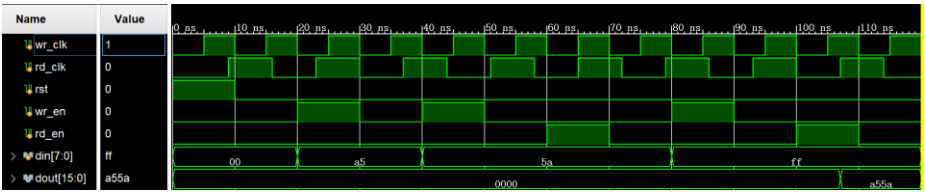


Fig. 3. Data stitching function test (Photo/Picture credit: Original)

3.2 Discussion

This study introduces a bit-width conversion function to enhance the Verilog-based implementation of an asynchronous FIFO buffer. The added function is controlled by the standard read-write clock, read-write enable, and full-empty flags, enabling more flexible data handling. The core of this function is to manage the discrepancy between

input and output bit widths, a challenge commonly encountered in asynchronous FIFO designs. Before writing data into the FIFO, the bit width of the input and output is determined by a temporary variable, which temporarily stores incoming data for further processing. In cases where the input data bit width is larger than the output bit width, the function module concatenates the input data sequentially into the temp_data variable. Once the concatenated bit width reaches or exceeds the output bit width, the function module extracts the required number of bits. It then stores this cropped data into the FIFO buffer. Conversely, if the input bit width is smaller than the output bit width, the function module stores the incoming data directly into temp_data and performs the necessary cropping before placing it into the FIFO. This approach ensures that the FIFO can handle a variety of input and output bit-width configurations while maintaining data integrity.

The proposed design significantly enhances the versatility of asynchronous FIFOs, making them suitable for more complex data processing environments. This flexibility opens up possibilities for a wider range of practical applications where input and output data bit widths may vary. However, the integration of the bit-width conversion function, along with FIFO control signals and clock synchronization, introduces some challenges. Specifically, to mitigate the risk of metastability, two-stage synchronizers are often added. While this effectively reduces the occurrence of metastable states, it also results in a longer time delay between the input and output signals, thus decreasing the FIFO system's responsiveness to rapid data changes. Moreover, the increased complexity of the FIFO structure necessitates more intricate control signals, which raises the likelihood of design errors. These challenges are primarily attributed to the logical constraints of the Verilog language, particularly its reliance on constants when defining variable bit widths, which complicates the design process. As a result, this leads to the appearance of multiple functions across different designs, which may complicate implementation and maintenance. Despite these limitations, the addition of the bit-width conversion function significantly improves the functionality and adaptability of asynchronous FIFO buffers, making them more capable of handling a broader range of data widths in real-world applications. Future work could explore optimizing the design to minimize control complexity and further mitigate metastability-related issues.

4 Conclusion

This study explores the design and implementation of asynchronous FIFOs with adjustable input and output bit widths, a solution that addresses the increasing complexity of integrated circuits. The proposed architecture adds function for concatenation and cropping, providing flexibility in adjusting input and output bit widths. By integrating these features, the asynchronous FIFO can be tailored to meet the requirements of various applications, making it suitable for more complex scenarios. The ability to adapt the bit width provides significant advantages in terms of performance optimization and resource utilization, ensuring that the FIFO can handle varying data sizes efficiently. This innovation enables broader applications by

addressing the diverse data transfer requirements in complex systems. Overall, the research demonstrates how adjustable asynchronous FIFOs can improve the scalability and versatility of integrated circuit designs in modern electronic systems.

References

1. Da-Chao, J., Jian-Wei, L.: 'Implementation of Signal Synchronization in Asynchronous Clock Domains', J. Tianjin Univ. Technol., 2017, 33, (3), 40-44
2. Jia-He, Z., Run-Quan, S., Wei-Chao, X., Zhang, L.: 'Design of Lightweight Multi-Master-Slave Cross-Clock Domain On-Chip Bus Based on OCP', Appl. Electron. Tech., 2023, 49, (2), 45-49
3. Huan-Qi, G., Ya-Jie, Y., Jun-Feng, G.: 'Synchronization Method for Bit Data Transmission Across Clock Domains', Mod. Comput., 2020, 24, (24), 9-14
4. Hong-Ke, L., Qing-Chun, W., Shun-Yuan, Y.: 'Design of Asynchronous FIFO Based on Verilog HDL', Electron. Des. Eng., 2021, 29, (19), 107-111
5. Wen-Cheng, J., Song-Ren, H.: 'Design and Verification of Asynchronous FIFO Based on Chisel Language', Electron. Packag., 2024, 24, (9), 70-74
6. Chao, T., Jun-Ming, D., Liang, X., Chen, W.: 'Design of High-Speed Data Acquisition System Based on STM32 and Asynchronous FIFO in Ping-Pong Mode', Electr. Eng. Mater., 2023, 1, (1), 55-59
7. Yu-Yang, F., Zhi, D., Zi-Hang, L.: 'Verification and Reliability Analysis of Airborne Electronics Cross-Clock Domain Synchronization Circuit', J. Northwest. Polytech. Univ., 2022, 40, (2), 369-376
8. Sung, G.M., Tung, L.F., Wang, H.K., Lin, J.H.: 'USB Transceiver With a Serial Interface Engine and FIFO Queue for Efficient FPGA-to-FPGA Communication', IEEE Access, 2020, 8, 69788-69799
9. Liu, K.: 'A Verilog design for cascading a large-capacity asynchronous FIFO with a synchronous FIFO', Appl. Comput. Eng., 2023, 14, 25-30
10. Li, X.J., Fu, S.M., Wang, X.B., Wang, Y.Q.: 'Design of asynchronous FIFO with adjustable depth based on Verilog', Appl. Comput. Eng., 2023, 13, 282-287