Optimizing Data Transfer Across Asynchronous Clock Domains: A Comprehensive Approach to Asynchronous FIFO Circuit Design

Xipeng Zhang *

School of Electrical and Electronic Engineering, Wenzhou university, Wenzhou, 325006, China

* Corresponding Author Email: 19211740122@stu.wzu.edu.cn

Abstract. As integrated circuit (IC) design continues to advance in scale, there is a growing trend towards integrating more components onto the same IC board, resulting in larger circuit boards. These expanded circuit boards often encompass diverse asynchronous clock domains, leading to inevitable data transfer challenges between these distinct clock domains. This research article focuses on Verilog-based asynchronous first in first out (FIFO) design as a solution to effectively address this issue and offers a robust design approach to enhance circuit integration and stability. By employing conventional asynchronous FIFO design principles, such as the utilization of binary conversion Gray code to synchronize clock operations for pointer-to-address mapping and the incorporation of empty/full flag status bits to facilitate empty/full signal generation determination, seamless data transfer between various asynchronous clock domains is successfully achieved. Furthermore, this article includes comprehensive testing and simulation using Testbench to validate the asynchronous FIFO circuit, with successful results. The primary objective of this article is to present a well-established and comprehensive design methodology for asynchronous FIFO circuits, equipping designers with a mature and reliable approach to tackle the challenges posed by increasingly complex IC designs.

Keywords: Verilog; IC; asynchronous FIFO; Gray code.

1. Introduction

The design scale of IC is increasing with the development of technology, and under the development of Moore's Law, more and more components are integrated on the same circuit board. This design also makes the size of the circuit board increasingly large, making it increasingly difficult to synchronize the design of these circuits. The usual circuit synchronization design methods are gradually being replaced by some new design methods, such as overall asynchronous local synchronization (GALS structure). In such a design method of overall asynchronous and local synchronization, a circuit board often contains multiple clocks that are different. During the operation of the circuit, data must also undergo transmission between different clock domains. At this point, under the premise of system stability, how to smoothly transmit data has become an important issue. The general method to address this issue is to design an asynchronous FIFO at the intersection of two clock domains to achieve data exchange. Asynchronous FIFO is a first in, first out storage circuit used to store and buffer data transmission between two asynchronous clock domains. Because asynchronous FIFO effectively achieves data buffering while transmitting data to each other, this is an ideal method [1, 2].

Based on this premise, this study provides a design method for asynchronous FIFO and also provides a Testbench test circuit to verify the feasibility of the asynchronous FIFO circuit using Modelsim. The design of asynchronous FIFO aims to better transfer data between different clock domains, and it can also effectively prevent metastable phenomena caused by data transmission and storage across clock domains. By utilizing its reasonable asynchronous clock design with different frequencies and phases, the circuit can become more stable and integrated, making it particularly important for the circuit itself. The asynchronous FIFO circuit consists of several different modules, such as the read/write address conversion module, the empty/full mark generation module, and the RAM module. By using clocks of different frequencies and phases on the read and write pointers, the

judgment of the empty/full state of the read/write address bar is completed. At the same time, the asynchronous FIFO also uses a binary to Gray code conversion circuit to complete the transmission of address data between two clocks. The converted Gray codes of the two clocks are compared with each other to determine the change in address, and to achieve the judgment function of reading and writing the empty and full status of the address bar. In addition, asynchronous FIFO has a wide range of applications in fields such as radar, digital media, signal processing and analysis [3]. The expected keywords for searching literature in this study include asynchronous FIFO, IC integrated circuit, asynchronous clock, etc . This article will first elaborate on the basic knowledge of asynchronous FIFO, such as principles, design difficulties, etc., and then provide a specific description and analysis of the proposed asynchronous FIFO design. Finally, based on the design, simulation results will be obtained and summarized.

2. Principle and Application Explanation

2.1. Explanation of the principle of asynchronous FIFO

As shown in Figure 1, it is the internal module construction diagram of asynchronous FIFO. In the entire asynchronous FIFO internal module construction diagram, the left side is the write clock and write address and full load signal generation module, and the right side is the read clock and read address and empty signal generation module [4].

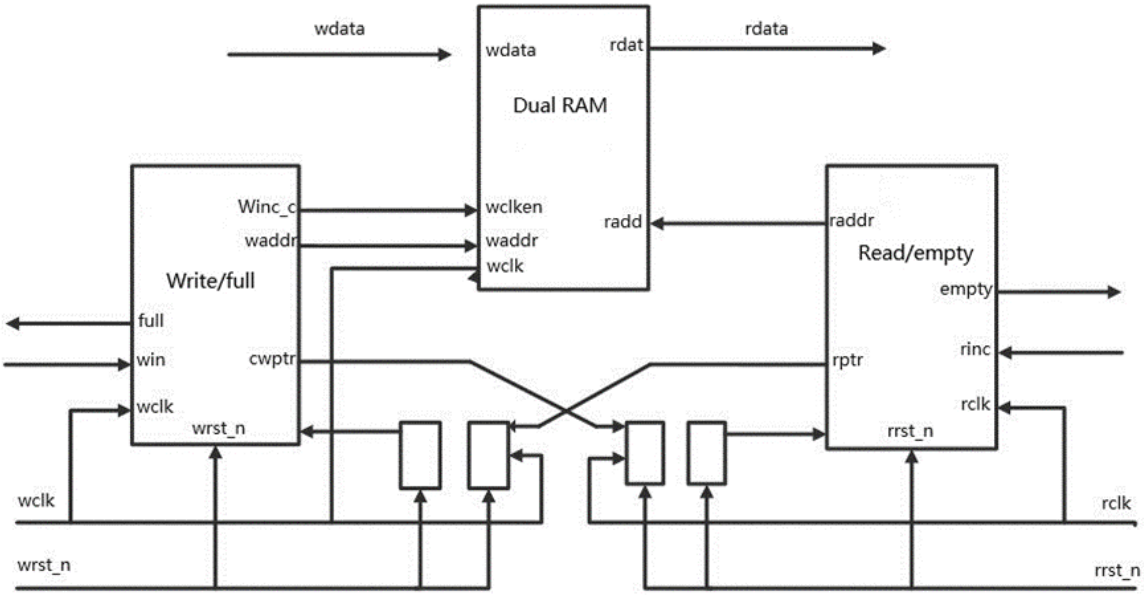

Fig 1. Asynchronous FIFO internal module construction diagram.

The entire asynchronous FIFO module completes the complete write and read process as follows: In order to complete the complete write and read process, it is necessary to reset the entire asynchronous FIFO before this, When FIFO is reset, both the write and read pointers are 0, while the full signal is invalid and the empty signal is valid. When a write signal input becomes effective, the write pointer writes data to an address with each beat of the write clock, and adds one to the write address. The write pointer points to the next address to be written, facilitating subsequent write operations. Repeat this process until the full signal becomes effective or the write signal stops, and then stop writing. When a read signal input becomes effective, the read pointer follows each beat of the read clock, read data from one address separately and subtract the read address by one. The read pointer points to the next address to be read, facilitating subsequent reading work. Repeat this process until the null signal is valid or the read signal stops, and then stop reading work. The entire asynchronous FIFO relies on these two key modules to complete the first in, first out write and read data operations.

2.2. The Difficulties and Corresponding Solutions of Implementing Asynchronous FIFO Based on Verilog

There are two main difficulties in implementing asynchronous FIFO based on Verilog. Firstly, during the process of building FIFO, the registers in the FIFO are prone to metastable states, making the circuit's operating state unstable. Secondly, it is difficult to transfer data between different asynchronous clock domains. The next step in this article is to elaborate on how to solve these two major problems.

2.2.1. Metastable state

Metastable state is the most important issue of data transmission between different clock domains. When a data signal passes through the intersection of two asynchronous clock domains, there will be two different asynchronous clocks to control the value of the signal. At this time, if the sensitive edges of the two signals are very close and exceed the allowed limit, instability of the data signal will occur, that is, the circuit will enter metastable state [5]. In addition, this metastable phenomenon generally occurs on registers in circuits, as in digital integrated circuits, registers generally meet two timing requirements: establishment time and hold time. As shown in Figure 2, the establishment time is the minimum time for the data input to remain stable and effective before the rising edge of the clock, while the holding time is the minimum time for the data input to remain stable and effective after the rising edge of the clock [6]. If the data signal passes between two asynchronous clock domains and the data signal and clock do not meet this timing requirement, it will cause the register to operate in an unstable state, the digital signal will be fixed at a random high or low level irregularly, and this register enters a metastable state [7]. For this situation, the usual practice is to add a two-stage synchronizer after the register that will appear in the metastable state, which can greatly avoid such situations. The function of the synchronizer in the circuit is roughly to use its own circuit structure to stabilize the sampled signal in the circuit from a metastable state and ultimately become a fixed 0 or 1 output result. It itself cannot accurately determine what the sampling result is, it can only randomly output a stable result signal.

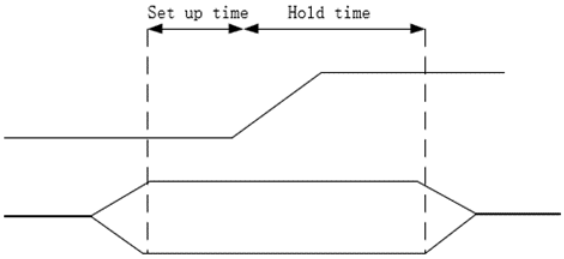

Fig 2. Register establishment time and hold time.

2.2.2. Data transfer between different asynchronous clock domains

When an asynchronous FIFO is located between two different asynchronous clock domains, its write and read addresses need to undergo certain synchronization operations before being transmitted and compared. But the read address obtained after synchronization operation may have no relation to the previously used read address. This is the difficulty in data transmission between different asynchronous clock domains. To solve this phenomenon and problem, the usual solution is to convert the original binary data in the circuit into Gray code and then perform synchronous operations. This can greatly improve the accuracy of reading addresses after synchronous operations. The characteristics and manifestation of Gray code, in simple terms, is a coding method where only one bit changes between adjacent items [8]. Therefore, when performing synchronous operations on it in a circuit, there is at most one possibility of metastable state occurring. Under this premise, if the circuit needs to lock the read address after synchronization operation, the locked address is either the current address or the address of the previous bit of the current address. It can be seen that this method can greatly reduce the error rate of data transmission between different asynchronous clock domains and improve the accuracy of data transmission through the application of Gray code.

2.3. The Generation Process of Empty/Full Flag

How to correctly generate the empty/full flag is also a key focus in every FIFO design. Generally speaking, the logic for generating the empty/full flag is: no overflow when the FIFO is full, and no more reads when the FIFO reads data and determines it to be empty [9]. The design concept of empty/full flag judgment in this article is roughly based on the traditional design concept of empty/full flag judgment. The specific content is expressed as follows: to prevent asynchronous FIFO from misoperation, it is necessary to set the empty/full flag status bit, naming the empty/full flag bits empty and full respectively. When empty=1, FIFO cannot read data, and when full=1, FIFO cannot write data again [10]. Because the asynchronous FIFO designed in this article has a bit width of 8 bits, which is the third azimuth bit width of 2, at least three binary digits are required to represent the address bits.

The read null condition for this asynchronous FIFO is set as follows: write pointer (wr_pointer) = read pointer (rd_pointer), as shown in Figure 3 (a) below. As mentioned earlier, because the read and write pointers work under different asynchronous clocks, it is necessary to convert the read and write pointers to the same clock domain for comparison. The read clock domain where the read pointer is located is converted to the write clock domain. When the read pointer is equal to the write pointer (w2r bincnt==rdunbincint), empty=1, and the FIFO is in a read empty state.

The write full conditions for this asynchronous FIFO are set as follows: For write full operations, there will be two scenarios for discussion: (1) Write all at once. When writing multiple data in the clock domain until the binary pointer points to the last register storage unit, (wr_pointer=15) & (rd_pointer=0), full=1 asynchronous FIFO is written full, as shown in Figure 3 (b); (2) Multiple intermittent writes to full, each time only a portion of data is written to the asynchronous FIFO, and after multiple asynchronous FIFO writes to full. For example, the first time only 10 data are written, 3 data are read, and the second time 9 data are written, the asynchronous FIFO is filled. When the write pointer in the write clock domain points to the nth register storage unit, and the pointer is synchronized to the binary write pointer in the read clock domain pointing to the nth+1 register storage unit (wr_pioner=n) & (rd_pointer=n+1) in the presence unit, full=1, the asynchronous FIFO is written to full, as shown in Figure 3 (c).

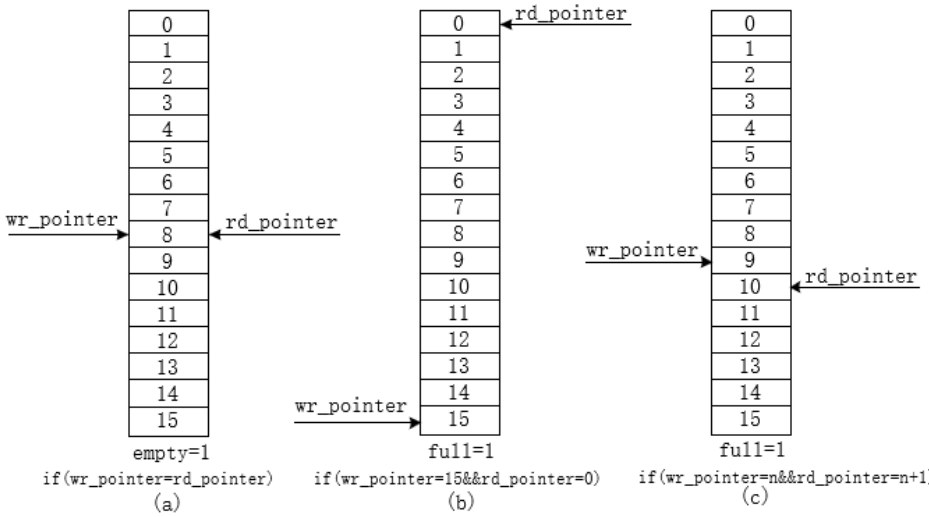

Fig 3. Asynchronous FIFO empty/full state generation logic diagram.

3. Specific Framework and Module Analysis of Asynchronous FIFO Design

As shown in Figure 4, the specific design framework diagram of the circuit is shown after using the RTL viewer of Quartus II. It is not difficult to see that the sync_ W2r and sync_ R2w is the synchronous clock domain module for writing pointer conversion binary Gray code and the synchronous clock domain module for reading pointer conversion binary Gray code, respectively_

Empty and wptr_ Full refers to the empty signal generation module and the full signal generation module, while the remaining DualRAM is a FIFO dual port RAM design module. Each module is correctly connected to each other and is called and allocated by the top-level module (FIFO memory allocation module), successfully achieving asynchronous FIFO circuit function. Due to limited space, only some of the main modules will be analyzed in detail below.

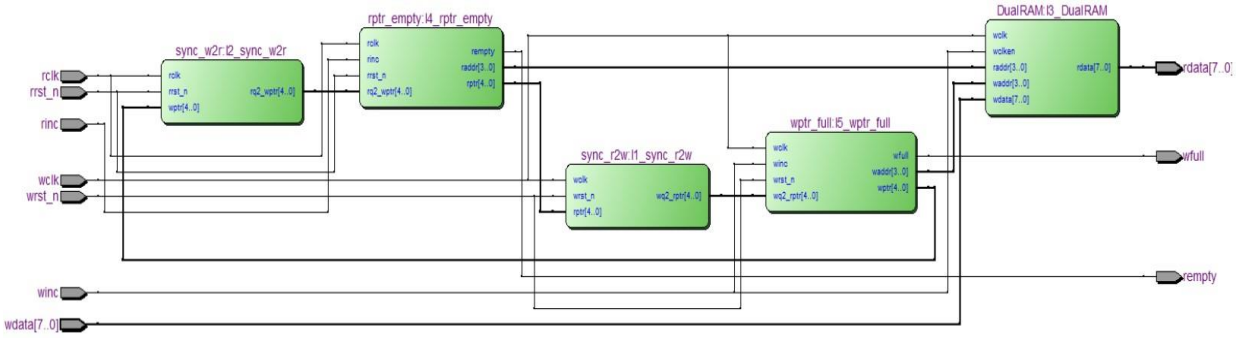

Fig 4. Asynchronous FIFO specific design framework.

3.1. Write/Read Pointer Conversion Binary Gray Code Synchronous Read/Write Clock Domain Module

The write/read pointer conversion binary Gray code synchronous read/write clock domain module is one of the key modules in the circuit, which can achieve data transfer between different asynchronous clock domains and correctly synchronize the address or data being written/read by both the write clock and read clock.

3.2. Empty/full signal generation module

The judgment method of the empty/full signal generation module is to set the empty/full flag status bit, so that the read/write pointer is one bit wider than the bit width required for asynchronous FIFO addressing, thereby achieving empty and full empty/full flag signal generation judgment work.

4. Comprehensive simulation verification results

As shown in Figure 5, this is a screenshot of the simulation results obtained on Modelsim after running the corresponding Testbench program on the circuit. Due to the limited space in this article, the specific analysis of the Testbench program content will no longer be overly expanded. It is not difficult to see that in the waveform output of the simulation results, when the reset is cancelled (the reset signal is low and effective), and after the enable winc is high and effective, the write address begins to accumulate continuously, Then the write data also began to change, and the empty mark of the memory also began to become non empty after a few beats (there is an asynchronous conversion written to the read side). After the read enable ring is raised and effective, the read data also starts to change in the next beat, and the read address also starts to continuously increase by one.

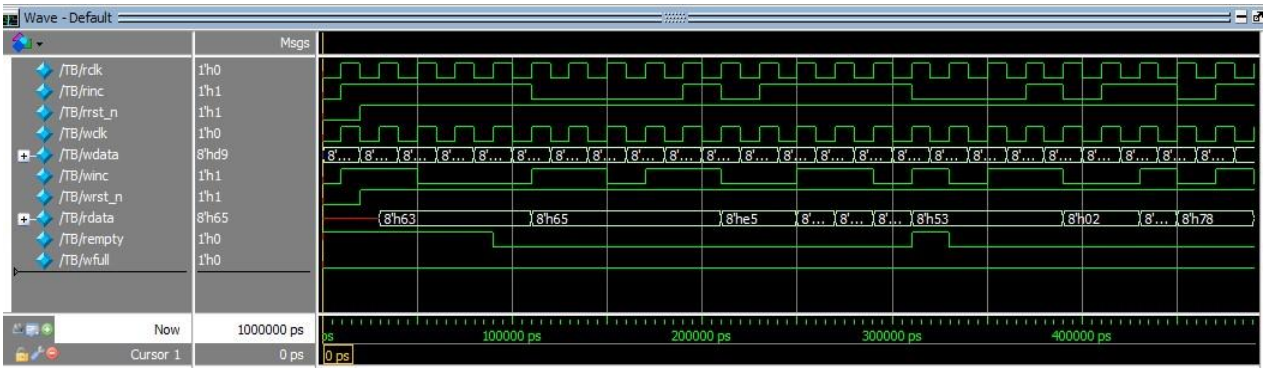

Fig 5. Simulation verification result diagram.

5. Conclusion

By conducting in-depth analysis and thorough module representation, this article successfully realized the functionality of an asynchronous FIFO circuit using Verilog. The implemented asynchronous FIFO module boasts diverse practical functions. The incorporation of these modules into large-scale integrated circuit designs facilitates seamless data transfer between asynchronous clock domains, a critical factor in enhancing circuit integration and stability.

It is worth noting that the asynchronous FIFO design concept introduced in this article aligns with traditional asynchronous FIFO design principles. However, it's essential to recognize that traditional asynchronous FIFO designs are typically tailored for low operating frequencies and extensive system footprints, making them less suitable for large-capacity asynchronous FIFO circuits. The judgment method for empty/full flags may require further refinement when designing high-capacity asynchronous FIFO circuits, opening the door to future research opportunities in this domain. The aspiration is to continually enhance the asynchronous FIFO design circuit proposed in this article in the future.

References

[1] Jin Dachao, Leng Jianwei. Implementation of Asynchronous Clock Domain Signal Synchronization. Journal of Tianjin University of Technology, 2017,33 (03): 40-44

[2] Zhao Yang, Liang Buge, Yang Degui, et al. Design of Cross Clock Domain Synchronous Circuit in Multi Clock Systems. Electronic Technology Applications, 2018,44 (02): 6-9

[3] Wang S, Xu Y, Tang J, et al. Design of Asynchronous FIFO Controller Based on FPGA. International Core Journal of Engineering, 2021, 7 (1): 153-159

[4] Bu Xianxian. Verilog Design of Asynchronous FIFO. Computer and Digital Engineering, 2007 (06): 191-194+202

[5] Cai Fazhi, Su Jin, Ye Bing. Verilog HDL Design for Asynchronous FIFO. Instrument User, 2008 (03): 68-69

[6] Zhu Yongfeng, Lu Shengli, Mao Bangqin. Multi clock domain processing in SOC design. Electronic Engineer, 2003 (11): 60-63

[7] Rabaey JM, Chandrakasan A., Nikolic B., Translated by Zhou Runde. Digital Integrated Circuits: Circuits, Systems, and Design. Electronic Industry Press, 2004 (10): 237-240

[8] Zou Jiancheng, Li Guofu, Qi Dongxu. Generalized Gray Code and Its Application in Digital Image Scrambling. Journal of Applied Mathematics in Higher Education A (Chinese Edition), 2002 (03): 363-370

[9] Wei Fang, Liu Zhijun, Ma Kejie. Design and Implementation of Asynchronous FIFO Based on Verilog HDL. Electronic Technology Applications, 2006 (07): 97-99+106

[10] Li Hongke, Wang Qingchun, Yu Shunyuan. Asynchronous FIFO design based on Verilog HDL. Electronic Design Engineering, 2021,29 (19): 107-111+116