Dean&Francis

ISSN 2959-6157

Implementation of Pre Fetch Function FIFO

Jinyuan Li

Department of Communication Engineering, HuBei University, Wuhan, China

Corresponding author: xejua@ldy.edu.rs

Abstract:

When the data processing speed is mismatched or multitasking occurs, frequent data reads may lead to a decrease in the processing efficiency of FIFO. To solve this problem, an asynchronous FIFO circuit with a prefetching function was designed based on the traditional asynchronous FIFO circuit. This circuit adds an enable controller and output unit register on the basis of the original asynchronous FIFO module. The enable controller completes the conversion between the read pointer and null pointer of the ordinary FIFO and the pre-fetch FIFO, and the output register outputs the data as read data to achieve data pre-fetch. Using Modelsim for functional verification, simulations were conducted in various modes such as writing only one data read and continuous reading after completion. The simulation results showed that the design can quickly and accurately read data from the FIFO. The design of this module can effectively improve system performance and resource utilization and has a guiding role in the future development of fields such as network communication and image and video processing.

Keywords: Asynchronous FIFO; enable controller; output unit register; simulation verification.

1. Introduction

With the advent of the information age, integrated circuits, as the core of the information technology industry, have become a powerful engine for the electronics industry to move towards the digital age [1].

Asynchronous FIFO follows different clock sources through read and write operations, which can solve the synchronization problem between different clock systems and ensure the reliability of real-time data transmission. It plays an important role in digital communication, image processing, embedded system design and other fields[2]. There are two difficulties in asynchronous FIFO design. Firstly, how to avoid the occurrence of metastable states when read and write pointers are transmitted across clock domains. Secondly, how to correctly indicate when the FIFO is empty or full[3].

At present, asynchronous FIFO circuits with different functions have been designed for different needs, such as delay-controllable asynchronous FIFO circuits, large-capacity asynchronous FIFO circuits, etc. Or design specific FIFO modules for specific application scenarios, such as anti-SEU design for asynchronous FIFOs.

In practical applications, the real-time performance of asynchronous FIFO may be reduced when facing multi-tasking and data flow transformation. To solve the above problems, a research and design of an asynchronous FIFO circuit with a prefetching function is proposed to improve data reading efficiency.

2. Research Content and Ideas

2.1 General Architecture of Asynchronous FIFO

Asynchronous FIFO mainly consists of five parts: RAM, write control terminal, a read control terminal, and two clock synchronization terminals. The schematic diagram is shown in Fig 1. The dual port RAM is used for data storage and read/write operations. The write control terminal and the full signal generator are used to determine whether data can be written. The read control terminal and the empty signal generator are used to determine whether data can be read, and the read/write connections are established through two clock synchronization terminals.

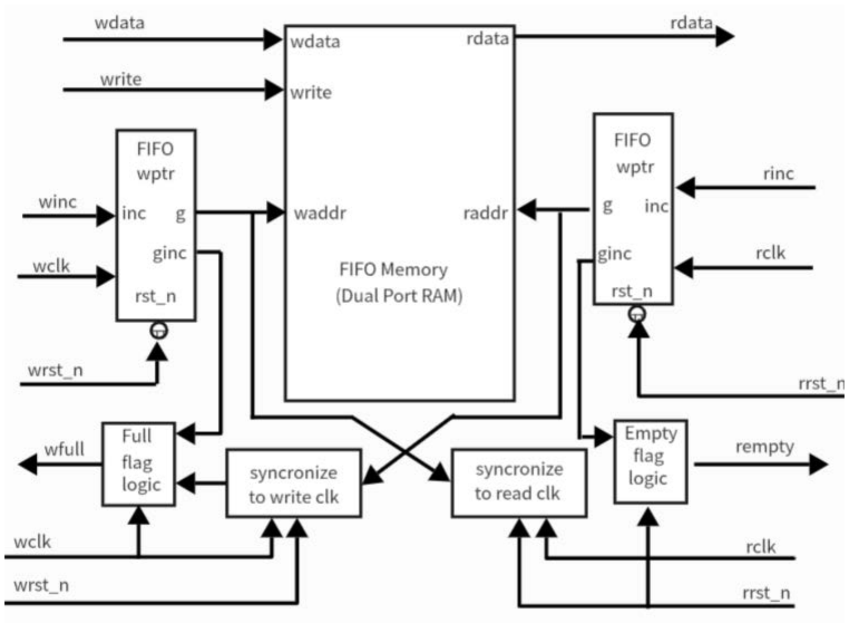

Fig.1 General architecture of FIFO

To achieve the prefetching function of the FIFO circuit, this paper adds a prefetching module on the basis of a traditional asynchronous FIFO circuit. The control diagram of the prefetching module is shown in Fig 2.

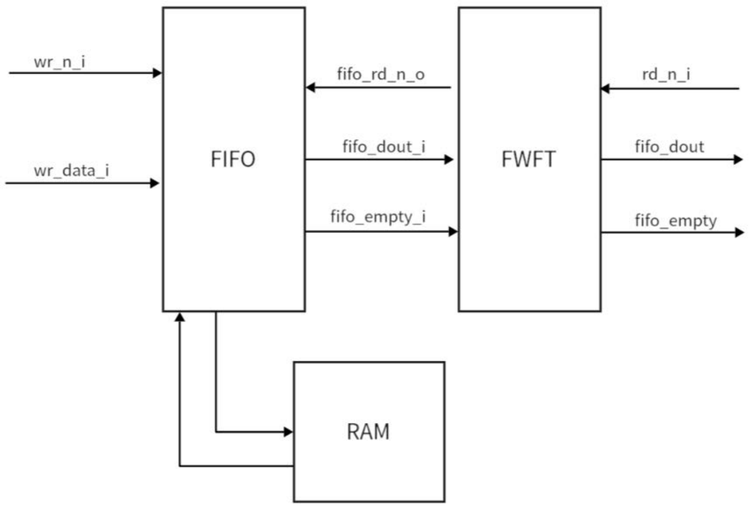

Fig. 2 FIFO design with prefetching function

ISSN 2959-6157

This module adds an enable controller and output register unit on the basis of the original FIFO. Enable the control register to complete the conversion between rd and empty in FIFO and FWFT FIFO. The output register uses the data output from FIFO or RAM as the read data rdata for FWFT FIFO.

When data is stored in RAM, the FIFO controller will increment the read address in RAM by 1 in advance. When no read signal is received, the output register will point the read address to the data address to be read. When the FIFO receives a read request instruction, it can read the current data rdata from RAM on the rising edge of the current clock cycle.

2.2 Metastable State

In the practical application of asynchronous FIFO, metastability is an important issue that needs to be addressed. In FIFO design, the presence of data in the FIFO is determined by the edempty and rdusedw signals before reading out the data. Due to the asynchronous nature of rdclk and wrclk clocks, there may be write operations that are relatively fast compared to the read/write clock, causing the read end to be judging the read/write address while the write end's write enable is valid, resulting in a metastable state when the write address is incremented by 1. This article uses Gray code addressing and two-stage flip flops to eliminate metastability. Only one bit between adjacent numbers in Gray code changes, which can greatly reduce errors that may occur during data transmission. The two-stage flip flops can prevent metastability in the second stage flip flops. The output result of metastable state cannot be completely eliminated in asynchronous circuits, but it can greatly reduce the probability of metastable state occurrence.

3. Research Methods and Processes

3.1 Design and Implementation of Circuits

The asynchronous FIFO circuit in this article consists of data storage, pointer synchronization, empty/full detection, and prefetching modules, with a depth of 16 and a data bit width of 16 bits. The design of modules such as data storage, pointer synchronization, and empty/full detection adopts previously existing design methods [4-5]. The design description of the prefetching module is as follows.

The storage module is composed of registers. Write the data to be input into the storage unit when the FIFO receives a valid write clock signal, and read the data from the storage unit when it receives a valid read clock signal. Every time data is read or written, the read and write addresses are automatically accumulated [6].

Due to the fact that read and write pointers operate in different clock domains, it is necessary to set up a synchronization module to synchronize the input signal and avoid violating timing requirements. In this design, a two-stage flip-flop is used to form a double buffer pointer with Gray code, thereby achieving synchronization of asynchronous input signals and avoiding the occurrence of metastable states caused by asynchronous clock read and write clock signals [7].

In practical applications, it is often necessary to process large amounts of data, and the data processing speed of FIFO is limited. If a FIFO with a prefetching function (FWFT FIFO) can be designed, the data can be pre-stored in advance. When a read signal is received, the data will be read at the rising edge of the signal, thereby reducing access delay and improving data throughput.

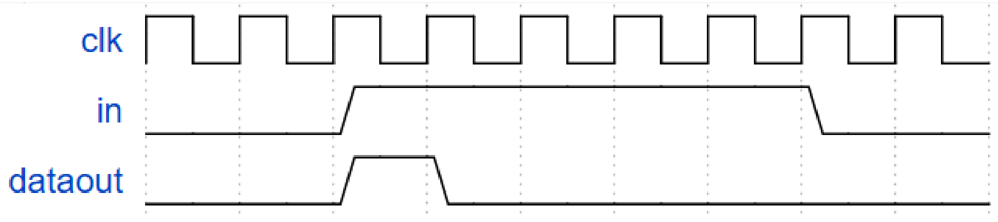

Fig.3 Pre fetch timing read

Simulate the prefetching timing diagram using WaveDrom Editor as shown in the fig 3. When the enable signal arrives and in arrives, the prefetching FIFO can immediately output the data.

The empty/full detection module is used to indicate the current status of the FIFO. When there is no data available for reading in the FIFO, the empty/full detection module will output a signal (empty) indicating that the FIFO is in an empty state. Prevent read operations without data, avoid data errors or undefined behavior. When the FIFO reaches its maximum storage capacity, the detection module will output a signal (such as full) indicating that the FIFO is in a full state. Prevent writing data again when the FIFO is full, to avoid data loss or overwrite. Reset the FIFO circuit when the circuit detects an empty or full state.

4. Research on Circuit Simulation and Analysis

4.1 Simulation Verification

This article uses Modelsim to simulate the designed circuit. By comparing the waveform diagrams of a regular FIFO and a FIFO with prefetching function, the conclusion can be drawn.

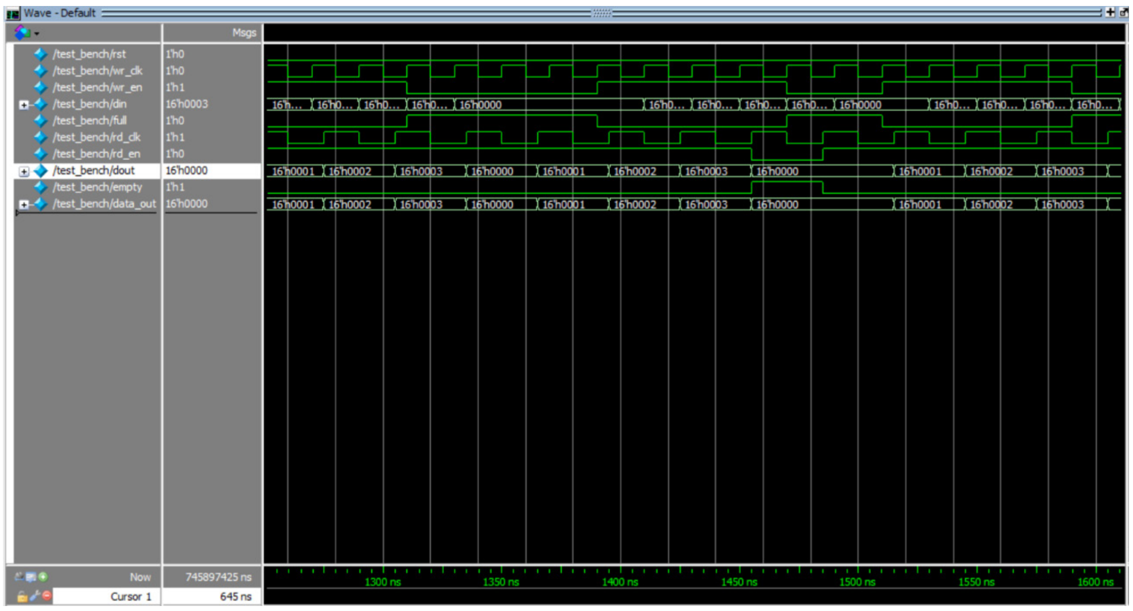

Fig.4 Simulation results of ordinary FIFO

This fig 4 shows the Verilog code implementation of a regular asynchronous FIFO. During the design process, two-stage register synchronization and Gray code were used to avoid the generation of metastable states. This design tends to conservatively fill the FIFO, but does not generate errors and only slightly affects its performance. As shown in the figure, when the FIFO receives the rd signal, it will generate dout as data output at the end of the current beat and the beginning of the next beat.

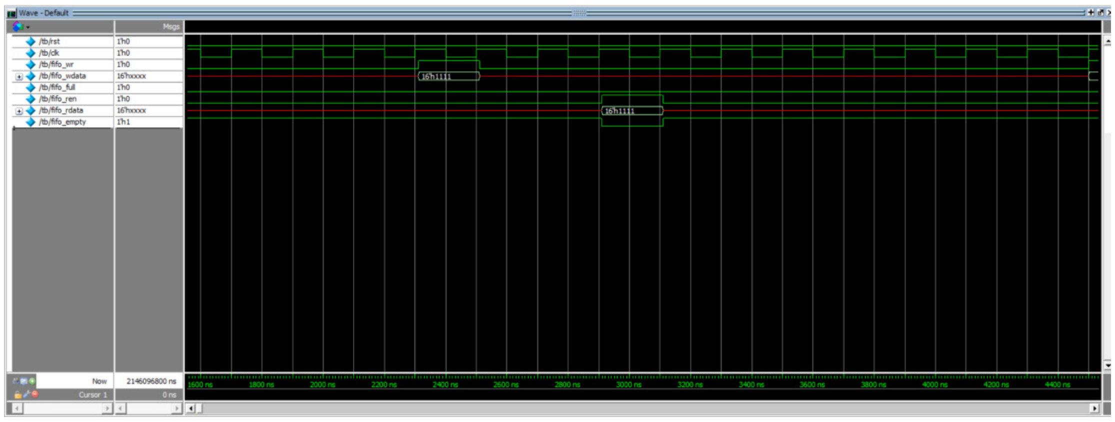

Fig.5 Pre-fetch FIFO only writes one data read/write

Fig 5 shows the simulation verification of a FIFO with prefetching function when reading only one data. In the simulation, an LFIFO module is first used to complete the basic functions of a regular asynchronous FIFO, and then the SYNC. FIFO module is used to finally form the prefetching FIFO. In the prefetching FIFO, when the write clock edge arrives, 16 bit data 1111 can be immediately written, and when the read enable signal FIFO. ren arrives, 1111 is read out.

ISSN 2959-6157

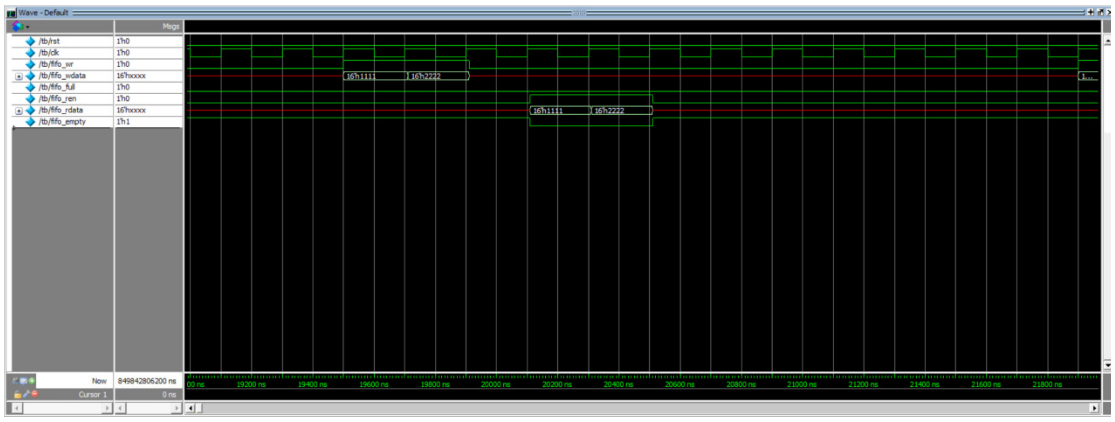

Fig.6 Pre fetch FIFO for reading and writing multiple data

Fig 6 shows the simulation of the function of continuously reading and writing multiple data in the pre fetch FIFO. When the write enable signal Fifo-wr arrives, two data 1111 and 2222 are written in the clock when the clock beats Fifo-uwdata. After one clock edge, the write enable signal Fifo-ren arrives, and the pre fetch FIFO reads out the data 1111 and 2222.

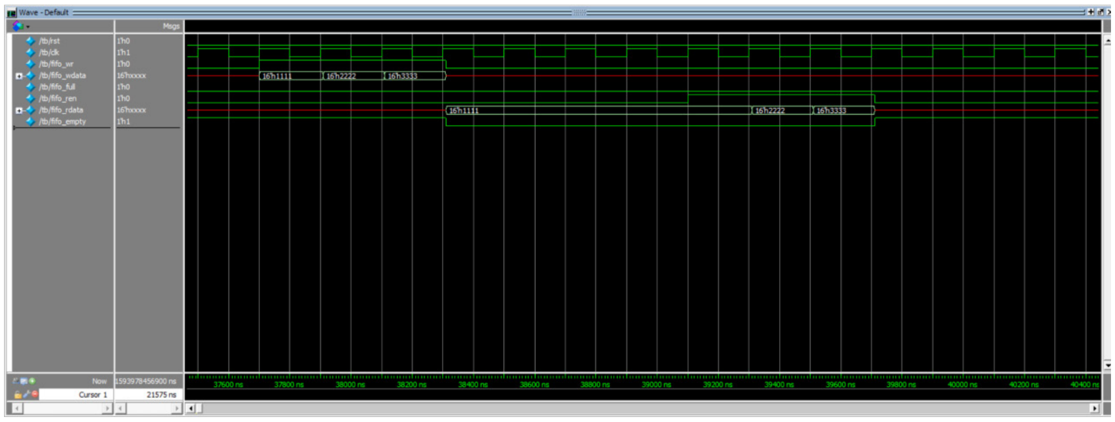

Fig.7 Read continuously after completing the pre fetch FIFO

Fig 7 shows the simulation of the continuous read function after pre fetching FIFO write. When the write enables signal Fifo-wr arrives, two data 1111222 and 3333 are written to Fifo-uwdata when the clock beats. After writing, the data is immediately read and output.

From the above simulation results, it can be seen that when the pre fetch function FIFO is working and receives a read enable signal, the data can be read out immediately on the rising edge of the enable signal, indicating that this design function can achieve the expected goal.

Comparing the simulation results of a regular FIFO and a FIFO with prefetching function, the regular FIFO generates data output at the end of the current beat and the beginning of the next beat after the read enable signal arrives, while the FIFO with a prefetching function can immediately output data when the read enable signal arrives, which can reduce the waiting time for data.

5. Discussion

The difficulty in FIFO design is how to avoid the problem of empty full indication and metastability. This article mainly proposes the concept of FIFO with a prefetching function, so it adopts the same design ideas as previous researchers to avoid this problem. Currently, the FIFO design is more complete, and some scholars have studied the use of four states: empty, full, empty, and to full to judge and avoid indication problems. In practical life, with the continuous progress of technology, various new triggers have been proposed, which have stronger resistance to metastable states and can effectively avoid metastable states.

6. Conclusion

By improving the pre fetch FIFO in the later stage, it can be applied to machine learning training of deep learning models. The pre fetch function can read family data, improve training efficiency. In game development, pre fetch FIFO can be used to load resources in advance, reducing players' waiting time during the game process and enhancing user experience. Or in network communication, packet processing can be optimized to improve network throughput and transmission efficiency.

Today, with the widespread application of integrated circuits, FIFO plays an important role as a simple module in circuits and is widely used in the communication industry, video surveillance processing, and other fields. The FIFO module with prefetching function designed in this article, although implementing prefetching function, is only a simple module design in electronic circuits and has not been customized for practical production and life. The next step of scientific research exploration can consider placing FIFOs with prefetching function in specific occasions and environments to greatly improve data processing speed for a certain scenario. In today's constantly developing and mature technology, in the future, FIFO may also incorporate intelligent design, dynamically adjusting cache strategies through machine learning and data analysis.

References

[1] Jin Dachao, Leng Jianwei. Implementation of Asynchronous Clock Domain Signal Synchronization. Journal of Tianjin University of Technology, 2017, 33 (3): 40-44

[2] Shi Huajun. Design and Implementation of Efficient Asynchronous FIFO. Changsha, Hunan University, 2013

[3] Chen Tingting, Lu Feng, Wan Shuqin, et al. A Design of Asynchronous FIFO Circuit with Controllable Delay. Microelectronics, 2022, 52 (01): 42-46.

[4] Cummingsce. Simulation and Synthesis Techniques for Asynchronous FIFO Design Asynchronous Pointer Comparisons. http:// www.sunburst-design.com/papers,2021-07.

[5] Li Sai, Jiang Lin. Design and Implementation of Asynchronous FIFO in OTN. Research on Optical Communication, 2015 (5): 55-58

[6] Liu Mengying, Fu Jianjun, Liu Yunjing. Design and Implementation of an Improved SPI Interface. Electronics and Packaging, 2019, 19 (12): 17-22

[7] Zhou Yunsong, Ran Wanning. 10 Gigabit Ethernet Card Based on FMC Standard. Electronics and Packaging, 2020, 20 (4): 39-42