Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/14/20230770

Research and analysis of FIFO related working principles

Yiting Gao

Xidian University & Heriot-Watt University, Xi'an, China & Edinburgh, UK

yg2027@hw.ac.uk

Abstract. An asynchronous FIFO is a classic circuit device used for caching data and accommodating the frequency of asynchronous signals. With the continuous improvement of technology, the advantages of this device in real-time transmission, multi-core processing, and data acquisition have become more apparent. As a result, the original structure has been optimized and extended using algorithm and design techniques to achieve greater efficiency and applicability in various academic and industrial applications. This article discusses several methods for optimizing FIFO in different aspects and examines how an unlocked FIFO queue is applied in a CAN bus data acquisition system, as well as its use in data hybrid framing technology. The findings show that FIFO performs exceptionally well as a cache structure, particularly in situations involving large amounts of data acquisition and real-time procession. The various optimization techniques discussed in this article include using pipeline registers, implementing a dual-clock FIFO, and employing Gray-code-based pointers. These techniques can increase the speed and reduce the power consumption of FIFO, thereby improving its overall efficiency. In addition, the article introduces the unlocked FIFO queue, which provides more flexibility than conventional locked FIFOs. Unlocked FIFO queues allow multiple cores to access the buffer simultaneously, which makes them ideal for high-performance systems. Finally, the article explores the application of FIFO in CAN bus data acquisition systems, where it is utilized as a buffer between the bus and other devices. Additionally, the use of FIFO in data hybrid framing technology is discussed, where it is essential for maintaining the integrity of the data stream.

Keywords: FIFO, CAN, gray codes.

1. Introduction

The asynchronous FIFO has the characteristics to write data in a buffer and read data in the same buffer with the original order. Asynchronous FIFOs pass data from one clock domain to another asynchronous clock domain. They are widely used as a typical cache to mainly improve throughput rate. Many ways to design an asynchronous FIFO are extensions of existing method which aims to satisfy application requirements. These optimizations show better performances and are suitable for particular circumstances.This paper states the typical FIFO design and discusses key points which are considered in optimization by placing these FIFO designs into different circumstances, and shows application scenarios in current technology.

2. Basic asynchronous FIFO system architecture/problems/issues

The main difficulties lie on the design of pointers and empty or full conditions. First, metastable state occurs when the read/write pointer is transmitted across the clock domain. Second, how to indicate full

DOI: 10.54254/2755-2721/14/20230770

and empty conditions correctly. To analyse the resolution, one needs to understand the working principles first.

2.1. Asynchronous FIFO pointers

There are two pointers in the FIFO points to addresses. The write pointer always point to the word to be read out currently while the read pointer points to the word which need to be read next, in other words, the word to be written next. One way to ensure the process in to make the write pointer and read pointer incremented after write data and read data separately, so that pointer will point to the next location to be written or read. Both of the two types of pointers have a common point, which is they are set to zero on reset.

Normally, the write pointer increases as soon as the first data word is written to the FIFO. Then the empty flag is cleared, and the read pointer, which is still addressing the first FIFO memory word's contents, immediately drives the first valid word onto the FIFO data output port so that the receiver logic can read it. For the read pointer is always pointing to the following FIFO word to be read, the receiver logic does not need to require two clocks. In the first clock in the receiver domain, data will be output from the FIFO, followed by a second clock if it had to increase the read pointer before receiving a FIFO data word [1]. When the read and write pointers point at the same location, there are two possible conditions: One is that the FIFO is empty, the other is that the FIFO is full. These two states both appear as write and read pointers are equal. How to tell the difference is the essential problems when designing the pointers.

2.2. Compare binary pointers and Gray pointers

In simulations, delays are bound to happening when the FIFO is set in a more realistic environment instead of an ideal one. Problems occur when binary-count FIFO pointers are used. Pointers always have multi-bits, so there is a chance that all the bits may change simultaneously. The reason is that on one clock edge, the bit sequence do not change at the same time due to the delays and a period of time is required to ensure state stability. It is a great chance to misjudge the state and make logic errors in the unstable period. As a result, Gray code counter is taken into consideration to resolve this problems [1].

A typical property of Gray code is that a number in gray code changes only one bit with each increment of one. This feature is very useful in the representation of pointers. When the Gray code address increases, only one bit is always changed, which greatly reduces the frequency of signal hopping compared with the binary representation of pointer, thus greatly reducing the probability of metastable condition.

2.3. The existence and solution of metastable problem as a type of optimization method

Using Gray code instead of binary code to represent pointers is actually a method to optimize FIFO design to solve problems. Seen from the codes above, this type of optimization actually aims to deal with the basic problems lie on the source code.

A universal resolution is to use binary bit strings in local clock domain. When synchronizing data in two different clock domains, change the natural binary into Gray code to avoid metastable condition during 2 clock periods. After synchronization is finished, all the data are back to binary bits for following process.

The relative algorithm is simple to implement in Verilog.

```
// Write poninter: binary code to Gray code
always @(posedge wrclk or posedge wr_rst_n) begin
    if (wr_rst_n) begin                wr_gray <= 'b0;
    end
    else begin
        wr_gray <= { wr_bin[PTR], wr_bin[PTR:1] ^ wr_bin[PTR-1:0] };  end
end
// Synchronize the read pointer(in Gray) into the write clock domain
```

DOI: 10.54254/2755-2721/14/20230770

```
always @(posedge wrclk or posedge wr_rst_n) begin
    if(wr_rst_n) begin                rd_gray_ff1 <= 'b0;
                                              rd_gray_ff2 <= 'b0;
                  end
    else begin                        rd_gray_ff1 <= rd_gray;
                                              rd_gray_ff2 <= rd_gray_ff1;
    end
end
// Read pointer after synchronization from Gray to binary
always @(*) begin
rd_bin_wr[PTR] = rd_gray_ff2[PTR];
    for ( i=PTR-1; i>=0; i=i-1 )
        rd_bin_wr[i] = rd_bin_wr[i+1] ^ rd_gray_ff2[i];
end
```

Following parts of this paper share the same idea, which has the goal of obtaining better performance.

3. CAN bus data acquisition system based on lockless FIFO queue

In a recent research conducted by a Chinese team from Wuhan University of Science and Technology, unlocked FIFO has been used in the process of data acquisition function development, as the main structure in the control block of CAN [2].

CAN(Controller Area Network) has been used as main communication network between vehicle controllers because of its outstanding reliability, real-time and flexibility. In the application scenario with a large amount of CAN bus data, FIFO, as a suitable device, resolve the problem of complex acquisition process and frame losses, improving the performances especially in the software part [3].

3.1. System working principle

In this application scenario, FIFO works in control block as buffers to maintain safety, integrity and orders of real-time data acquisition. When the acquisition system listens that there is CAN message data on the bus, the system will be automatically interrupted, and the data collected real-time are temporarily stored in the unlocked FIFO queue, which can automatically adjust the queue size based on the amount of data in the guaranteed bus. The system resource usage is changed correspondingly while the data store is correct [2].

3.2. Unlocked FIFO queue

One typical condition is that the data arrival rate changes greatly in real-time data transmission. Existing lock-free queue algorithms all assume the arrival of data rate is stable. If the input data arrival rate fluctuates relatively fast, the queue will frequently tend to empty or full, resulting in a sharp decrease of efficiency for the unlocked queue.

As a result, unlock FIFO is designed as an intern process communication mechanism, combining FIFO structure with unlock queue algorithm [4]. When the data arrival rate is low, the queue adaptively shortens and thus less memory is occupied, using the processor cache more efficiently. In the condition data amount is large, in other words, the data arrival rate is higher, the queue is adaptively increased in order to prevent data loss.

Moreover, the unlocked FIFO queue only allows deletion operations at the front end of the queue and insertion operations at the back end of the queue. Because of this feature, the first element entering the queue are the first to be removed from the queue, known as first-in-first-out, FIFO [5]. Since only one end of the queue can be inserted, each element in the queue is arranged in order of precedence.

To improve the performance of the software queue through the data structure without clock is the essential ideas when designing the FIFO queue algorithm, which is also a resolution for large data flows in CAN bus. As a special condition, where there is only one producer and one consumer, it does not need to take ABA problem into consideration and avoid this problem essentially [4]. It is also a

DOI: 10.54254/2755-2721/14/20230770

preferable resolution to achieve high-speed inter process communication [6-9]. The CAN bus data acquisition system designed in this paper belongs to this condition.

According to these characteristics, the lockless FIFO queue is often used in the field of traffic peak clipping and log processing field.

3.3. Applications in CAN bus acquisition

Combined with the characteristics of FIFO queue and the characteristics of large amount of data in application scenarios, FIFO queue is integrated into the data acquisition and storage functions in the design and development. For there are numerous nodes are attached to the CAN bus and thus cause a large amount of data flow, two 3-level FIFO are used as a buffer structure to prevent lost in acquisition and interference by unrelated signals.

The main advantage is that FIFO can adapt the size of buffers according to the amount of data transmitted by the CAN bus. When the amount of data is small, the queue size is adaptively reduced to reduce system resource usage and release the system space, while the queue size increase when data surge, to avoid data loss. As shown in Figure 1.

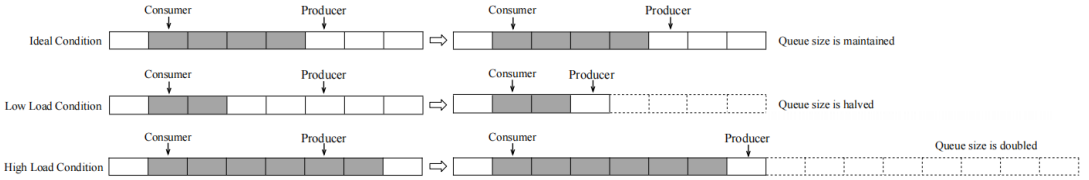

Figure 1. The algorithm principle of unlocked FIFO queue.

4. Data hybrid framing technology based on FIFO cache

In the previous design, full & empty flags are generated based on the comparison of write and read pointers. But in this design, a given value is set as a limit of pointers, showing how many of the amount of data in FIFO in the corresponding clock. Multiple FIFOs are used as different levels in the structure. Problems occur when many empty flags are generated at the same time, which leads to disorder of the program.

The data acquisition and recording device designed in this part can simultaneously acquire five types of data among which 3 of them belong to PCM and the other belong to LVDS. All the data are packaged and forwarded to the composite memories then [10]. Data hybrid framing technique built on FIFO cache serves as a solution for successfully covering various data kinds over a single path while maintaining security and rapid acquisition speed.

4.1. The choice of data cache

There are usually two ways to cache data across clock domains in FPGA, one is through dual-port RAM, the other is through FIFO. Dual-port RAM is mostly used in the design that needs to communicate with each other. After the two sides determine the address, they must control the timing of RAM read and write. FIFO itself has the feature of first in first out, which is conducive to one-way data transmission. Because this part of the data transmission is one-way, choose the FIFO memory as the data cache medium, relying on the FIFO for data transmission across the clock domain, so that in this design do not have to consider how to deal with the synchronization problem [10]. More energy and time can be spent on data flow and FIFO interface control.

In addition, FIFO does not need to consume a large number of clock cycles in the synchronization design like handshake signal or synchronization logic processing mechanism, so it can effectively improve the throughput rate of data of both sides in the communication process [11].

DOI: 10.54254/2755-2721/14/20230770

4.2. Process of hybrid framing with FIFO

The aim is to eventually creates the FIFO-based cache structure and converges various data kinds into one channel for fast and dependable transmission. After each data acquisition module collects the data, all of them are sent to its corresponding cache FIFO, then the hybrid framing module will root loop check prog_empty flags in every FIFO according to the priority.

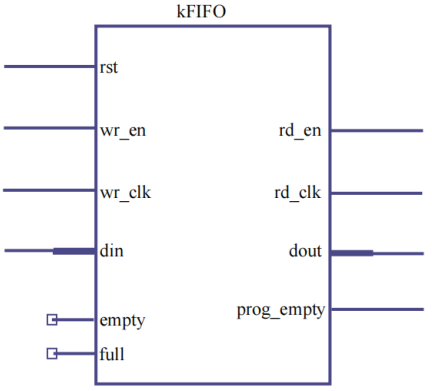

Figure 2. FIFO structure.

The asynchronous FIFO interfaces used in this design are shown in Fig.2. prog_empty is a programmable half-empty flag which show the data amount in FIFO buffer to avoid read out invalid data. A limit of data amount has been set. When the data amount is less than or equal to the set value, the flag is set to 1. When prog_empty equals 0, the system believes that there are enough data to be read out.

4.3. Data hybrid framing process with priorities

To prevent the condition that multiple FIFOs from displaying prog_empty flag 0 at the same time leads to program disorder, the priority of data framing is designed. The data with a higher bit rate is set to a higher priority, for it can increase the reliability of both acquisition and transmission, as well as reducing the usage of FPGA internal resources. As shown in Figure 3.

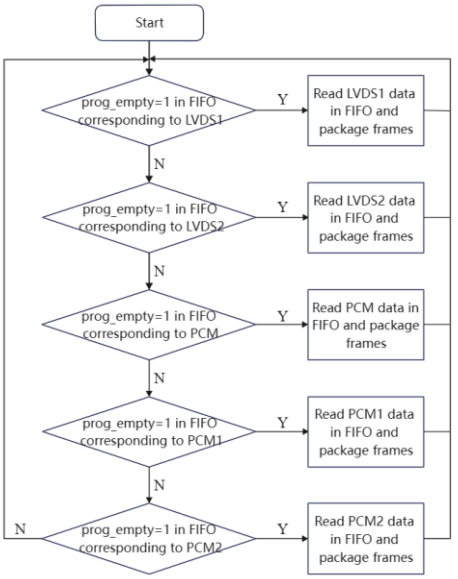

Figure 3. Data hybrid framing process with priorities.

DOI: 10.54254/2755-2721/14/20230770

5. Universal applications in wide fields

In addition to all the statements above, there are still many types of academic filed that FIFO can play a role in. In AI (Artificial Intelligence), FIFO is introduced to memory the calculation output data of convolution and implement convolutional neural network parallel design, which improve the throughput to a certain extent [12]. The property of first-in first-out is suitable to achieve multi access signal check and the results are always exact and without frame loss. Combined with it, an accurate and lossless control strategy is proposed [13-14], which meets the requirements of high detection rate and high precision for color separators. FIFOs significantly improve the detection accuracy and production of color separators [15].

This paper cannot list every application, but the breadth of the application fields can express the values of continuously improving FIFO concerned researches.

6. Conclusion

The design of write and read pointers and details concerned is important and have influences on whether the full and empty flags are correctly generated. Gray codes as the previous and the most renowned algorithm optimization for FIFO pointer design has been widely acknowledged. CAN bus acquisition based on unlocked FIFO and data hybrid framing with FIFO cache are two of the successful examples of current optimization produced due to the requirements of application in different fields. In the aspects of saving space and improving efficiency according to real-time amount of data flow as well as avoiding losing frames and packets in acquisition system, optimized FIFOs show great performances to satisfy the needs, which also reduce the cost in real application.

In the current information era, numerous amounts of data acquisition and procession are bound to be the developing tendency. It turns out that FIFO still has values in advanced technologies and there is great potential in real-time data procession, reliable transmission and mixed transmission for variety of types of data. The following research will focus on the further detailed algorithms and developing prospects.

References

[1] Attar H, Khosravi M R, Igorovich S S, et al. Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks[J]. International Journal of Distributed Sensor Networks, 2020, 16(6): 1550147720913233.

[2] Wieder A, Brandenburg B B. On spin locks in AUTOSAR: Blocking analysis of FIFO, unordered, and priority-ordered spin locks[C]//2013 IEEE 34th Real-Time Systems Symposium. IEEE, 2013: 45-56.

[3] Bouajjani A, Habermehl P. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations[J]. Theoretical Computer Science, 1999, 221(1-2): 211-250.

[4] Strano A, Ludovici D, Bertozzi D. A library of dual-clock FIFOs for cost-effective and flexible MPSoC design[C]//2010 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. IEEE, 2010: 20-27.

[5] Martin J. FIFO rosters and workers' health and safety: a case study of the impacts of extended shift rosters on electrical workers in construction in the resources sector[J]. Labour & Industry: a journal of the social and economic relations of work, 2020, 30(4): 378-400.

[6] Yoon D H, Erez M. Flexible cache error protection using an ECC FIFO[C]//Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. 2009: 1-12.

[7] Bouajjani A, Habermehl P. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations[C]//ICALP. 1997, 1256: 560-570.

[8] Immich P K, Bhagavatula R S, Pendse R. Performance analysis of five interprocess communication mechanisms across UNIX operating systems[J]. Journal of Systems and Software, 2003, 68(1): 27-43.

[9] Khan M A, Ansari A Q. n-Bit multiple read and write FIFO memory model for network-on-chip[C]//2011 World Congress on Information and Communication Technologies. IEEE, 2011:

DOI: 10.54254/2755-2721/14/20230770

1322-1327.

[10] Krom P, Ahmad R, Mustafa S A, et al. Implementation of kanban-based FIFO system to minimize lead time at automated optical inspection operation-a case study in semiconductor industry[C]//Advances in Mechatronics, Manufacturing, and Mechanical Engineering: Selected articles from MUCET 2019. Springer Singapore, 2021: 97-108.\

[11] Shurman M M, Al-Rashdan R M, Al-Bataineh M K. Comparison study of FIFO and MDRR queuing mechanisms On 5G cellular network[C]//2018 9th International Conference on Information and Communication Systems (ICICS). IEEE, 2018: 61-65.

[12] Kesselman A, Kogan K, Segal M. Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing[C]//Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing. 2008: 335-344.

[13] Dittman C K, Henriquez A, Roxburgh N. When a non-resident worker is a non-resident parent: investigating the family impact of Fly-In, Fly-Out work practices in Australia[J]. Journal of Child and Family Studies, 2016, 25: 2778-2796.

[14] Kogan K, López-Ortiz A, Nikolenko S I, et al. A taxonomy of semi-FIFO policies[C]//2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC). IEEE, 2012: 295-304.

[15] Bahaweres R B, Fauzi A, Alaydrus M. Comparative analysis of LLQ traffic scheduler to FIFO and CBWFQ on IP phone-based applications (VoIP) using Opnet (Riverbed)[C]//2015 1st International Conference on Wireless and Telematics (ICWT). IEEE, 2015: 1-5.