Asynchronous FIFO Design Based on Verilog

Ying Xu*

School of Mechanical Electronic & Information Engineering, China University of Mining and Technology-Beijing, Beijing, China

* Corresponding Author Email: 2010490303@student.cumtb.edu.cn

Abstract. With the rapid development of integrated circuits, asynchronous First Input First Output (FIFO) is often used to solve the problem of data transmission across the clock domain. This paper mainly studies the key problem of asynchronous FIFO design - the generation of empty - full signal. To solve this problem, it is necessary to realize the synchronization of signal across the clock domain and convert binary code into gray code to reduce the probability of metastable state. The null and full signals generated by the asynchronous FIFO designed in this paper are false null and false full, but this does not affect the function of the asynchronous FIFO, and will only lose part of the performance. Through Modelsim simulation verification, the designed asynchronous FIFO can realize first-in, first-out of data and correctly generate empty and full signal, which meets the design requirements. The research of this paper is helpful for further application of asynchronous FIFO in data transmission across clock domains.

Keywords: Asynchronous FIFO; Metastable state; Empty - full signal; Cross clock domain.

1. Introduction

With the rapid development of integrated circuits, modern digital systems often adopt multi-clock domain design in order to improve the performance. Signals that cross the clock domain can generate metastases when they are transmitted. Asynchronous FIFO can effectively solve the metastable phenomenon of data transmission and storage across the clock domain [1].

To solve the above problems, this paper presents the research problem of how to generate empty full signal with asynchronous FIFO on the premise of reducing the probability of metastable state. This paper first introduces the four basic modules of asynchronous FIFO and the key problem of asynchronous FIFO design -- how to generate empty and full signal, and then carries on the Verilog design of the asynchronous FIFO, and finally carries on the simulation and result analysis of the designed asynchronous FIFO, and further analyzes the significance of empty and full signal.

The research in this paper is helpful to the further development of asynchronous FIFO and makes a contribution to the application of asynchronous FIFO in signal transmission across the clock domain.

2. Basic knowledge of asynchronous FIFO

2.1. Asynchronous FIFO basic module

First In First Out, or FIFO, refers to a first-in-first-out data cache. It differs from conventional memory in that there is no external read-write address line, making it very simple to use, but it has the drawback of only supporting sequential write and sequential read operations, with the data address being automatically added by the internal read-write pointer to complete rather than being determined by the address line to read or write, as in conventional memory. The location is set.

Compared with synchronous FIFO read and write in the same clock signal, asynchronous FIFO read data and write data in different clock signals, so asynchronous FIFO is mainly applied to realize data transmission between different clock domains or as a data interface of different data widths. Asynchronous FIFO includes four modules: dual port RAM, control module, control module, the clock synchronization module [2].

Dual port RAM: DPRAM is short for dual port RAM. Dual port RAM has two ports, allowing the read controller and the write controller to access the storage unit asynchronously at the same time. In

the asynchronous FIFO, data can be written and read simultaneously, and real-time cache of written data can be realized. Dual port RAM can be read and written at any time and is very fast [3].

Write control module (full check module): The function of the write control module is to control the write data and determine whether the DPRAM is full. The write address pointer is generated by the simultaneous action of the write clock and the write enable signal, and data is sequentially written to the dual port RAM. After the clock synchronization module delays two beats, the read address pointer is compared with the write address pointer, which generates a full write signal in the write control module (write clock domain).

Read control module (null detection module): The function of the read control module is to control the read data and judge whether the DPRAM is null read. The read address pointer is created by the simultaneous operation of the read clock and the read enable signal, and it reads data sequentially from the DPRAM. After the clock synchronization module delays two beats, the write address pointer is compared with the read address pointer, and then the read control module (read clock domain) generates a null read signal.

Clock synchronization module: Compares the read address pointer with the write address pointer in the write clock domain after a delay of two beats, and generates a full write signal. The write full signal is fed back to the write control module. The write enable function is lowered to stop data writing. The write address pointer is delayed by two beats. Comparing the read clock domain with the read address pointer generates a null read signal. The null read signal is fed back to the read control module, and the read enable is lowered to stop reading data [4].

The basic modules of an asynchronous FIFO are shown in the figure1 [5].

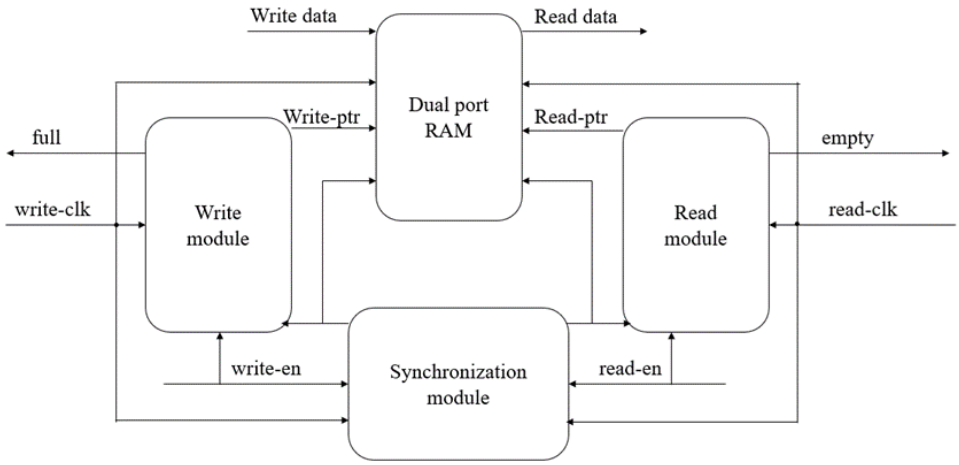

Fig. 1 Asynchronous FIFO basic module

2.2. The key of asynchronous FIFO design - the generation of empty and full signal

In a synchronous FIFO, since the write and read data occur in the same clock domain, a counter method can be used to generate empty and full signals [6]. Write a data count+1, when the count reaches the depth defined by the two-port RAM, the write up signal is generated; Read a data count-1, when the count is reduced to 0, generate a null read signal. Synchronous FIFO can also use high extension method [7].

Asynchronous FIFO reads and writes to different clocks, compared to synchronous FIFO reads and writes to the same clock. Therefore, the asynchronous FIFO cannot use the counter method to generate empty and full signal, and must use the high extension method: define the read and write address pointer more than one bit more than the two-port RAM. If the highest bit is different and the other bits are the same, it is full. The highest bit is the same, and the other bits are the same. Because read address pointer and write address pointer operate on distinct clocks, it is important to continue with synchronous processing rather than directly comparing the two. In the write clock domain,

contrast the read address pointer and the write address pointer. A comparison between the write address pointer and the read address pointer in the read clock domain [8].

When the read and write address pointer moves, if binary is used, multi-bit hopping may occur, and metastable state may occur. Metastability is inevitable, but the probability of metastability can be reduced by using gray code [9]. The lowest bit of a gray code is the XOR of the lowest bit and the highest bit of a binary code, while the highest bit of a gray code is the highest bit of a binary code. Only one bit of gray code changes, which greatly reduces the probability of metastable state. When the gray code of the read address pointer is the same as that of the write address pointer after a delay of two beats, the null read signal is generated. When the gray code of the write address pointer is different from the high and secondary high of the gray code of the read address pointer after the delay of two beats, the full signal is generated when the other bits are the same [10].

3. The asynchronous FIFO Verilog implementation

3.1. basic module of the asynchronous FIFO Verilog implementation

Dual port RAM asynchronous FIFO storage module design code is as follows: reg [7:0] DPRAM [7:0]; Dual port RAM to implement in a two-dimensional array, the definition of dual port RAM is 8-bit depth and width. The width of the DPRAM is the width of the write and read data, here is 8 bits. Write control module design code is as follows:

```
always @ (negedge wr_rst_n or posedge wr_clk) begin
  if (! wr_rst_n)
       wr_ptr <= 0;
  else if (! full && wr_en) begin
       wr_ptr <= wr_ptr + 1'd1;
       DPRAM [wr_ptr_true] <= data_in;
  end
end
```

When wr_rst_n is 0, asynchronous zeroclearing is performed. At the uptick of the write clock, the write address is updated and data is written to the dual-port RAM when the write enable is active and not in a full state.

Read the control module design code is as follows:

```
always @ (posedge rd_clk or negedge rd_rst_n) begin
  if (! rd_rst_n)
       rd_ptr <= 'd0;
  else if (rd_en &&! empty) begin
       data_out <= DPRAM [rd_ptr_true];
       rd_ptr <= rd_ptr + 1'd1;
  end
end
```

When rd_rst_n is 0, asynchronous zeroclearing is performed. The read clock rises, updates the read address, and reads data from the two-port RAM when the read enable is active and non-empty.

The design of the clock synchronization module code is as follows:

The read address pointer to the write clock domain:

```
always @ (posedge wr_clk or negedge wr_rst_n) begin
  if (! wr_rst_n) begin
       rd_ptr_g_d1 <= 0;
       rd_ptr_g_d2 <= 0;
  end
  else begin
       rd_ptr_g_d1 <= rd_ptr_g;
       rd_ptr_g_d2 <= rd_ptr_g_d1;
```

```
  end
end
```

When wr_rst_n is 0, asynchronous zeroclearing is performed. At the risetime of the write clock, the read address pointer is synchronized to the write clock domain, that is, the gray code of the read address pointer is delayed by two beats. The result of one-beat delay is rd_ptr_g_d1, and the result of two-beat delay is rd_ptr_g_d2. Will write address pointer to read clock domain:

```
always @ (posedge rd_clk or negedge rd_rst_n) begin
  if (! rd_rst_n) begin
        wr_ptr_g_d1 <= 0;
        wr_ptr_g_d2 <= 0;
  end
  else begin
        wr_ptr_g_d1 <= wr_ptr_g;
        wr_ptr_g_d2 <= wr_ptr_g_d1;
  end
end
```

When rd_rst_n is 0, asynchronous zeroclearing is performed. The rising edge of the read clock synchronizes the write address pointer to the read clock domain, that is, the gray code delay of the write address pointer is two beats. The result of a one-beat delay is wr_ptr_g_d1, and the result of a two-beat delay is wr_ptr_g_d2.

3.2. Verilog design generated by empty and full signal

To reduce the probability of metastable state, the following code transforms binary code into gray code:

```
assign      wr_ptr_g = wr_ptr ^ (wr_ptr >> 1);
assign      rd_ptr_g = rd_ptr ^ (rd_ptr >> 1);
```

In a gray code, the highest bit is identical to the highest bit in a binary code, while the lowest bit is created by XORing the lowest bit and highest bit in a binary code.

Reading the judge read clock domain empty signal, the code is as follows:

```
assign
full = (wr_ptr_g == {~ (rd_ptr_g_d2[3: 2]), rd_ptr_g_d2[1: 0]})? 1'b1: 1'b0;
```

rd_ptr_g_d2 indicates the result of synchronizing the read address pointer to the write clock domain after a two-beat delay. At this time, compare rd_ptr_g_d2 with wr_ptr_g. If the high and low levels of the gray code are different and the other bits are the same, the write up signal is generated.

The write clock domain and the read pointer are in sync. It is a false full write if the original read pointer exceeds the synchronized read pointer for one circle but does not exceed the real read pointer and the original read pointer is greater than or equal to the synchronized read pointer.

When the write pointer is synchronized to the read clock domain, the original write pointer is greater than or equal to the synchronized write pointer, and the read pointer exceeds the synchronized write pointer for one circle but does not exceed the real write pointer. This is a false read null.

When not read empty times empty, not write full times full, will not have an impact on the function, will only cause a certain loss of performance (reduce capacity), to the design margin, has a certain protection.

4. Simulation and result analysis

For asynchronous FIFO design good functional test:

To write data in asynchronous FIFO, write random number here. The simulation results are shown in Figure 2.

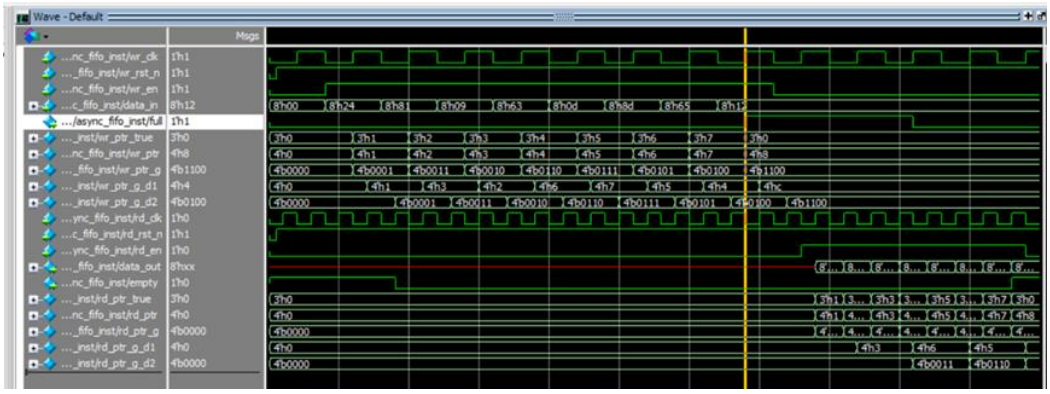

Fig. 2 Write data in asynchronous FIFO

The depth of dual-port RAM defined is 8bit. After eight data are written, the maximum storage space of dual-port RAM has been reached, wr ptr g=1100,rd ptr g d2=0000. The high and low levels of the two are the same, but the other bits are different. It can be seen that the designed asynchronous FIFO gives a full signal when the write is full, and the write enable can reduce the stop of writing data, which meets the design requirements.

Read the data written in the previous step from the asynchronous FIFO. The simulation results are shown in Figure 3.

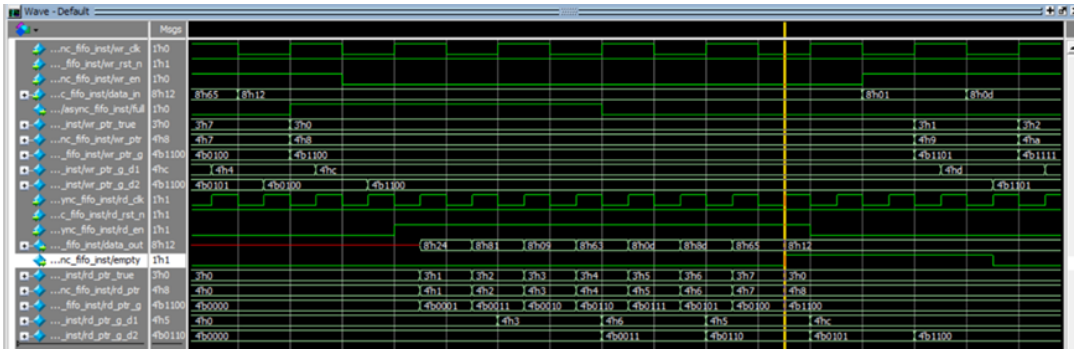

Fig. 3 Read data from asynchronous FIFO

As can be seen from the simulation results, the data written first is read out first, which conforms to the design concept of asynchronous FIFO. After eight data is read out, there is no data in the dual-port RAM, wr ptr g d2=1100, rd ptr g=1100, all bits of both are the same, generate null read signal, stop reading data. The designed asynchronous FIFO gives null signal in space reading time, and the read enable can reduce the stop reading data, which meets the design requirements.

Write data and read data simultaneously in asynchronous FIFO, with fast read and slow write. The simulation results are shown in Figure 4.

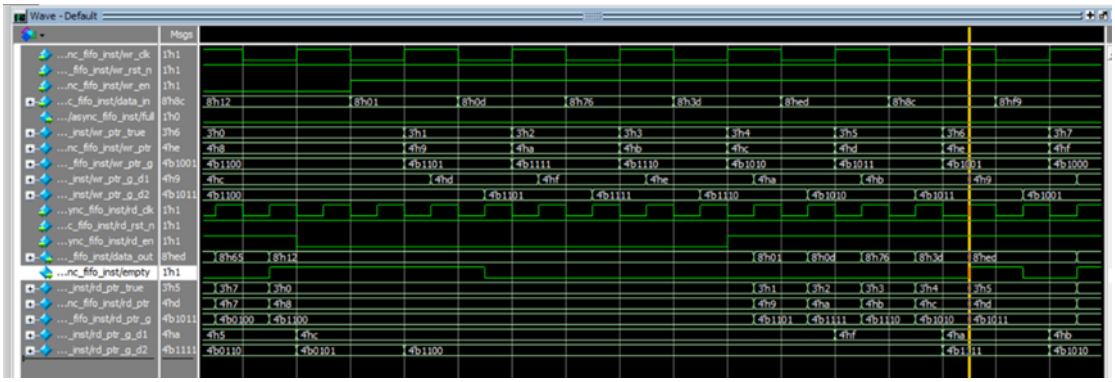

Fig. 4 Write data and read data simultaneously in asynchronous FIFO

As can be seen from the figure, data is first written into the dual-port RAM, and then read and write simultaneously, with fast read and slow write. Reading fast and writing slow will produce empty

signals. As shown in the figure, when there is a data 8 'h8c left in the asynchronous FIFO, that is, when there is no null read, the asynchronous FIFO generates null signals. The empty signal here is "false empty signal".

In unread empty empty times, will not affect the function of the asynchronous FIFO, will only cause some performance loss, the loss of a small part of the storage space. But this loss in exchange for the safety margin, otherwise when the write is full still write data, read blank still read data, will cause the asynchronous FIFO function error.

5. Conclusion

Through the above research, this paper draws the following conclusions:

In this paper, the basic module of asynchronous FIFO is designed and implemented by Verilog, and the null and full signal is generated under the condition of reducing the probability of metastable state. It can be seen from the simulation diagram that the designed asynchronous FIFO meets the requirement that the data written first is read out first. When the data in the asynchronous FIFO reaches or approaches the maximum depth of double-port RAM, full signal is generated; when there is no data in the asynchronous FIFO or near the empty signal, the expected effect is achieved. A further explanation is given for the empty - full signal: the empty - full signal generated by asynchronous FIFO is sometimes not the true vacuum full. This design uses a small part of performance loss to exchange for the safety margin, which is in line with the design goal.

References

[1] Wang GC, Ma YP, Lu HT, Chen XQ, Li P. Clock Domains Cross FIFO Interface Design of Multichannel Continuous DDR2 Read and Write. Applied Mechanics and Materials. 2014;556-562:1622-1626.

[2] Yadlapati A, Kishore Kakarla H. Design and Verification of Asynchronous FIFO with Novel Architecture Using Verilog HDL. Journal of Engineering and Applied Sciences. 2019;14(1):159-163.

[3] Van De Goor AJ, Zorian Y. Effective march algorithms for testing single-order addressed memories. Journal of Electronic Testing. 1994;5(4):337-345.

[4] Kumar A, Shankar S, Sharma N. Verification of Asynchronous FIFO using System Verilog. International Journal of Computer Applications. 2014;86(11):16-20.

[5] Liu BQ, Liu MZ, Yang G, Mao XB, Li HL. Research and Design of Asynchronous FIFO Based on FPGA. Applied Mechanics and Materials. 2014;644-650:3440-3444.

[6] Konstantinou D, Psarras A, Nicopoulos C, Dimitrakopoulos G. The Mesochronous Dual-Clock FIFO Buffer. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems. 2020;28(1):302-306.

[7] Akhare* M, Narkhede N. Design and Verification of Generic FIFO using Layered Test bench and Assertion Technique. International Journal of Engineering and Advanced Technology. 2019;8(6):5254-5260.

[8] Kim HK, Wang LT, Wu YL, Jone WB. Testing of Synchronizers in Asynchronous FIFO. Journal of Electronic Testing. 2013;29(1):49-72.

[9] Cui ZJ, Hu LL. Design of SDH Positive/Zero/Negative Justification Circuits Based on FPGA. Advanced Materials Research. 2013; 748:874-878.

[10] Abdel - hafeez S, Gordon - Ross A. Reconfigurable FIFO memory circuit for synchronous and asynchronous communication. International Journal of Circuit Theory and Applications. 2021;49(4):938-952.