

Optimization of Asynchronous FIFO Design Difficulties Using Verilog HDL

Zichao Zhang*

Department of Electrical Engineering, Jilin University, Jilin, China

* Corresponding Author Email: zhangzc6519@mails.jlu.edu.cn

Abstract. FIFO is a first-in first-out data storage and buffer, which can cache the continuous data stream to prevent data loss during incoming and storing operations. A special interface technology, asynchronous FIFO, needs to be introduced to ensure the correct data transmission when data needs to be transmitted between different clock domains or when the input device and the output device do not match the data interface width. This paper uses Verilog HDL language to design an asynchronous FIFO which can realize the conversion of different data bit widths. In the paper, the difficulties of this design are analyzed in detail, including the method of synchronizing different clock domains, the judgment of FIFO's empty and full state, the method of reducing metastability during data transmission and the method of data bit width conversion. Finally, the designed program is simulated and verified by modelsim software, and the waveform results obtained by simulation are analyzed in detail.

Keywords: Asynchronous FIFO; data bit widths; metastability; synchronize.

1. Introduction

FIFO is a first-in-first-out data storage and buffer. It can cache the continuous data stream, and prevent the data from being lost during incoming and storing operations. When there are multiple clock signals in the design, and data needs to be transmitted among these clock domains, the register will lose data due to the frequency mismatch of the clock signals. At this time, asynchronous FIFO should be used for caching to ensure the correct data transmission. Or when the input device and the output device transmit data with mismatched data interface widths, to realize data transmission between them, asynchronous FIFO can also be used as data interfaces with different data widths. Aiming at the design of asynchronous FIFO, Clifford E. Cummings introduced the design idea of asynchronous FIFO with the same data width in detail in his article and put forward his own unique method to judge the empty and full state of FIFO [1]. Lifan Huang and Xueren Zheng analyzed the generation and elimination of metastability in FPGA design and emphasized the use of "handshake protocol" to weaken the metastability in asynchronous FIFO design [2]. Under the condition of increasing data transmission, Yin Qiao Wang, Xiao Guang Hu, and Li Fu have designed an asynchronous FIFO that supports a multifunctional serial port and is high-speed and has large capacity by studying the working mode of SDRAM and multifunctional serial port [3]. This paper focuses on data transmission between devices with mismatched data bit widths when the clock frequencies of input and output devices are different. It also includes the research and application of clock domain synchronization and the elimination of metastability in asynchronous FIFO, which makes up for the lack of previous research on data bit width conversion in asynchronous FIFO. This research can be used to transmit the data collected by AD to PC, or to transmit the data output by MCU with data-bit width of 8 to DSP with data-bit width of 16, which has certain industrial practical value.

2. Framework of System Design

2.1. Framework of System Design

The system framework adopted in this design is shown in the figure 1 [4].

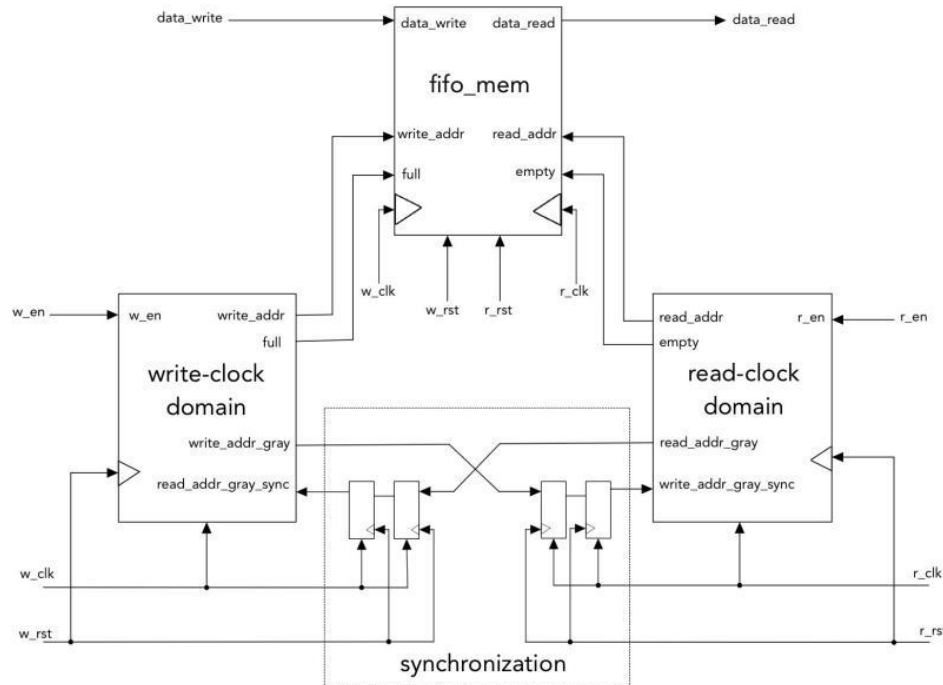


Fig. 1 System framework

The framework can be divided into four parts:

1)The "fifo_mem" module, which is equivalent to the designed data register (fifo_mem) of FIFO, can realize that the data written first in FIFO can be read out first, and can perform the conversion of data with different bit widths;

2)The clock synchronization module, which is used to synchronize two different clock domains and weaken the metastability by making use of two cascaded D flip-flops;

3)The read-clock domain module, whose function is to generate the read-address pointer(read_addr), the read-empty signal(empty) of FIFO and the Gray code(read_addr_gray) of the extended pointer of the read-address pointer;

4)The write-clock domain module, whose function is to generate the write-address pointer(write_addr), the write-full signal (full) of FIFO and the Gray code (write_addr_gray) of the extended pointer of the write-address pointer.

2.2. Difficulties in asynchronous FIFO Design

There are three key technical difficulties in asynchronous FIFO design with different bit widths:1)Influence of Metastability; 2)Generation of the empty flag and the full flag; 3)Conversion of data bit width. The following is the analysis of these three difficulties.

2.2.1 Influence of metastability

Metastability is a verifiable state that the Flip-Flop cannot reach within a specified period. Metastability in asynchronous FIFO is caused by the different frequencies of the two input clock signals. When a flip-flop enters Metastability, the output terminal Q of the flip-flop is in an uncertain state for a long time after the arrival of the effective clock edge, and during this time, the value of the output terminal D is in an oscillating state between 0 and 1, instead of being equal to the value of the data input terminal D [5]. The generation of metastability leads to an invalid reset of the reset signal, but counters and state machines are usually used in FPGA design, including the application of asynchronous reset design. However, due to the influence of metastability, using asynchronous reset

can easily lead to the failure of the counter to reset normally and the state machine hanging in a certain state, which is fatal to the system.

2.2.2 Generation of the empty flag and the full flag

The write-full flag is a signal transmitted by the status circuit of FIFO when the FIFO is full or about to be full, to restrain the write operation of FIFO from continuing to write data into FIFO, resulting in overflow [6]. The read-empty flag is a signal transmitted by the status circuit of the FIFO when the FIFO is empty or about to be empty, to prevent the read operation of the FIFO from continuing to read data from the FIFO, resulting in the underflow. By setting a read pointer and a write pointer, and then comparing the two pointers, the empty full flag is generated. As for asynchronous FIFO, because these two pointers are in different clock domains, the read pointer and the write pointer need to be synchronized before comparison. Moreover, because synchronization takes time, and it stands a good chance that there will be data written into FIFO during the synchronization time, the synchronized pointer would be less than or equal to the current actual pointer, which needs attention in the design.

2.2.3 Generation of the empty flag and the full flag

The conversion of data bit width refers to converting the input data bit width into the data bit width of the output of FIFO. When two devices with mismatched data bit widths are connected for data transmission, the output data may be lost or the transmission data may be disordered. In this case, FIFO can be used for data matching for data with different widths. For example, the MCU (Microcontroller Unit) outputs 8-bit data, while the DSP (Digital Signal Processing) may input 16-bit data. When data transmission is needed between them, FIFO can be used to achieve the purpose of data matching. In this research, the FIFO for conversion between data with different bit widths is designed and simulated and one of the difficulties in the design is the increment of the read pointer.

3. Overall Design Method

3.1. The "fifo_mem" Module

The functions of "fifo_mem" module are subdivided into the following aspects.

3.1.1 Clearing the Contents of FIFO

Set a counter "i" whose value is the same as the depth of FIFO. When "w_rst" (write-clock reset signal) is valid, the counter is used to clear the contents of FIFO. In this case, "i" is used as the address of the FIFO storage space, and every time a piece of data in a FIFO storage space is cleared, "i" is incremented by one to address the next data in the FIFO. The purpose of clearing the storage is to eliminate the influence of possible data in the original FIFO storage space(fifo_mem) on the output data.

3.1.2 Storing externally written data into FIFO

When "w_rst" and full flag signal "full" are zero and "w_en" is equal to 1 at the same time, FIFO writes external data into "fifo_mem", and it is stipulated that the data width of "fifo_mem"(fifo_depth) must be consistent with the external input data width(datain-width).

3.1.3 Reading out the Data in "fifo_mem"

The process of reading data data_read from data "fifo_mem" includes the judgment of input data and output bit width, and the two cases of input data bit width being larger than output data and input data bit width being smaller than output data bit width corresponding to different data output methods. The process of reading data from "fifo_mem" to data_read includes judging the bit width of input data and output data, and the two cases of input data (datain-width) bit width being larger than output data (dataout-width) and input data bit width being smaller than output data bit width correspond to different data output methods. The specific Verilog code of this part is as follows:

```

always @(posedge r_clk or posedge r_rst) begin
    if(r_rst)
        cnt <= 0;
    else if (r_en && (!empty))
        if((mul==10'd2) && (!div))
            data_read <= {fifo_mem[read_addr+1],fifo_mem[read_addr]};
        else if((div==10'd2) && (!mul)) begin
            if(cnt==0)
                data_read<=fifo_mem[read_addr][(datain_width-1):(datain_width/2)];
            else
                data_read <= fifo_mem[read_addr][(datain_width/2-1):0];
            cnt <= (cnt==div-1) ? 0 : cnt+1;
        end
    else;
end

```

In the above code, "div" is the ratio of datain_width to dataout_width (the ratio of output data width), "mul" is the ratio of dataout_width to datain_width, and "cnt" is a counter. This design only involves two situations: the width of input data is twice that of output data and the width of output data is twice that of input data. Therefore, in this paper, when "mul" is equal to 2, it means that the width of output data is larger than that of input data, while "div"=2 means that the width of input data is larger than that of output data.

According to the above code, when "r_rst" is 1, the counter "cnt" is cleared. The function of "cnt" is that when the width of input data is larger than that of output data, the data in "fifo_mem" should be output in high order or low order, and the value of "cnt" determines whether the high order (the first half of a data in "fifo_mem") or the low order (the second half of a data in "fifo_mem") should be assigned to data_read. When the enable signal "r_en" of the read clock is 1, and both "empty" and "r_rst" are zero, it is judged whether the "mul" is equal to 2. In case the result is true, the data in the "fifo_mem" addressed by the read-address pointer "read_addr" added with 1 being taken as the high bits of long bit width data and the data in the "fifo_mem" addressed by "read_addr" being taken as the low bits of long bit width data synthesize a new data which has long bit width. Then assign the synthesized new data to "data_read". The above steps realize data bit width conversion and output long bit width data. If the result is false, and when "div" is equal to 2, for the data in "fifo_mem" addressed by "read_addr", firstly assign its high bits to "data_read", and modify the value of "cnt" so that the value assigned to "data_read" in the next clock cycle is the low bit value of the same data. And when the low-bit data is sent to "data_read", the value of "cnt" is restored to output the high bits of the next data.

3.2. Read-clock Domain Module

In the module of reading clock domain, an extended pointer of "read_addr", which is named "read_addr_ptr", is designed for asynchronous FIFO. The bit width of this extended pointer is one bit larger than the bit width of "read_addr", and its values from the second highest bit to the lowest bit correspond to the values from the highest bit to the lowest bit of "read_addr" respectively[7]. And the purpose of designing this pointer is to use it to judge the empty and full state of FIFO at this moment. However, to weaken the influence of metastability, the gray code form "read_addr_gray" of "read_addr_ptr" is used to judge the empty and full state[8].

The biggest feature of Gray code is that only one bit is different between two adjacent codes. For example, four bits of the four-bit binary number 0111 (corresponding to the decimal number 7) to 1000 (corresponding to the decimal number 8) in the ordinary coding form have changed. But for Gray codes, the Gray codes corresponding to decimal numbers 7 and 8 are 0100 and 1100, respectively. It can be seen that only one-bit changes. One of the reasons for the emergence of metastability is that there is not enough time to set up and maintain the data when it changes, so

effectively reducing the number of data changes can weaken the influence of metastability. The generation code of Gray code of "read_addr_ptr" is as follows:

```
assign read_addr_gray = (read_addr_ptr>>1) ^ read_addr_ptr;
```

As can be seen from the above code, Gray code is obtained by XOR operation between "read_addr_ptr" and the pointer obtained by performing arithmetic shifted right on "read_addr_ptr" itself.

When judging whether the FIFO is in an empty state, compare "read_addr_gray" with "write_addr_gray_sync" synchronized by two D flip-flops. When they are identical, it is judged that the FIFO is in an empty state, that is, "empty" is equal to 1. The flow chart corresponding to the module of the read-clock domain is as follows:

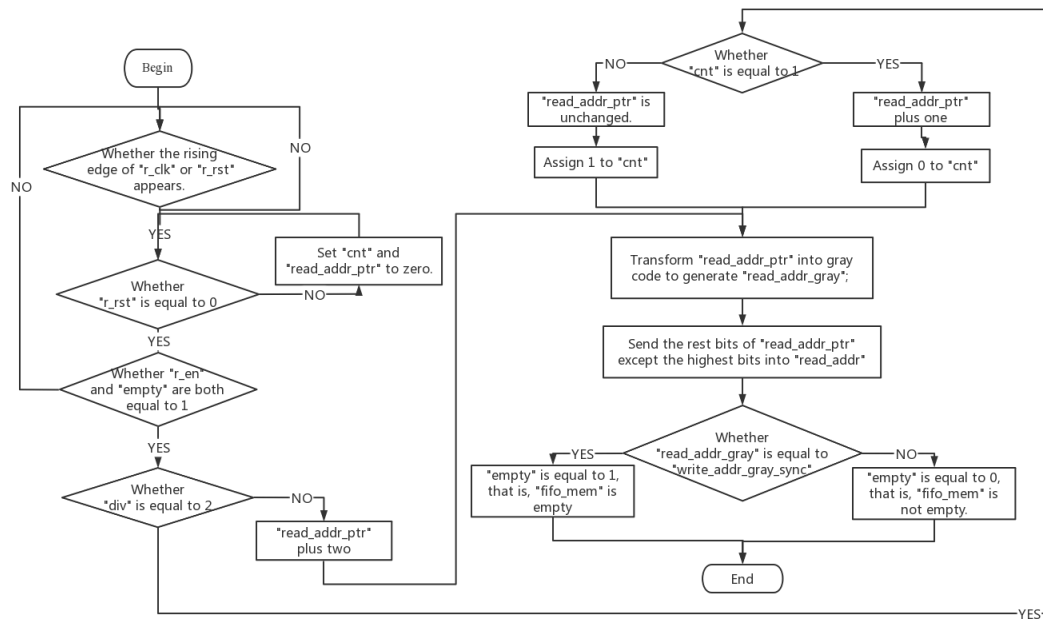


Fig. 2 The flow chart of read-clock domain

As shown in Fig.2, when the rising edge of "r_clk" or "r_rst" comes, judge whether the value of "r_rst" is zero at this time. If it is not equal to zero, reset "cnt" and "read_addr_ptr". Otherwise, judge whether the bit width of input data is twice that of output data at this time. If "div" is not equal to two, it means that the bit width of the output data at this time is twice that of the input data, and the "read_addr_ptr" related to the next addressing is the result of adding 2 to the current "read_addr_ptr".The reason is that two pieces of data with a bit width of N can be merged into one piece of data with a bit width of 2N, so two pieces of data (corresponding to two addresses) in "fifo_mem" are needed to send a piece of data to "data_read".Therefore, the next address is the current address plus 2. If "div" is equal to 2, judge whether "cnt" is equal to 0. In this paper, it is stipulated that when "div" is equal to 2, "data_read" reads the high-order data first, and then reads the low-order data. When "cnt" is equal to 0, according to the Verilog code of the "fifo_mem" module, the upper N bits of the addressed data will be sent to "data_read" in the next clock cycle, so the "read_addr_ptr" related to the next addressing will remain unchanged; When "cnt" is equal to 1, according to the Verilog code of "fifo_mem" module, the lower N bits of the addressed data will be sent to "data_read" in the next clock cycle. At this time, one data addressed in "fifo_mem" has been completely sent out, so the "read_addr_ptr" related to the next addressing is obtained by adding 1 to the current "read_addr_ptr".When the value of "read_addr_ptr" is updated, the data from the sub-high bit to the lowest bit is intercepted to "read_addr_ptr", and "read_addr_ptr" is converted to "read_addr_gray". At the same time, "read_addr_gray" is compared with "write_addr_gray_sync" to generate a read-empty signal "empty"[1].

3.3. Write-clock Domain Module

Like the read-clock domain module, the write-clock domain module also needs to set an extended pointer "write_addr_ptr" of the write address pointer, and it also needs to be converted to the form of Gray code (the conversion method is the same as that of obtaining "read_addr_gray"). "write_addr_gray" and "read_addr_gray_sync" should be used to judge whether the FIFO is full or not. Furthermore, because both of them are in the form of Gray code, the flag of FIFO full state is that the highest and second highest bits of "write_addr_gray" and "read_addr_gray_sync" are opposite, and the rest bits are the same. The flow chart corresponding to the module of the write-clock domain is as follows:

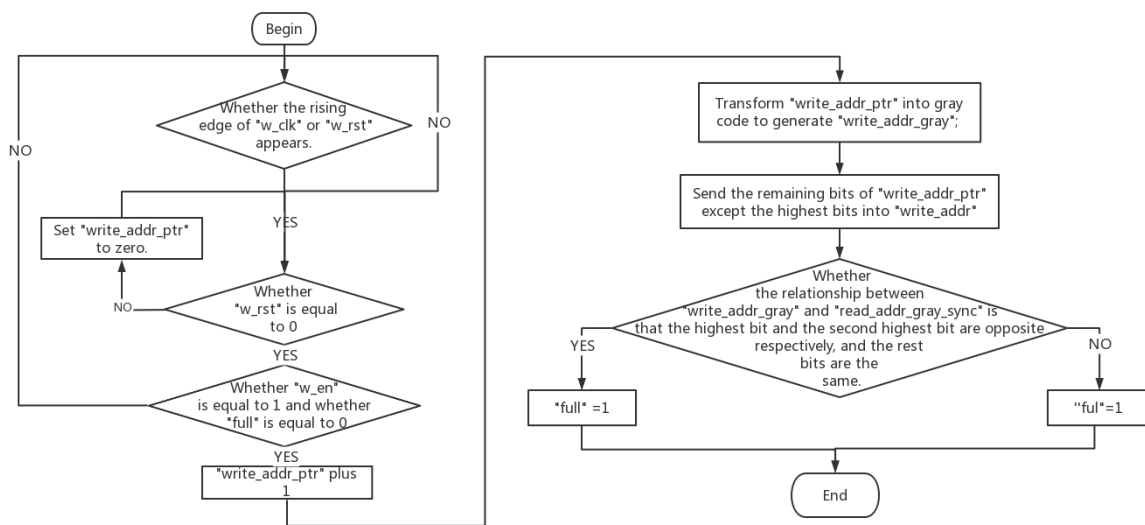


Fig. 3 The flow chart of write-clock domain

As can be seen in Fig.3, when the rising edge of "w_clk" or "w_rst" comes, judge whether the value of "w_rst" is zero at this time, and if it is not zero, reset "write_addr_ptr". Otherwise, judge, whether "w_en" is equal to 1 and "full", is equal to 0, if so, add 1 to "write_addr_ptr". Then intercept the data from the next highest bit to the lowest bit of "write_addr_ptr" to "write_addr". and then convert "write_addr_ptr" into "write_addr_gray". At last, compare "write_addr_gray" with "read_addr_gray_sync" to determine whether the FIFO is full[1].

3.4. Clock-Domain Synchronization Module

In this paper, two cascaded D flip-flops are used to synchronize the clock signals of different clock domains[9]. "write_addr_gray_temp" and "read_addr_gray_temp" are transitional registers for delay operation. The Verilog code is as follows:

```

always @(posedge r_clk or posedge r_rst ) begin
    write_addr_gray_temp <= write_addr_gray;
    write_addr_gray_sync <= write_addr_gray_temp;
end
always @(posedge w_clk or posedge "w_rst") begin
    "read_addr_gray"_temp <= "read_addr_gray";
    "read_addr_gray_sync"<= "read_addr_gray"_temp;
end
  
```

As described in the above code, in the timing logic with "r_clk" and "r_rst" as sensitive signals, "write_addr_gray" is delayed twice to obtain the synchronized pointer "Write _ Addr _ Gray _ Sync"; In the timing logic with "w_clk" and "w_rst" as sensitive signals, the "read_addr_gray" is delayed twice, and the synchronized pointer "write_addr_gray_sync" is obtained. The purpose of delaying twice here is to weaken the influence of metastability[10]. Its principle is: assuming that the input of

the first-stage flip-flop does not meet its setting and holding time, its output data is metastable after the first pulse edge arrives. Then before the next pulse edge comes, the output metastable data must be stable after a recovery time, and the stable data must meet the setup time of the second-stage flip-flop. In case all the above conditions are met, when the next pulse edge comes, the second-stage flip-flop will not be metastable, because the data at its input meets its setup and hold time. Take the synchronization process of the read-clock domain as an example, as shown in Fig.4:

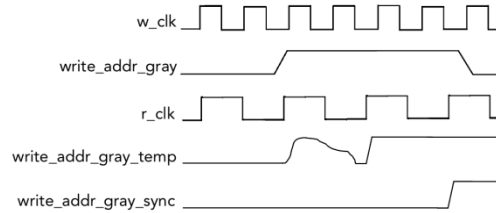


Fig. 4 The process of eliminating metastability through cascaded D flip-flops.

4. Results and Analysis of Simulation

Through the above description, this design is verified in two cases, one is that "mul" equals 2 and the other is that "div" equals 2.

4.1. "mul" is Equal to Two

When "mul" is equal to 2, set "datain_width" to 8 and "dataout _ width" to 16, and write "8'h01" to "8'h09" into "data_write"(data that is designed to write into the FIFO). The Simulation results using Modelsim software are as follows:

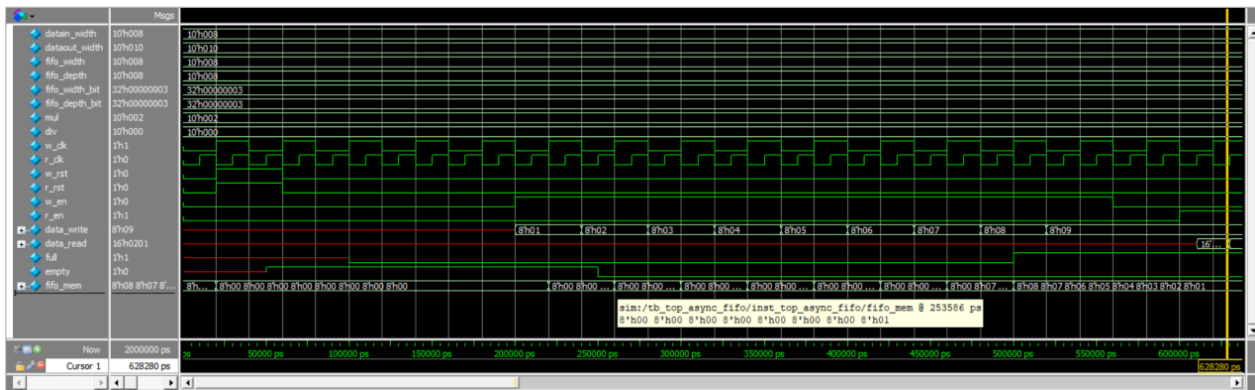


Fig. 5 The running result of the first half of the Simulation waveform when "mul" = 2

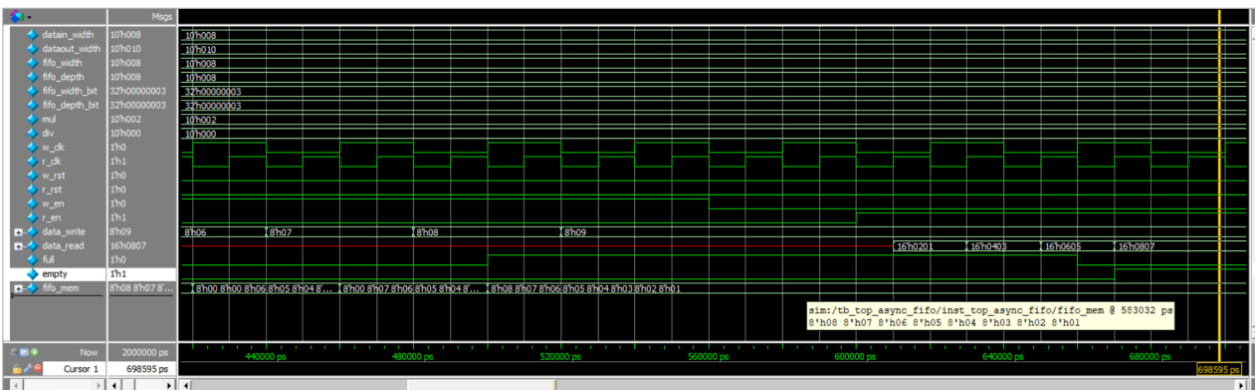


Fig. 6 The running result of the second half of the Simulation waveform when "mul" = 2

It can be seen from Fig.5 that "fifo_mem" is cleared before "data_write" starts transmitting data to FIFO. At this time, because "w_rst" and "r_rst" are both zero, "empty" is 1. However, when the time

is 300000ps, although 8'h01 is written in "fifo_mem", the "empty" is still equal to 1, which is the "pessimistic empty" phenomenon caused by the delay-operation of two-stage D flip-flops[1]. This phenomenon is because there are two "r_clk" delays in synchronizing the write pointer to the read-clock domain, so the "write_addr_gray_sync" sampled by the read-clock domain at the same time is smaller than or equal to the "write_addr_gray" of the write-clock domain. Meanwhile, the "fifo_mem" addressed according to the "write_addr" associated with "write_addr_gray" has been written, but the "read_addr_gray" is compared with the "write_addr_gray_sync" generated before two "r_clk", and the result is that "empty" is still equal to 1.

It can be seen from Fig.6 that two 8-bit data are successfully converted into 16-bit data and then output. Because the depth of FIFO is 8, but "data_write" wants to transmit 9 data to "fifo_mem", so "full" becomes 1 at 500000ps. And it can be seen from the number in the yellow box in the figure (the data in it is the data in "fifo_mem" at 583032ps) that when "full" equals 1, "fifo_mem" will no longer be written with data. And "data_read" has been read at 600000ps, but "full" doesn't jump to 0 until 660000ps, which is the "pessimistic full" phenomenon caused by the delay-operation of two-stage D flip-flops[1]. This phenomenon is because there are two "w_clk" delays in synchronizing the read pointer to the write-clock domain, so the "read_addr_gray_sync" sampled by the write-clock domain at the same time is smaller than or equal to the "read_addr_gray" of the read-clock domain. Meanwhile, the "fifo_mem" addressed according to the "read_addr" associated with "read_addr_gray" has been read, but the "write_addr_gray" is compared with the "read_addr_gray_sync" generated before two "w_clk", and the result is that "full" is still equal to 1.

The phenomena of "pessimistic empty" and "pessimistic full" do not affect the accuracy of data, and can leave a certain margin for the storage space of FIFO, which is helpful to prevent the data buffered in FIFO from being lost due to data overflow or underflow.

4.2. "div" is Equal to Two

When "div" is equal to 2, set "datain_width" to 16 and "dataout _ width" to 8. The Simulation results using modelsim software are as follows:

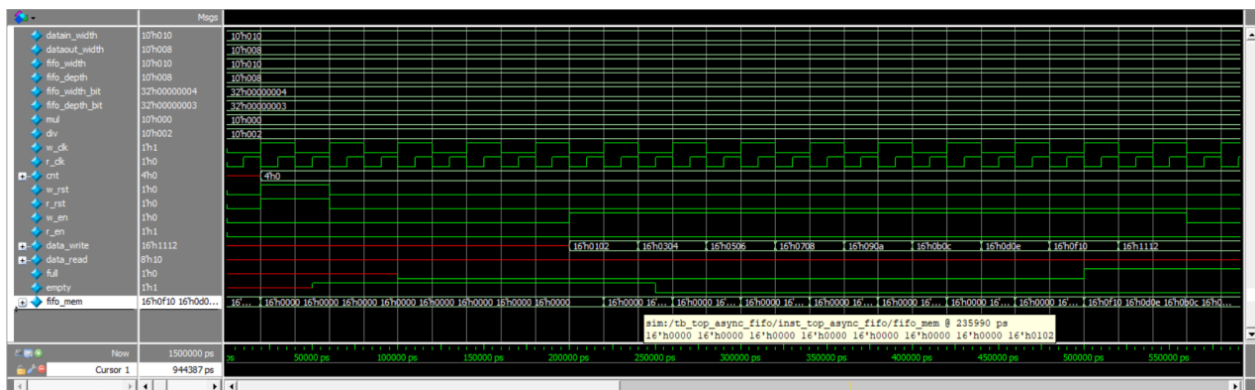


Fig. 7 The running result of the first half of the Simulation waveform when "div" =2

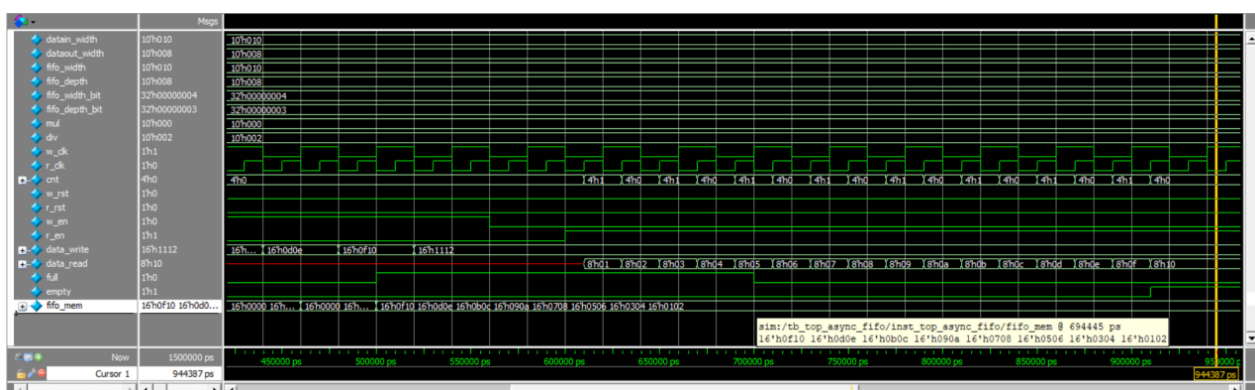


Fig. 8 The running result of the second half of the Simulation waveform when "div" = 2

As shown in Fig.7, the bit width of FIFO is changed to 16 at the same time as the bit width of "datain_width". At 235990ps, there is data in "fifo_mem", but at this time "empty" is still equal to 1, so it can be concluded that "pessimistic empty" also occurs from 220000ps to 250000 PS. Because the depth of FIFO is 8, but "data_write" wants to transmit 9 data to "fifo_mem", so "full" becomes 1 after the eighth data "16'h0f10" is sent into "fifo_mem" at 500000ps.

As shown in fig.8, the 16-bit data is split into the form that the upper 8 bits are output first and the lower 8 bits are output later, which meets the design requirements. In addition, there is a phenomenon of "pessimistic full" between 610000ps and 700000ps. Finally, at 910000ps, all the data in "fifo_mem" are read out, and "empty" becomes 1 again.

5. Summary

In this design, an asynchronous FIFO with data bit width conversion and clock domain synchronization is designed using Verilog HDL. To verify the function of the design, two situations are tested in the testbench file by using ModelSim software: 1) the input data bit width is 16 and the output data bit width is 8; 2) The input data bit width is 8, and the output data bit width is 16. The final waveform simulation results verify the correctness of this design. At the same time, the phenomena of "pessimistic emptiness" and "pessimistic fullness" in the results are analyzed and explained. And no metastability can be observed in the final waveform, which indicates that the design of the weakened part of metastability is successful. The FIFO designed in this paper has the characteristics of data bit width conversion and no metastability, which can realize accurate data transmission when the data bit widths of the input and output devices are different, which shows that the design has certain commercial value and industrial practicability.

References

- [1] Clifford E. Cummings. Simulation and Synthesis Techniques for Asynchronous FIFO Design. SNUG San Jose 2002, 2002.
- [2] Lifan Huang, Xueren Zheng. Metastability research in FPGA design. Microelectronics, 2011, 41(2):5.
- [3] Yingqiao Wang, Xiaoguang Hu, Fu Li. Design and realization of multi-function serial port with high-speed/large-capacity/asynchronous FIFO. 2016 12th IEEE International Conference on Control and Automation (ICCA), 2016.
- [4] 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.
- [5] Di Patrizio Stanchieri, G., De Marcellis, A., Palange, E., & Faccio, M. A true random number generator architecture based on a reduced number of FPGA primitives. AEU - International Journal of Electronics and Communications, 2019, 105, 15–23.
- [6] Bing Qi Liu, Ming Zhe Liu, Gang Yang, Xiao Bo Mao, Huai Liang Li. Research and Design of Asynchronous FIFO Based on FPGA. Applied Mechanics and Materials, 2014, 644–650, 3440–3444.
- [7] Kumar, A., Shankar, S., & Sharma, N. Verification of Asynchronous FIFO using System Verilog. International Journal of Computer Applications, 2014, 86(11), 16–20.
- [8] Sokolov, P. E., Butov, A. S., & Peplov, I. S. RTL design techniques of asynchronous FIFO memory blocks with area and power consumption optimizations. Interactive Science, 2016, 4, 74–78.
- [9] Hyoungh-Kook Kim, Laung-Terng Wang, Yu-Liang Wu, Wen-Ben Jone. Testing of Synchronizers in Asynchronous FIFO. Journal of Electronic Testing, 2019, 29(1), 49–72.
- [10] Govinda Sannena, Bishnu Prasad Das. Metastability immune and area efficient error masking flip-flop for timing error resilient designs. Integration, 2018, 61, 101–113.