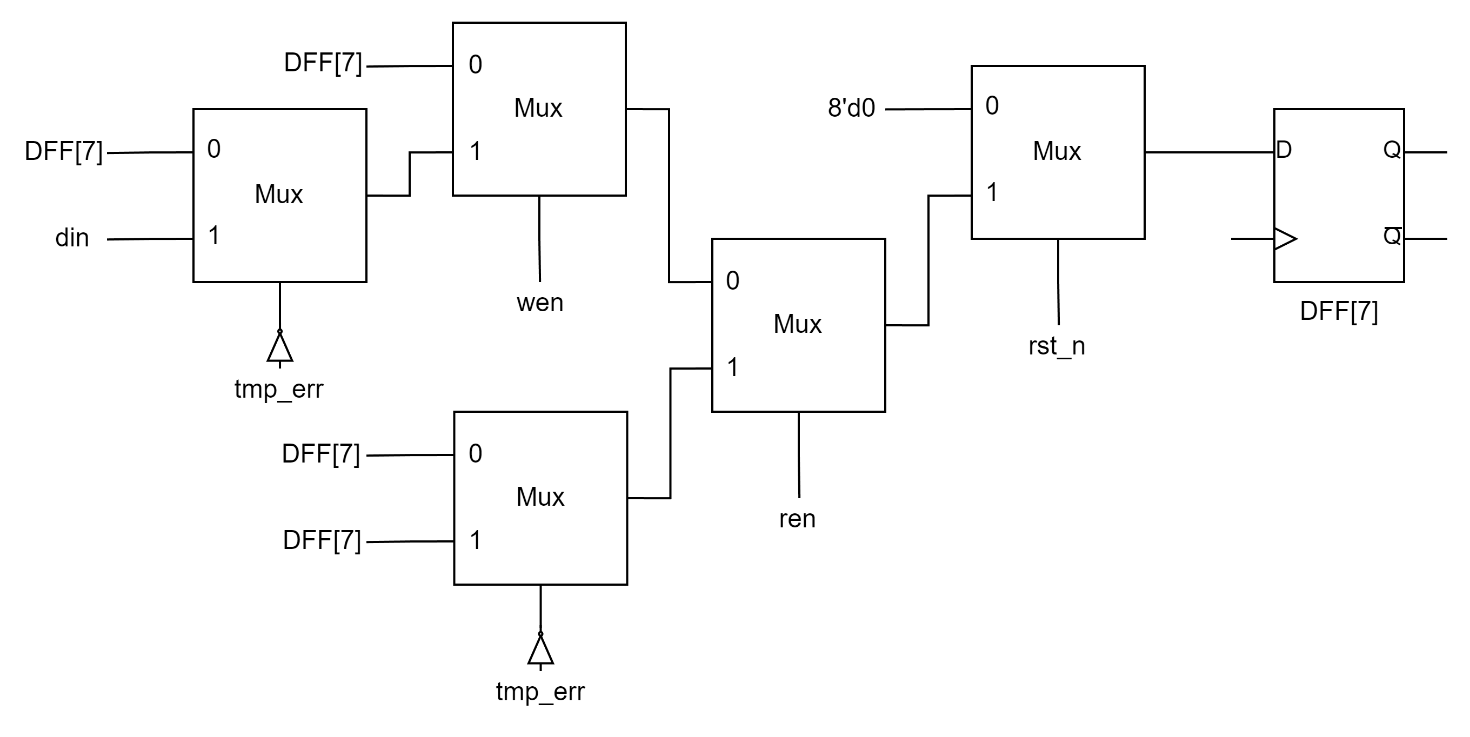
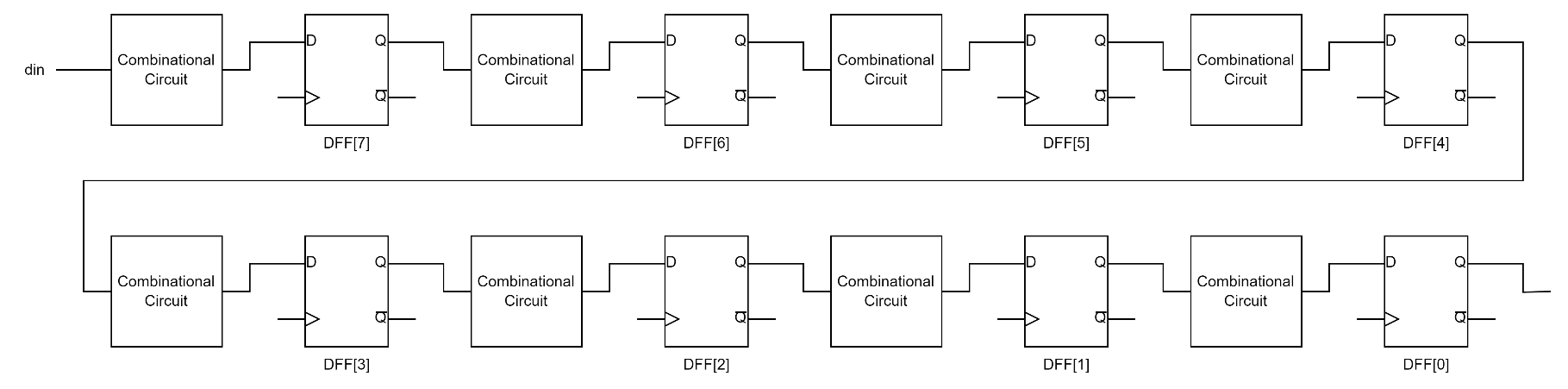
**Hardware Design and Lab: Lab3**

1. **Advanced Question: First-In First Out (FIFO) Queue**
2. **Block Diagram**

****

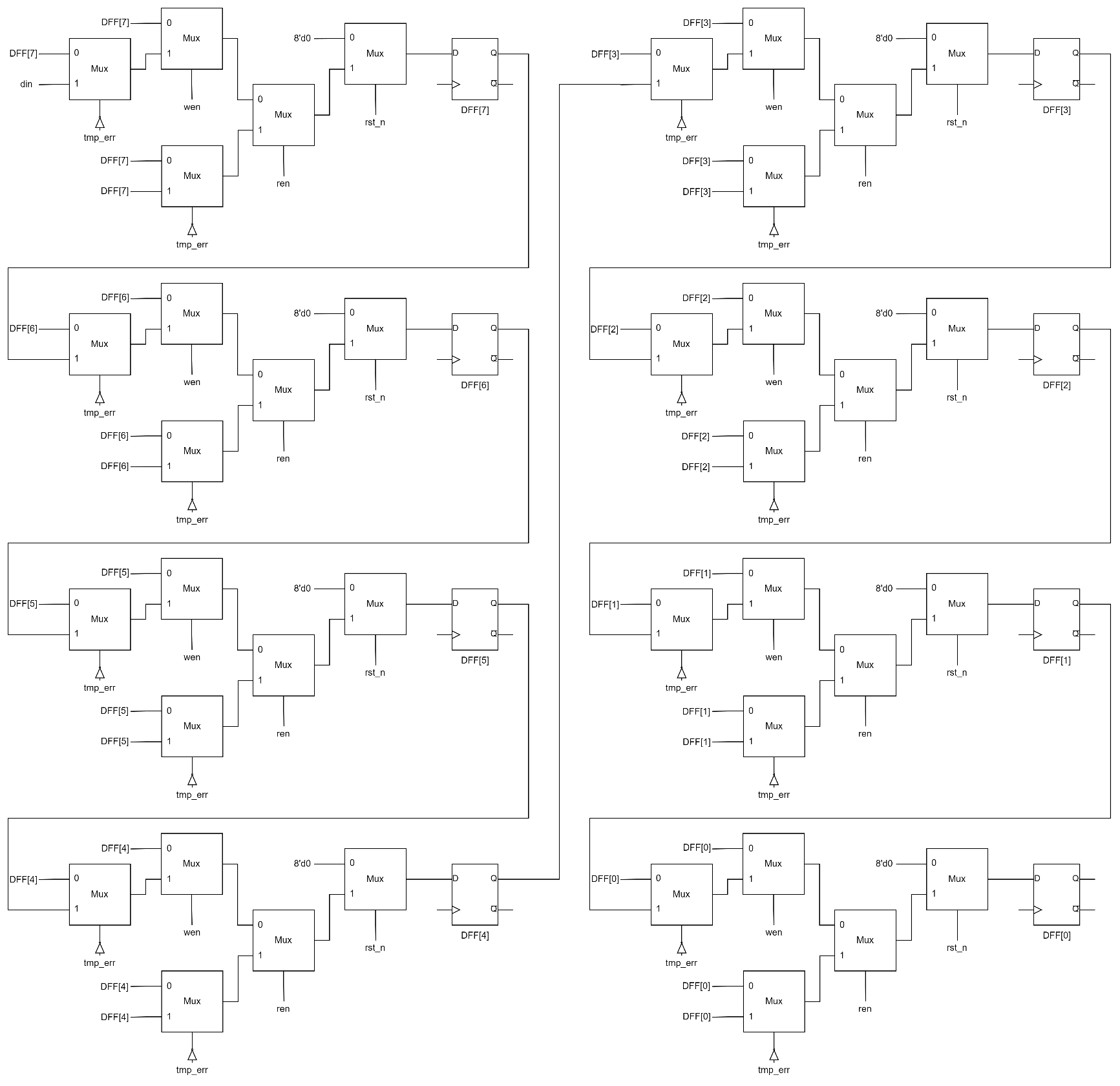
Picture 1.1

This is the combinational circuit for each DFF in the queue.

****

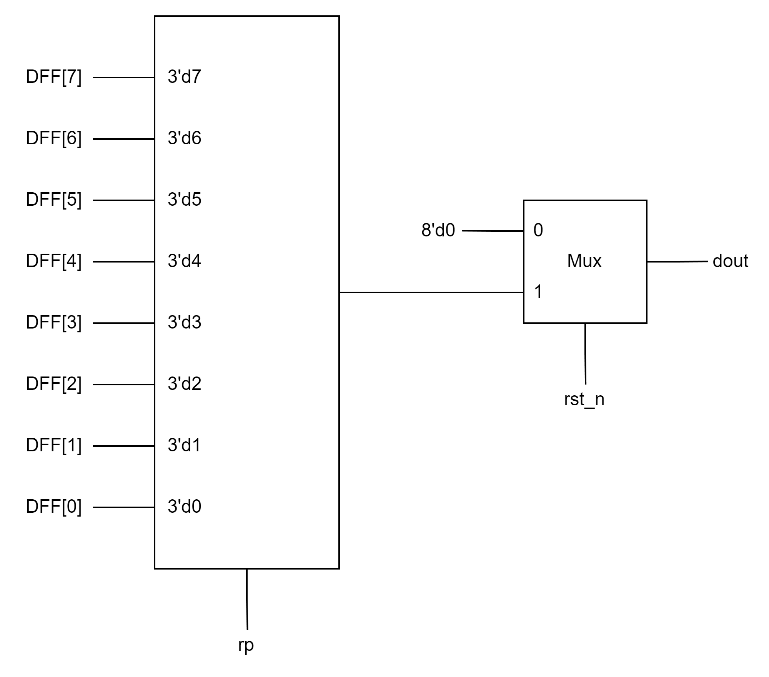
Picture 1.2

This is the architecture of the 8-bits FIFO Queue.

****

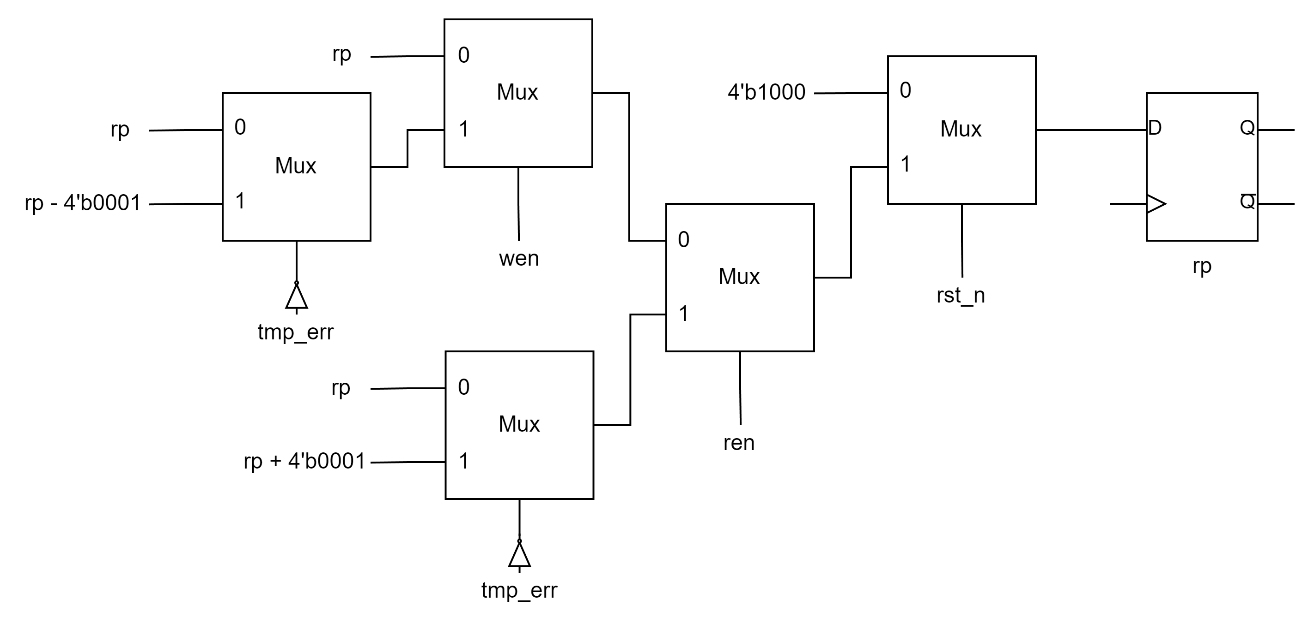
Picture 1.3

This is the block diagram of the 8-bits FIFO Queue.

****

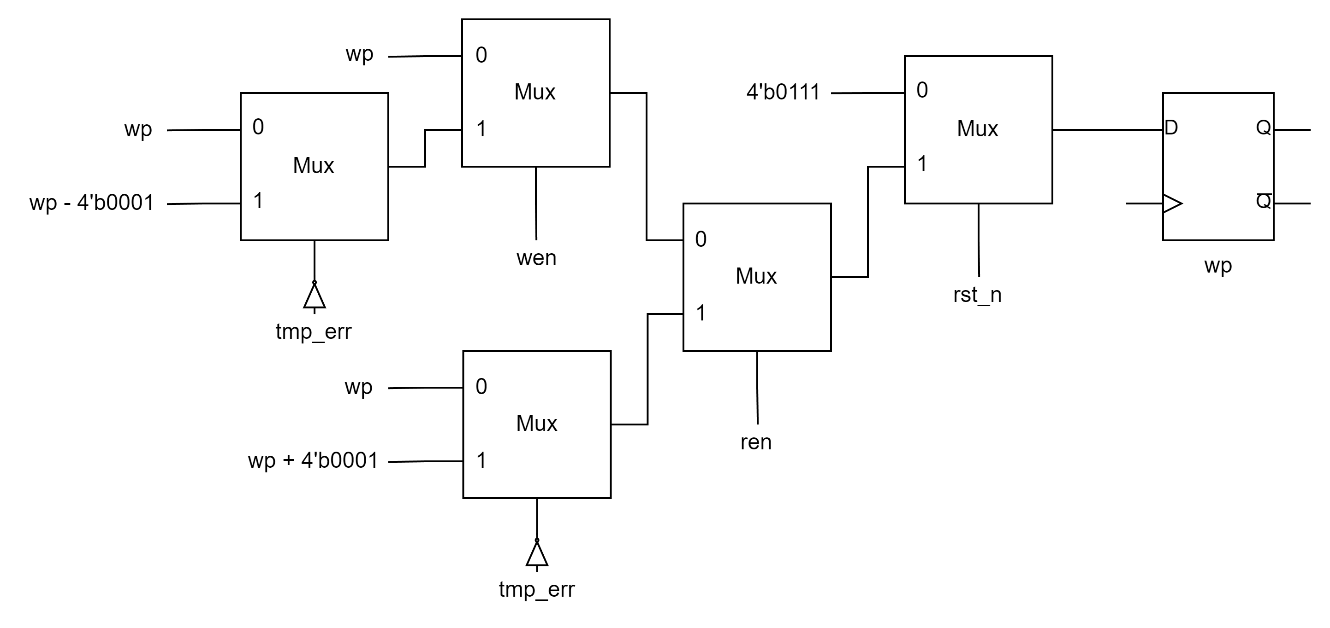
Picture 1.4

Choose the **dout** by **rp (read pointer)**.

****

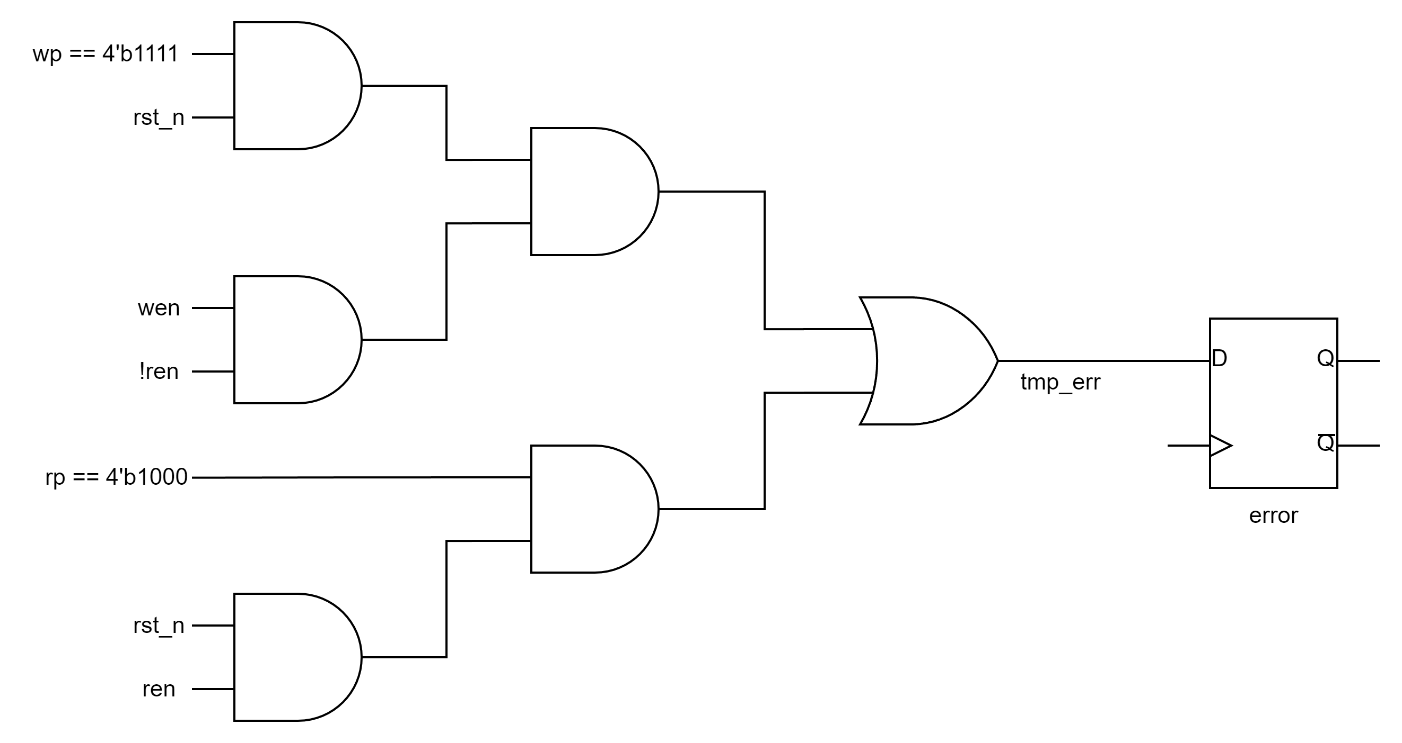
Picture 1.5

This is how I define **rp**. (read pointer)

****

Picture 1.6

This is how I define **wp**. (write pointer)

****

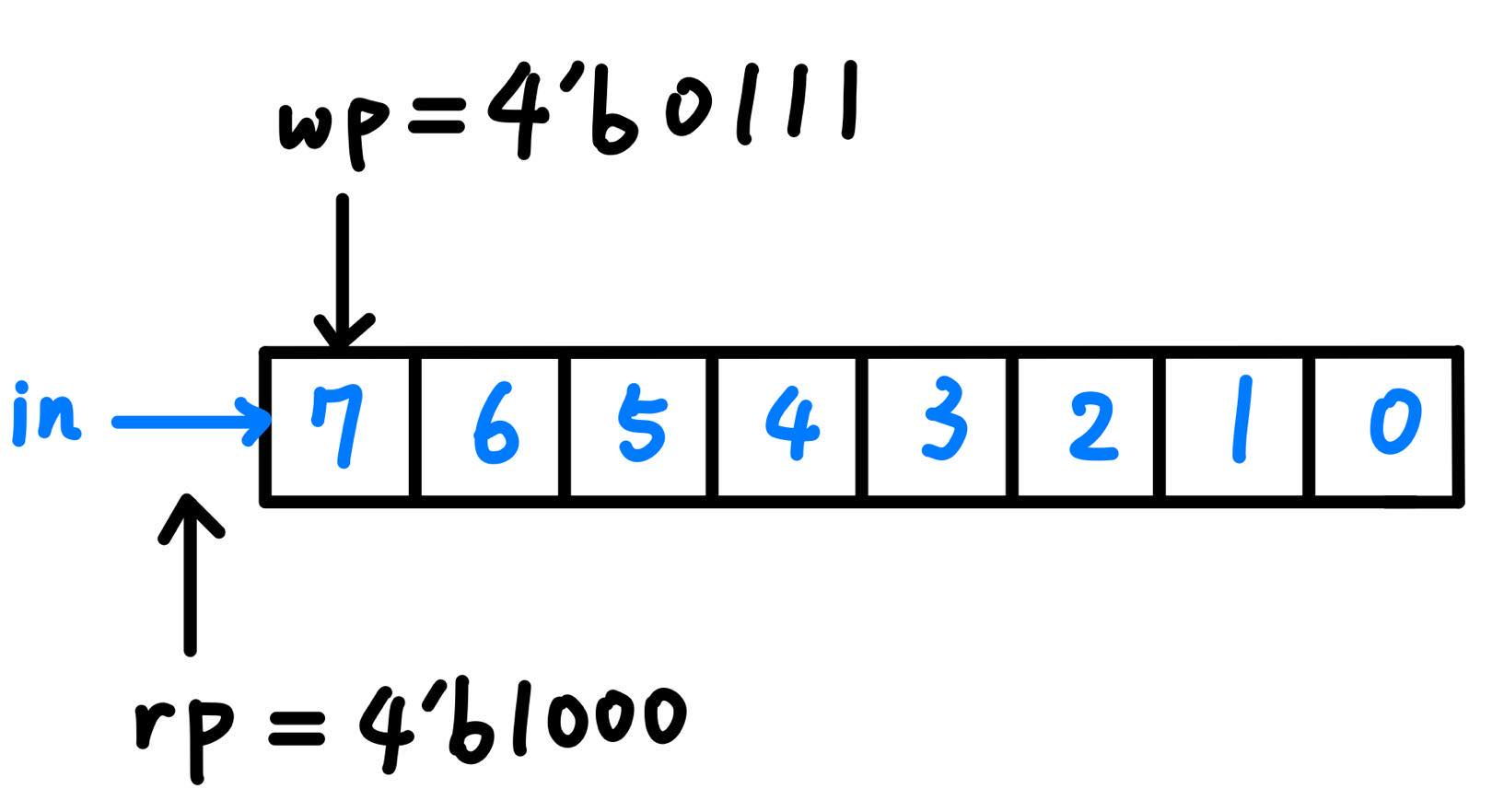
Picture 1.7

This is how I define **error** and **tmp\_err**.

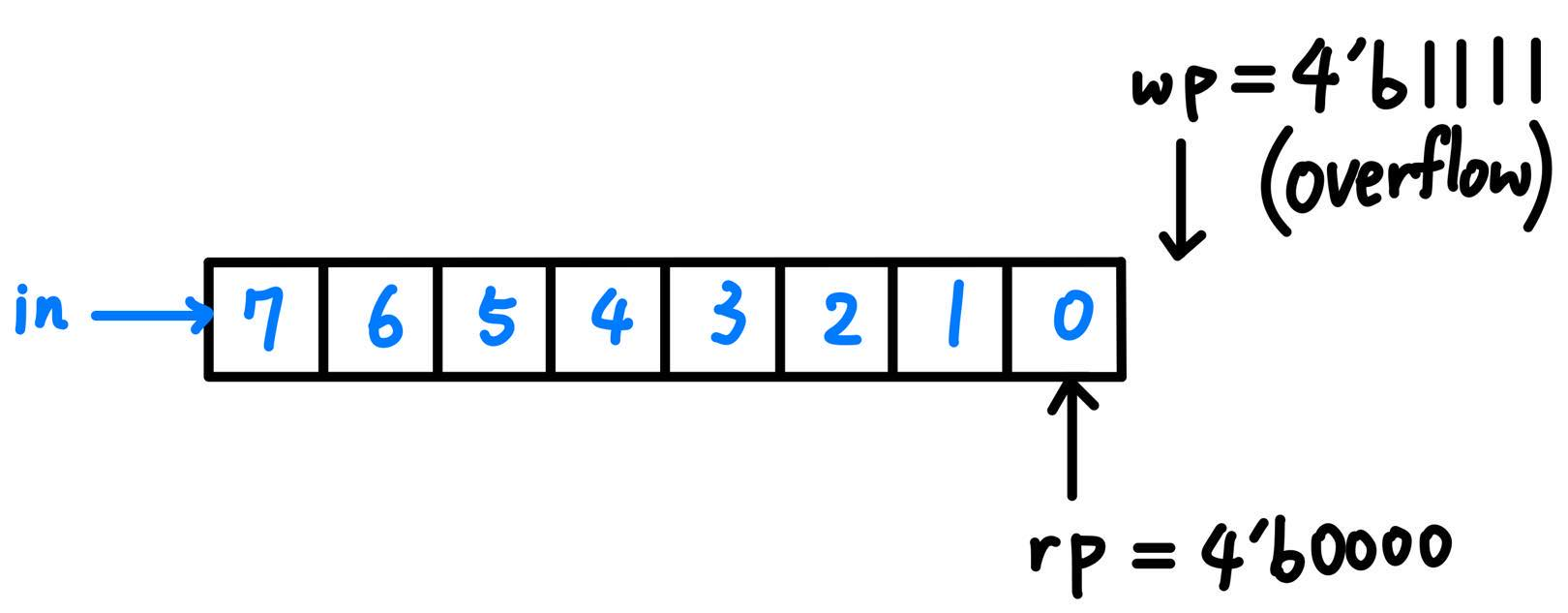
1. **Explanation**

In the beginning, I reset **rp** (read pointer) and **wp** (write pointer) to **4'b0111** and **4'b1111** respectively. If writing data into queue is successful, both of **rp** and **wp** will -1 as moving the pointers to the next DFF. If reading data out of queue is successful, both of **rp** and **wp** will +1 as moving the pointers to the previous DFF. The data in the queue will only be passed to the next DFF while writing is successful. Otherwise, they will be kept in the same DFF. The error cases are that

**rp == 4'b1000** and **ren == 1'b1**, which means reading failed, or **wp == 4'b1111 (overflow when writing)**, **ren == 1'b0** and **wen == 1'b1**, which means writing failed. In these cases, **error** will be pulled up for a clock cycle.

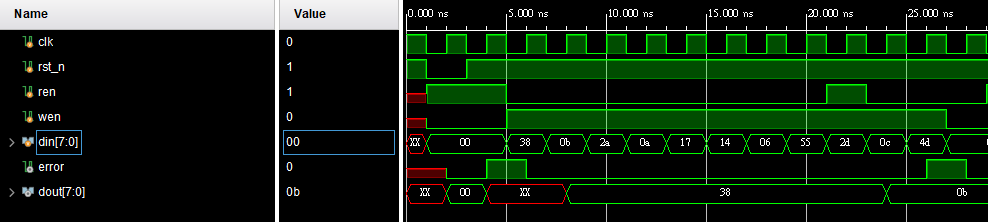


Picture 1.8 Reading failed



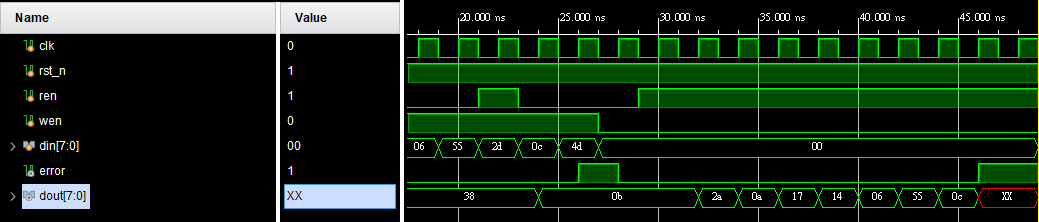
Picture 1.9 Writing failed

1. **Testbench**

****

Picture 1.10 Wave form 1

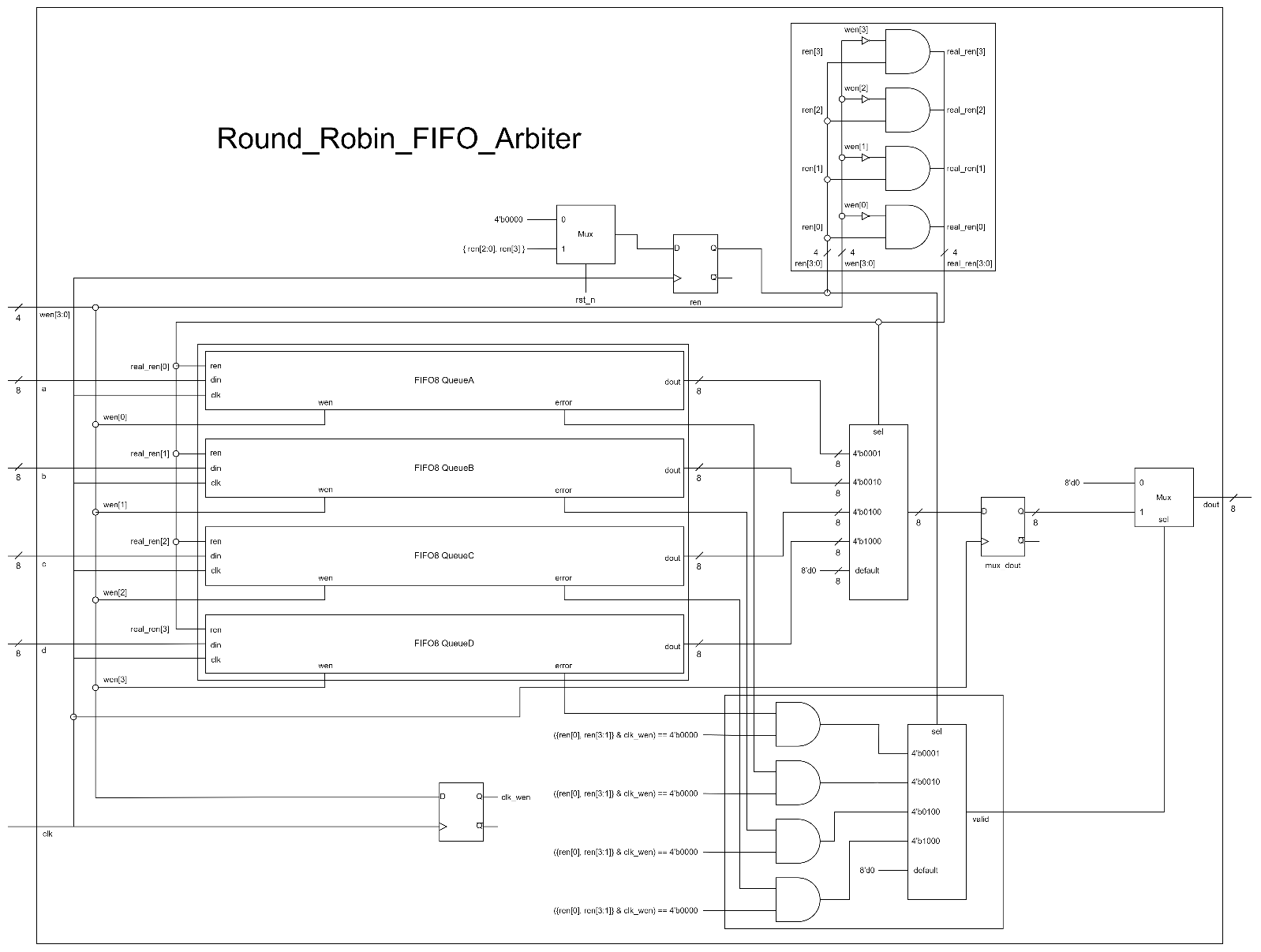
I test my design with the input in the lecture slide to see if there is something wrong or not. As the wave form showed in Picture 1.10m it looks like everything is correct. However, it only tests the condition of writing failed. Therefore, I let it read out all the data in the queue to check the condition of reading failed.



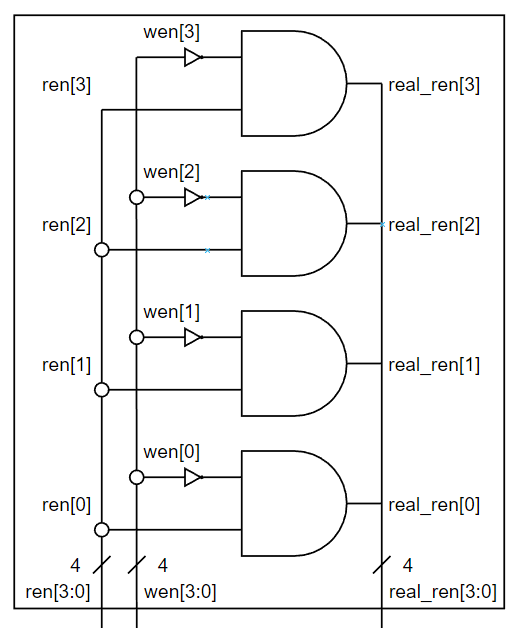
Picture 1.11 Wave form 2

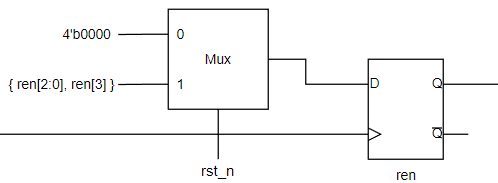
As the result showed in Picture 1.11, we can see that the condition of reading fail is also correct.

1. **Advanced Question: Round-Robin FIFO Arbiter**
2. **Block Diagram**

****

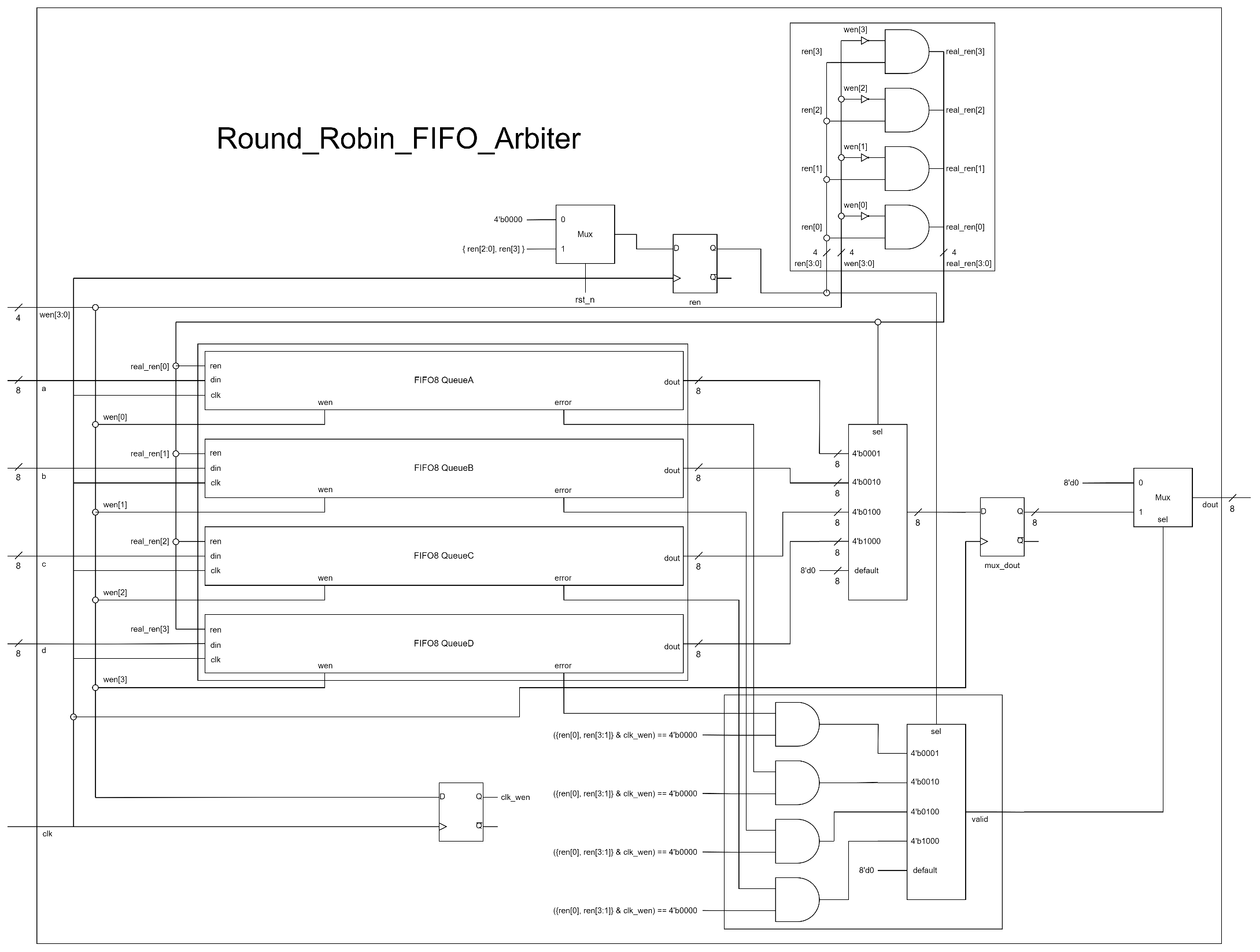
Picture 2.1 The whole design of Round-Robin FIFO Arbiter

****

****

Picture 2.3 ren

Picture 2.2 real\_ren

****

Picture 2.4 Valid and Out

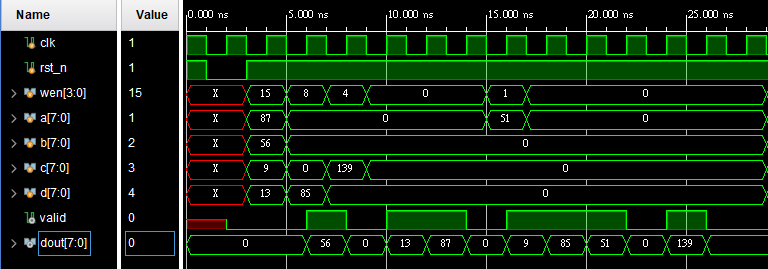
Out

Valid

1. **Explanation**

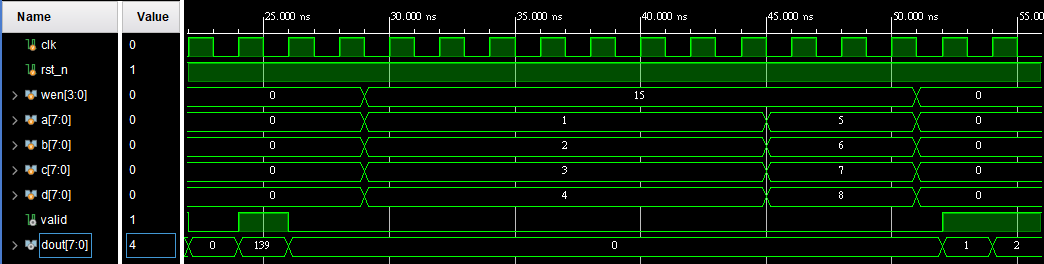
**FIFO8** here is totally the same as the **FIFO8** in the advanced question, First-In First Out (FIFO) Queue. For **valid**, **error** in **FIFO8** is synchronized, so **valid** isn't needed to be contained in a DFF. However, because **valid** is not contained in a DFF, which means that it will change if **wen** changes, so I use a DFF to keep **wen** for a clock cycle which called **clk\_wen** to help me define **valid** correctly. For **dout**, I use a 4-bit counter called **ren** to decide which queue should be access. Different from the advanced question, First-In First Out (FIFO) Queue, if both **ren** and **wen** is 1'b1, it will write instead of read. Therefore, I define an extra reg called **real\_ren** to fix this problem.

1. **Testbench**

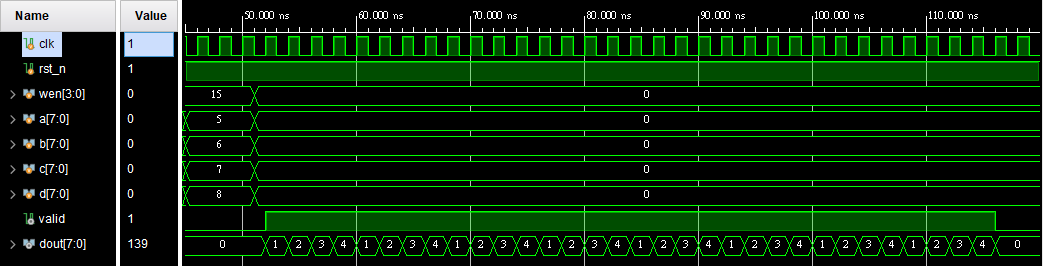
****

Picture 2.5 wave form 1

First, I use the input data from lecture slide to test if there is something wrong or not. As the result showed in Picture 2.5, we can see that everything works correctly. However, here we can't test the situation of writing failed. Therefore, after clear the queues, I let it write 8 times of same numbers and 1 times of different numbers. After that, I pop every queue 9 times to check if everything is correct.



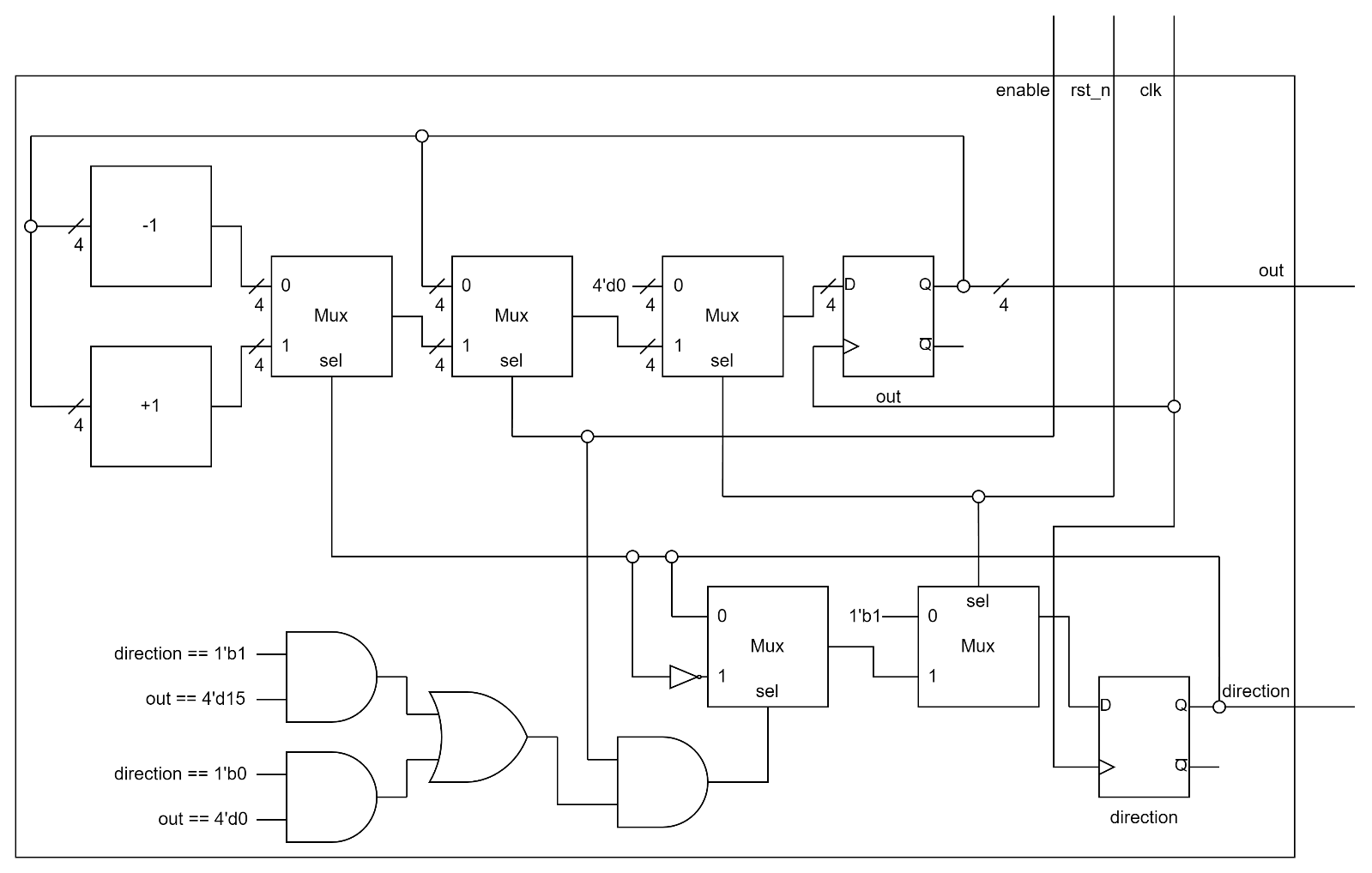
Picture 2.6 wave form 2



Picture 2.7 wave form 3

From Picture 2.6 and Picture 2.7, we can see that when the condition of reading failed or writing failed happened, my design can work correctly.

1. **Advanced Question: 4-bit Ping-Pong Counter**
2. **Block Diagram**

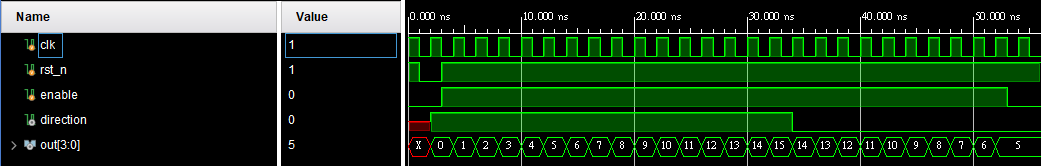
****

Picture 3.1 The whole design of 4-bit Ping-Pong Counter

1. **Explanation**

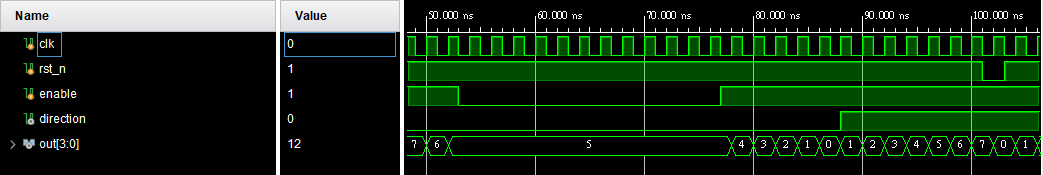
I use two DFF to contain **out** and **direction** respectively. If **direction == 1'b1** and **out == 4'd15**, which means the counter hits the upper bound, and **direction == 1'b0** and **out == 4'd0**, which means the counter hits the lower bound, **direction** should be inverted.

1. **Testbench**

****

Picture 3.2 wave form 1

For the beginning of the testbench, I test the counter without changing **enable** and **rst\_n**. As the result showed in Picture 3.2, we can see that everything works properly.



Picture 3.3 wave form 2

And then I test **enable** and **rst\_n** to check if they work or not. From the result in Picture 3.3. It seems that everything is correct.

1. **Advanced Question: 4-bit Parameterized Ping-Pong Counter**
2. **Block Diagram**