数字电路实验课程讲义

2021-12

实验 4、FIFO 实验

实验目标:

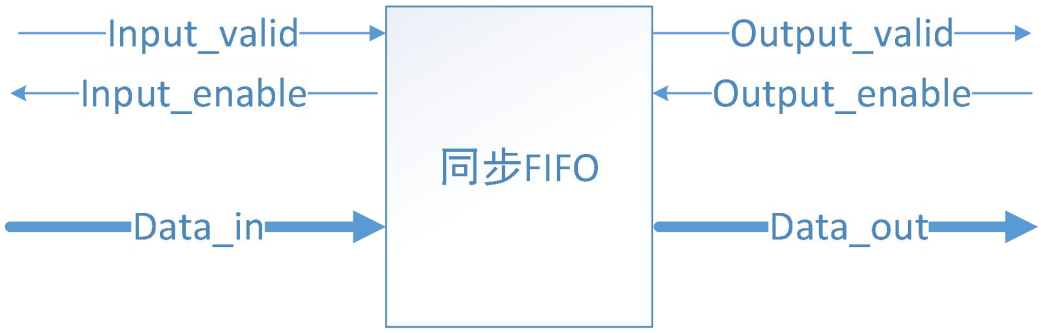
1、 熟悉 verilog 编程、调试

2、 熟悉 FIFO 工作原理

3、 实现功能较复杂的数字电路

实验内容:

FIFO： First In First Out，先进先出，先进入队列的数据先输出



如图，这是一个同步的 FIFO，其中：

1. data_in 为数据输入端口，位宽为 8 位。
2. data_out 为数据输出端口，位宽为 4 位。
3. 同步 FIFO 的缓冲区大小为 64bit。FIFO 的一项数据宽度是 4bits，每次写入 8bit，每次读出 4bit。
4. 每次读出的数据，先读出写入时 8bit 的低 4 位，后读出高 4 位。即写入 8bit 的低四位是第一个数据，高四位是第二个数据。
5. input_valid 和 input_enable 控制着数据的传入。
6. output_valid 和 output_enable 控制着数据的输出。
7. 对于同一组 valid 和 enable 信号而言，只有 2 者同时有效时才会工作，即 input_valid 和 input_enable 同时有效时才读入 data_in 的数据。同理如 output。

请设计一个同步 FIFO 模块，要求如下：

1. 同步的 FIFO 只存在一种工作状态，即要么只接收数据不输出数据，要么只输出数据，不接收数据。
2. FIFO 只有在填满数据之后才向外输出数据，只有在读空之后才允许写入数据。
3. 自行编写 Testbench

附加实验

修改同步 FIFO，要求：

1. FIFO 中有数据就可以向外读出数据，FIFO 没有填满就可以向内写入数据；
2. 考虑读和写同时发生的处理情况。
3. 自行编写 Testbench

FIFO 示例

FIFO 设计文件

```
`timescale 1ns / 1ps
module sim_fifo(
    input clk,
    input rstn,
    input write,
    input read,
    input [7:0] data,
    output reg [7:0] out
    );

    reg [7:0] mem [1:0];

    reg write_addr, read_addr;

    always @(posedge clk or negedge rstn) begin
        if (rstn == 0)begin
            write_addr <= 1'b0;
            read_addr <= 1'b0;
        end
        else begin
```

```
            if (write == 1'b1) begin
                write_addr <= ~write_addr;
                mem[write_addr][7:0] <= data[7:0];
            end
            else if (read == 1'b1) begin
                read_addr <= ~read_addr;
                out[7:0] <= mem[read_addr][7:0];
            end
            else begin

            end
        end
    end


endmodule
```

测试文件

```
`timescale 1ns / 1ps
module test_sim_fifo(

    );

    reg clk, rstn;
    reg [7:0] data;
    wire [7:0] out;
    reg write, read;

    sim_fifo inst_sim_fifo_0(.clk(clk),
        .rstn(rstn),
        .write(write),
        .read(read),
        .data(data),
        .out(out));

    always #2 begin
        clk = ~clk;
    end

    initial begin
        clk = 1'b0;
        rstn = 1'b1;
        write = 1'b0;
        read = 1'b0;
        #1 rstn = 1'b0;
```

```
        #2 rstn = 1'b1;
    end

    always begin
        #3;
        data = $random()%9'b1_0000_0000;
    end

    always begin
        #5;
        write = 1'b1;
        #6;
        write = 1'b0;
        read = 1'b1;
        #6;
        write = 1'b1;
        read = 1'b0;
        #3;
        $finish;
    end

endmodule
```

波形文件