浙江大学实验报告

课程名称：数字系统　　任课老师：李宇波

实验名称：Dedicate Microprocessor lab　　实验日期：2023/5/30

1 实验目的和要求

1.1 实验目的

To learn how to implement a dedicated microprocessor.

了解如何实现一个专用处理器。

1.2 实验要求

1) Manually design and implement on a FPGA a dedicated microprocessor to input one 8-bit value, and then determine whether the input value has an equal number of 0 and 1 bits. The microprocessor outputs a 1 if the input value has the same number of 0's and 1's; otherwise, it outputs a 0. For example, the number 10111011 will produce a 0 output; whereas, the number 00110011 will produce a 1 output. The algorithm is shown next. Draw the datapath and the corresponding FSM state diagram, FSM circuit, list the control words.

   在 FPGA 上设计和实现一个专用处理器，能够输入一个 8-bit 值，并判定 1 和 0 的数量是否相等。如果 1 和 0 的数量相等，处理器输出 1；否则，输出 0。算法如下。

```
1     Count = 0              // for counting the number of 1 bits
2     INPUT N
3     WHILE (N ≠ 0){
4        IF (N(0) = 1) THEN  // least significant bit of N
5           Count = Count + 1
6        END IF
7        N = N >> 1          // shift N right one bit
8     }
9     OUTPUT (Count = 4) // output 1 if the test (Count = 4) is
                         true
```

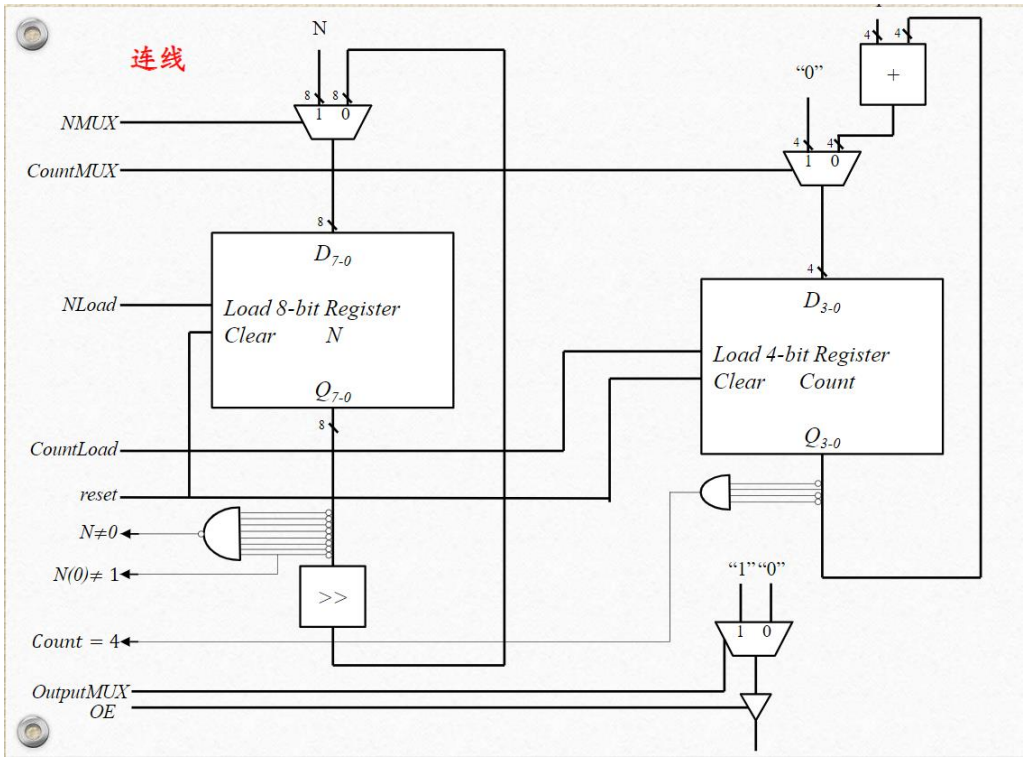
2) Implement the datapath circuit and FSM circuit separately in Verilog module, connect them together by using a top module, and implement it in your Basys3 board. Connect the switches in board to your input pin, and connect the output pin to a LED.

   分别用 Verilog 模块实现数据通路电路和有限状态机电路，再用顶层模块将二者结合，并在 Basys3 板上实现。开关与输入管脚相连，LED 与输出管脚相连。

2 数据通路设计

1） 确定器件：根据算法，数据通路需要：两个寄存器，分别存储 N 和 count；一个加法器；三个数据选择器；其他外围电路。

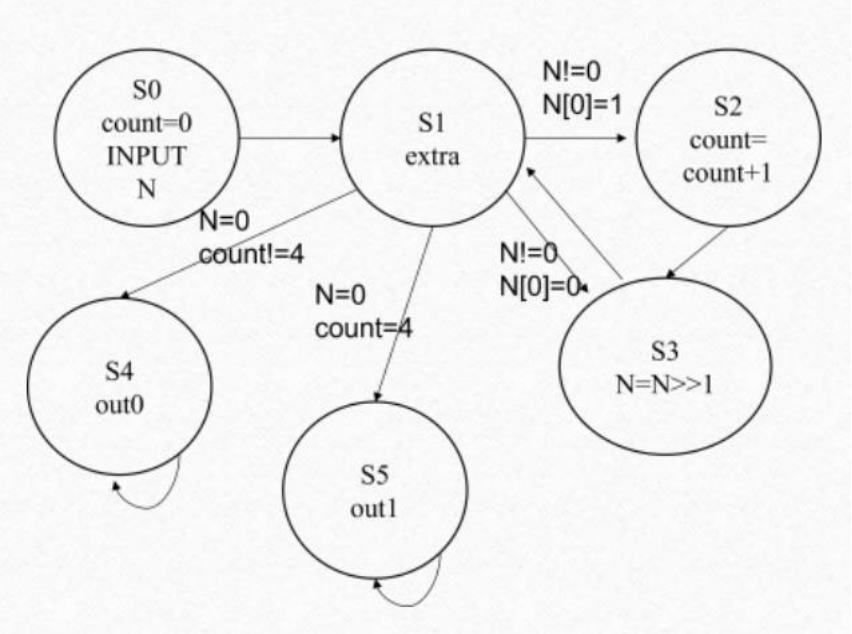
2） 连线



3 控制单元设计

1） 设计状态和状态图如下：

状态	执行操作
S0	读取 N
S1	额外状态，便于状态转移
S2	count+1
S3	N 右移一位
S4	N=0,读取完毕，若 count!=4，输出 0
S5	N=0,读取完毕，若 count=4，输出 1



实现上述状态转换需要三种状态反馈信号，分别用于判断 N 是否为 0、count 是否为 4 和 N 的第一位是否为 0。

2） 输出控制字设计

	NMUX	Count_MUX	NLoad	Count_Load	Output_MUX	OE
S0	1	1	1	1	X	0
S1	X	X	0	0	X	0
S2	X	0	0	1	X	0
S3	0	X	1	0	X	0
S4	X	X	0	0	0	1
S5	X	X	0	0	1	1

4 行为级模型代码

行为级模型代码设计按层次分类：顶层模块；状态机模块、数据通路模块；（数据通路下的）三个数据选择器模块、两个寄存器模块、一个加法器模块。完整代码见附录 1 。

5 行为级仿真

编写仿真代码（见附录 2），模拟实际的 N 输入。

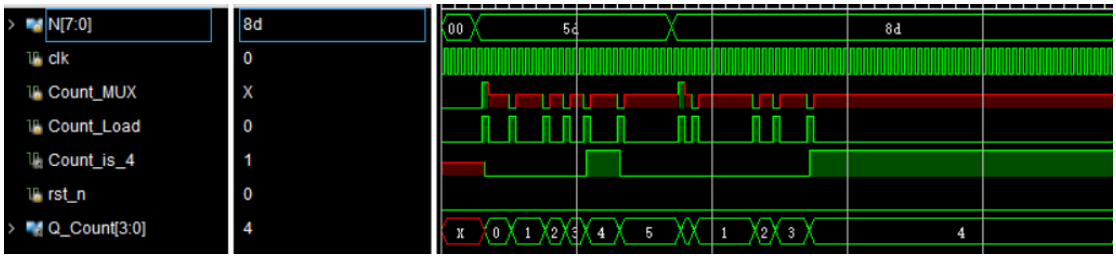
1） 数据通路仿真（主要考察 count 是否准确）

输入 N=01011101，状态经历：

s0->s1->s3->s1->s2->s3->s1->s3->s1->s2->s3->s1->s2->s3->s1->s2->s3->s1->s3->s1->s2->s3->s1->s4 最后 count=5;

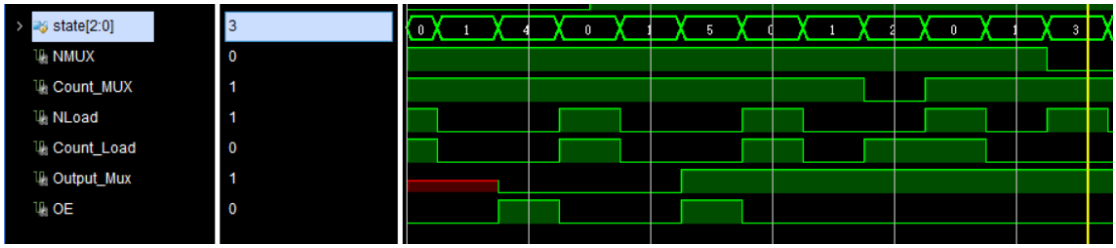
输入 N=10001101，状态经历：

s0->s1->s2->s3->s1->s3->s1->s3->s1->s3->s1->s2->s3->s1->s2->s3->s1->s3->s1->s2->s3->s1->s5 最后 count=4。



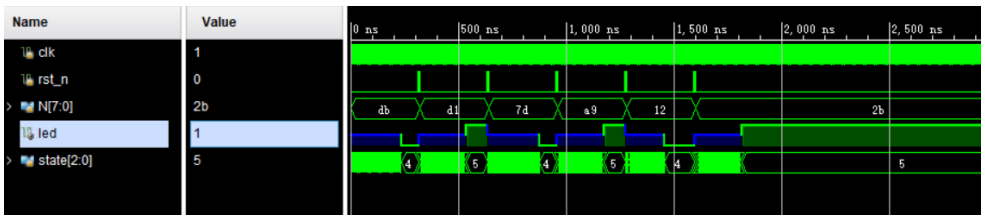
2） 有限状态机仿真

先后测试了 s1->s4, s1->s5, s1->s2, s1->s3, 无误。状态对应的控制字输出也正确对应。



3） 处理器仿真

设计六组输入，分别为 11011011、11010001、01111101、10101001、00010010、00101011，得到 led 输出分别为 0、1、0、1、0、1，正确。



综合以上仿真结果，认为设计无误。

附录 1

1. 顶层模块

```
module Microprocessor (
    input clk,
    input rst_n,
    input [7:0] N,
    output led
);
    wire NMUX, Count_MUX, NLoad, Count_Load, OE, Output_Mux;
    wire N0_is_1, N_is_0, Count_is_4;
    FSM fsm1(.clk(clk), .rst_n(rst_n), .N_is_0(N_is_0), .N0_is_1(N0_is_1), .Count_is_4(Count_is_4), .NMUX(NMUX), .Count_MUX(Count_MUX),
     .NLoad(NLoad), .Count_Load(Count_Load), .Output_Mux(Output_Mux), .OE(OE));
    Datapath d1(.clk(clk), .N(N), .NMUX(NMUX), .Count_MUX(Count_MUX), .NLoad(NLoad), .Count_Load(Count_Load), .rst_n(rst_n),
    .Output_Mux(Output_Mux), .OE(OE), .N_is_0(N_is_0), .N0_is_1(N0_is_1), .Count_is_4(Count_is_4), .Data_output(led));
endmodule
```

2. FSM 模块

```
module FSM (
    input clk,
    input rst_n,
    input N_is_0,
    input N0_is_1,
    input Count_is_4,
    output reg NMUX,
    output reg Count_MUX,
    output reg NLoad,
    output reg Count_Load,
    output reg Output_Mux,
    output reg OE
);
//states design
parameter s0 = 0;   parameter s1 = 1;
parameter s2 = 2;   parameter s3 = 3;
parameter s4 = 4;   parameter s5 = 5;
reg [2:0] state = 0, state_next = 0;

//next state logic
```

```
always @(*) begin
    case (state)
        s0: state_next = s1;
        s1: begin
            if(!N_is_0)begin
                if(N0_is_1) state_next = s2;
                else    state_next = s3;
            end
            else if(Count_is_4) state_next = s5;
            else state_next = s4;
        end
        s2: state_next = s3;
        s3: state_next = s1;
        s4: state_next = s4;
        s5: state_next = s5;
    endcase
end

always @(posedge clk) begin
    if(rst_n) state = 0;
    else state = state_next;
end

//output control signal
always @(*) begin
    case (state)
        s0: begin
            NMUX = 1;           NLoad = 1;
            Count_MUX = 1;      Count_Load = 1;
            Output_Mux = 1'bx;  OE = 0;
        end
        s1: begin
            NMUX = 1'bx;        Count_MUX = 1'bx;
            NLoad = 0;          Count_Load = 0;
            Output_Mux = 1'bx;  OE = 0;
        end
        s2: begin
            NMUX = 1'bx;        Count_MUX = 0;
            NLoad = 0;          Count_Load = 1;
            Output_Mux = 1'bx;  OE = 0;
        end
        s3: begin
            NMUX = 0;           Count_MUX = 1'bx;
            NLoad = 1;          Count_Load = 0;
```

```
            Output_Mux = 1'bx;  OE = 0;
        end
        s4: begin
            NMUX = 1'bx;        Count_MUX = 1'bx;
            NLoad = 0;          Count_Load = 0;
            Output_Mux = 0;     OE = 1;
        end
        s5: begin
            NMUX = 1'bx;        Count_MUX = 1'bx;
            NLoad = 0;          Count_Load = 0;
            Output_Mux = 1;     OE = 1;
        end
    endcase
end

endmodule
```

3. 数据通路模块

```
module Datapath (
    input [7:0] N,
    input NMUX,
    input Count_MUX,
    input NLoad,
    input Count_Load,
    input rst_n,
    input Output_Mux,
    input OE,
    input clk,
    output reg N_is_0,
    output reg N0_is_1,
    output reg Count_is_4,
    output reg Data_output
);

wire [7:0] D_N, Q_N;
reg [7:0] N_left_move;
wire [3:0] D_Count, Q_Count, Q_Count_Add_1;
wire data_out;

add addCount(.in1(4'b0001), .in2(Q_Count), .out(Q_Count_Add_1));
mux muxN(.sw(NMUX), .in1(N), .in2(N_left_move), .out(D_N));
```

```
mux
muxCount(.sw(Count_MUX), .in1(4'b0000), .in2(Q_Count_Add_1), .out(D_Cou
nt));
mux muxOut(.sw(Output_Mux), .in1(1'b1), .in2(1'b0), .out(data_out));
RegisterN
regN(.clk(clk), .load(NLoad), .clear(rst_n), .N_in(D_N), .N_out(Q_N));
RegisterCount
regCount(.clk(clk), .load(Count_Load), .clear(rst_n), .count_in(D_Count
), .count_out(Q_Count));

always @(*) begin
    N_is_0 <= (Q_N == 0);
    N0_is_1 <= (Q_N[7] == 1);
end
//not figure out it's combination or time
always @(*) begin
    N_left_move <= Q_N << 1;
end

always @(*) begin
    Count_is_4 <= (Q_Count == 4);
end

always @(*) begin
    Data_output <= OE?data_out:1'bz;
end
endmodule
```

4. 加法器模块

```
module add (
    input [3:0] in1,
    input [3:0] in2,
    output reg [3:0] out
);

always @(*) begin
    out <= in1 + in2;
end

endmodule
```

5. 数据选择器模块

```
module mux (
    input sw,
```

```
    input [7:0]in1, // in1 = N or "0"
    input [7:0]in2,
    output reg [7:0]out
);

always @(*) begin
    if (sw) out = in1;
    else out = in2;
end

endmodule
```

6. 寄存器模块

```
module RegisterN (
    input load,
    input clk,
    input clear,
    input [7:0] N_in,
    output reg [7:0] N_out
);

always @(posedge clk) begin
    if(clear)
        N_out = 0;
    else if(load)
        N_out = N_in;
end

endmodule
```

附录 2

1. FSM 仿真代码

```
initial
begin
    //test s1 to s4
    #10
    N_is_0 = 1; Count_is_4 = 0;
    //test s1 to s5
    #10 rst_n = 1;
    #10 rst_n = 0;
    N_is_0 = 1; Count_is_4 = 1;
    //test s1 to s2
    #20 rst_n = 1;
    #10 rst_n = 0;
    N_is_0 = 0; N0_is_1 = 1;
    //test s1 to s3
    #20 rst_n = 1;
    #10 rst_n = 0;
    N_is_0 = 0; N0_is_1 = 0;
end
```

2. 数据通路仿真代码

```
initial
begin
    #50
    N = 8'b01011101;
    //S0->s1->s3(1011101)->s1->s2->s3(011101)->s1->s3(11101)->s1->s2->s3
->s1->s2->s3(101)->s1->s2->s3->s1->s3->s1->s2->s3->s1->s4
    //s0
    #10 NMUX = 1; Count_MUX = 1; NLoad = 1; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
```

```
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0;
Output_Mux = 0; OE = 1;
```

```
    #50
    N = 8'b10001101;
    //s0->s1->s2->s3(0001101)->s1->s3->s1->s3->s1->s3(1101)->s1->s2->s3->s1->s2->s3->s1->s3->s1->s2->s3->s1->s5
    #10 NMUX = 1; Count_MUX = 1; NLoad = 1; Count_Load = 1; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0; Output_Mux = 1'bx;
```

```
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    #10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;

    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    #10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0;
Output_Mux = 1; OE = 1;

    //S1
    //#10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0; OE =
0; Output_Mux = 1'bx;
    //s2
    //#10 NMUX = 1'bx; Count_MUX = 0; NLoad = 0; Count_Load = 1; OE = 0;
Output_Mux = 1'bx;
    //S3
    //#10 NMUX = 0; Count_MUX = 1'bx; NLoad = 1; Count_Load = 0; OE = 0;
Output_Mux = 1'bx;
    //S4
    //#10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0;
Output_Mux = 0; OE = 1;
    //S5
    //#10 NMUX = 1'bx; Count_MUX = 1'bx; NLoad = 0; Count_Load = 0;
Output_Mux = 1; OE = 1;

end
```

3. 处理器仿真代码

```
initial
begin
    //N = 11011011, 1s ne 0s, 0
    N = 8'b11011011;
    //N = 11010001, 1s e 0s, 1
    #300
    #10    rst_n = 1;
    #10    rst_n = 0;
    N = 8'b11010001;
    //N = 01111101, 1s ne 0s, 0
    #300
    #10    rst_n = 1;
    #10    rst_n = 0;
```

```
    N = 8'b01111101;
    //N = 10101001, 1s e 0s, 1
    #300
    #10    rst_n = 1;
    #10    rst_n = 0;
    N = 8'b10101001;
    //N = 00010010, 1s ne 0s, 0
    #300
    #10    rst_n = 1;
    #10    rst_n = 0;
    N = 8'b00010010;
    //N = 00101011, 1s e 0s, 1
    #300
    #10    rst_n = 1;
    #10    rst_n = 0;
    N = 8'b00101011;
end
```