实验一：组合电路设计

李雨轩

西安交通大学

计算机 2205 班

2204112913

1. 实验目的与实验内容

本次实验主要内容为组合电路设计，包括基础门电路设计（多输入门电路、复用器）和基础功能模块设计（编码器、译码器）。实验内容如下：

- 用Vivado实现基础门电路（多输入门电路、复用器等）的设计和测试；
- 用Vivado实现基础功能模块（编码器、译码器等）的设计与测试。

2. 实验要求

实验要求如下：

- 掌握Vivado或 Logisim 开发工具的使用，掌握以上电路的设计和测试方法；
- 记录设计和调试过程（Verilog 代码/电路图/表达式/真值表，Vivado仿真结果，Logisim 验证结果等）；
- 分析Vivado仿真波形/Logism 验证结果，注重输入输出之间的对应关系。

3. 实验过程及分析-1：基础门电路设计

基础门电路部分要求设计和测试一个给定表达式的多输入门电路和复用器，复用器将会包括一个简单的 2 路复用器、一个 8 路复用器，以及使用三态缓存器实现一个 4 路复用器。

我在 Vivado 中使用 Verilog 语言分别完成了多输入门电路（表达式已给定）、二路复用器、多路复用器（8 路）以及使用三态缓存器实现的四路复用器的设计。通过仿真测试，验证了电路的逻辑正确性和时序关系。复用器设计体现了组合逻辑电路的选择功能，测试时各输入信号的变化可以实时反映在输出信号中。

3.1. 多输入门电路

根据实验指导的要求，设计一个多输入门电路，逻辑表达式为：

$$x = \overline{(a \cdot \overline{b} \cdot c \cdot (d + e))}$$

Verilog 代码 在Vivado平台设计如下的 Verilog 代码(Code-1)：

```
module multi_input_gate(
    input a, b, c, d, e,
    output x
);
    assign x = ~(a & ~b & c & (d | e));
endmodule
```

Code 1. 多输入门电路设计代码.

仿真 testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 testbench。(Code-2)：

```
module tb_multi_input_gate;
    reg a, b, c, d, e;
    wire x;
    multi_input_gate uut (
        .a(a),
        .b(b),
        .c(c),
        .d(d),
        .e(e),
        .x(x)
    );

    integer i;
    initial begin
        // 使用for循环遍历所有输入组合（5 个输入，共 32 种组合）
        for (i = 0; i < 32; i = i + 1) begin
            {a, b, c, d, e} = i;
            #10;
        end

        $finish;
    end

    // 监控信号变化
    initial begin
        $monitor("At time %t: a = %b, b = %b, c = %b, d = %b, e = %b -> x = %b",
                 $time, a, b, c, d, e, x);
    end

endmodule
```

Code 2. 多输入门电路 testbench 代码.

在创建的 TestBench 文件中，我编写测试激励代码，代码每隔 10ns 更改一次输入，遍历每一个可能的输出（构造了 2^5 个不同的五元组），以确保覆盖每一种输入。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation。选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 1 所示。[1]

3.2. 二路复用器

根据实验指导的要求，设计一个二路复用器。二路复用器的逻辑表达式为：

$$y = s \cdot d_1 + \overline{s} \cdot d_0$$

其中，d_0 和 d_1 为两个数据输入，s 为选择信号，y 为输出。

[1] 出于排版美观的考量，多输入门电路的仿真波形（图 1）放在文章的最后

真值表

S	$D1$	$D0$	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-3)：

```
module two_way_mux (
    input d0, d1, s,
    output y
);
    assign y = (s ? d1 : d0);
endmodule
```

Code 3. 二路复用器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-4)：

```
module tb_two_way_mux;
    reg d0, d1, s;
    wire y;
    two_way_mux uut (
        .d0(d0),
        .d1(d1),
        .s(s),
        .y(y)
    );

    initial begin
        d0 = 0;
        d1 = 0;
        s = 0;

        #10 {s, d1, d0} = 3'b000;
        #10 {s, d1, d0} = 3'b001;
        #10 {s, d1, d0} = 3'b010;
        #10 {s, d1, d0} = 3'b011;
        #10 {s, d1, d0} = 3'b100;
        #10 {s, d1, d0} = 3'b101;
        #10 {s, d1, d0} = 3'b110;
        #10 {s, d1, d0} = 3'b111;

        #10 $finish;
    end

    // 监控信号变化
    initial begin
        $monitor("At time %t: d0 = %b, d1 = %
    b, s = %b -> y = %b", $time, d0, d1, s, y
    );
    end
endmodule
```

Code 4. 二路复用器 testbench 代码.

在创建的 Testbench 文件中，我编写了测试激励代码，代码每隔 10ns 更改一次输入，遍历每一种可能的输入组合（共 2^3 个不同的三元组），以确保覆盖每一种输入情况。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation，选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 2 所示。[2]

3.3. 八路复用器

根据实验指导的要求，设计一个八路复用器。八路复用器的逻辑功能是根据选择信号选择一个输入信号作为输出。其逻辑表达式为：

$$y = d_0 \cdot \overline{s_2} \cdot \overline{s_1} \cdot \overline{s_0} + d_1 \cdot \overline{s_2} \cdot \overline{s_1} \cdot s_0 + \cdots + d_7 \cdot s_2 \cdot s_1 \cdot s_0$$

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-5)：

```
module eight_way_mux (
    input wire [7:0] d,
    input wire [2:0] s,
    output wire y
);
    assign y = (s == 3'b000) ? d[0] :
               (s == 3'b001) ? d[1] :
               (s == 3'b010) ? d[2] :
               (s == 3'b011) ? d[3] :
               (s == 3'b100) ? d[4] :
               (s == 3'b101) ? d[5] :
               (s == 3'b110) ? d[6] :
                               d[7];
endmodule
```

Code 5. 八路复用器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-6)：

```
module tb_eight_way_mux;
    reg [7:0] d;
    reg [2:0] s;
    wire y;
    eight_way_mux uut (
        .d(d),
        .s(s),
        .y(y)
    );
    initial begin
        d = 8'b00000000;
        s = 3'b000;
        // 遍历所有可能的输入组合
        repeat (512) begin
            #10 {d, s} = {d, s} + 1;
        end
        #10 $finish;
    end
    // 监控信号变化
    initial begin
        $monitor("At time %t: d = %b, s = %b
    -> y = %b", $time, d, s, y);
    end
endmodule
```

Code 6. 八路复用器 testbench 代码.

[2] 出于排版美观的考量，二路复用器的仿真波形（图 2）放在文章的最后

在创建的 Testbench 文件中，我编写了测试激励代码，每隔 10ns 更改一次输入信号和选择信号，遍历所有可能的输入组合（共 $2^8 \cdot 2^3$ 种组合）。为了简化测试，这里选择了一些典型的输入组合，以确保覆盖每一种选择信号情况下的输出。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation，选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 3 所示。[3]

3.4. 基于三态缓存器的四路复用器

根据实验指导的要求，设计一个基于三态缓存器的 4 路复用器。该复用器有 4 个数据输入 d_0、d_1、d_2、d_3，2 个选择输入 s_1、s_0，以及一个输出 y。利用三态缓存器实现，每次只有一个数据输入被选通到输出。

三态缓存器的真值表如下：

E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-7)：

```
module four_way_mux_tristate (
    input d0, d1, d2, d3,
    input [1:0] s,
    output y
);
    wire y0, y1, y2, y3;
    // 使用三态缓冲器实现复用器
    assign y0 = (s == 2'b00) ? d0 : 1'bz;
    assign y1 = (s == 2'b01) ? d1 : 1'bz;
    assign y2 = (s == 2'b10) ? d2 : 1'bz;
    assign y3 = (s == 2'b11) ? d3 : 1'bz;

    assign y = (s == 2'b00) ? y0 :
               (s == 2'b01) ? y1 :
               (s == 2'b10) ? y2 :
               (s == 2'b11) ? y3 : 1'bz;

endmodule
```

Code 7. 基于三态缓存器的四路复用器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-8)：

```
module tb_four_way_mux_tristate;
    reg d0, d1, d2, d3;
    reg [1:0] s;
    wire y;

    four_way_mux_tristate uut (
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3),
        .s(s),
        .y(y)
    );

    integer i;
    initial begin
        // 遍历所有输入组合
        for (i = 0; i < 64; i = i + 1) begin
            {s, d3, d2, d1, d0} = i;
            #10;
        end

        $finish;
    end

    // 监控信号变化
    initial begin
        $monitor("At time %t: s = %b, d0 = %b, d1 = %b, d2 = %b, d3 = %b -> y = %b",
    $time, s, d0, d1, d2, d3, y);
    end

endmodule
```

Code 8. 基于三态缓冲器的 4 路复用器 testbench 代码.

在创建的 Testbench 文件中，我编写了测试激励代码，代码每隔 10ns 更改一次输入，遍历每一种可能的选择输入组合（共 $2^4 \cdot 2^2$ 个不同的二元组），并对每个通道进行测试，以确保覆盖每一种输入情况。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation，选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 4 所示。[4]

4. 实验过程及分析-2：基础功能模块设计

基础功能模块的设计分为两个部分，分别是编码器和译码器。对于编码器，将首先使用三种方式：行为描述方式、结构化描述和数据流描述分别实现一个简单的 8-3 编码器（非优先级编码器）；然后再实现一个优先 8-3 编码器。对于译码器，将实现一个 3-8 译码器。

4.1. 8-3 编码器

根据实验指导的要求，设计一个 8-3 编码器，其功能是将 8 位输入信号编码为 3 位输出信号。当输入信号中有且仅有一位为高电平时，输出对应该高电平输入的位置编码。

真值表

$D7$	$D6$	$D5$	$D4$	$D3$	$D2$	$D1$	$D0$	$Y2$	$Y1$	$Y0$
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-9)：

[3] 出于排版美观的考量，八路复用器的仿真波形（图 3）放在文章的最后

[4] 出于排版美观的考量，基于三态缓存器的四路复用器的仿真波形（图 4）放在文章的最后

```verilog
// 结构化描述
module encoder_8to3_structural (
    input [7:0] d,
    output [2:0] y
);
    or (y[2], d[7], d[6], d[5], d[4]);
    or (y[1], d[7], d[6], d[3], d[2]);
    or (y[0], d[7], d[5], d[3], d[1]);
endmodule

// 行为级描述
module encoder_8to3_behavioral (
    input [7:0] d,
    output reg [2:0] y
);
    always @(*) begin
        y = 3'bXXX; // 默认输出为 XXX
        case (d)
            8'b00000001: y = 3'b000;
            8'b00000010: y = 3'b001;
            8'b00000100: y = 3'b010;
            8'b00001000: y = 3'b011;
            8'b00010000: y = 3'b100;
            8'b00100000: y = 3'b101;
            8'b01000000: y = 3'b110;
            8'b10000000: y = 3'b111;
            default: y = 3'bXXX;
        endcase
    end
endmodule

// 使用位运算和连续赋值语句描述数据流
module encoder_8to3_dataflow (
    input [7:0] d,
    output [2:0] y
);
    assign y[2] = |d[7:4];
    assign y[1] = |{d[7], d[6], d[3], d[2]};
    assign y[0] = |{d[7], d[5], d[3], d[1]};
endmodule
```

Code 9. 8-3 编码器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-10)：

```verilog
module tb_encoder_8to3;
    reg [7:0] d;
    wire [2:0] y;
    encoder_8to3_structural uut (
        .d(d),
        .y(y)
    );
    integer i;
    initial begin
        for (i = 0; i < 256; i = i + 1) begin
            d = i;
            if(256%i == 0) #10;
        end
        $finish;
    end
    initial begin
        $monitor("At time %t: d = %b -> y = %b", $time, d, y);
    end
endmodule
```

Code 10. 8-3 编码器 testbench 代码.

在创建的 TestBench 文件中，我编写了测试激励代码，代码每隔 10ns 更改一次输入，遍历所有合法的输入（只有一位为 1），以确保覆盖每一种输入情况。然后我又将代码10的第 14 行中跳过非法输入的部分略去后，使用行为级描述设计的电路（见10的第 15 行）再次进行测试，这时，对于非法输入，也能输出设定的值。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation。选择 Run Behavioral Simulation(行为仿真)。Vivado 编译设计源代码和 Testbench，生成仿真文件，只有合法输入的波形如图 6 所示，包含非法输入的波形如图 **??** 所示。[5]

4.2. 8-3 优先编码器

根据实验指导的要求，设计一个 8-3 优先编码器。优先编码器用于将 8 位输入信号中最高有效位的位置编码为 3 位二进制数。当多个输入为高电平时，输出对应最高优先级的输入信号。

真值表

$D7$	$D6$	$D5$	$D4$	$D3$	$D2$	$D1$	$D0$	$Y2$	$Y1$	$Y0$	V
0	0	0	0	0	0	0	0	X	X	X	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	X	0	0	1	1
0	0	0	0	0	1	X	X	0	1	0	1
0	0	0	0	1	X	X	X	0	1	1	1
0	0	0	1	X	X	X	X	1	0	0	1
0	0	1	X	X	X	X	X	1	0	1	1
0	1	X	X	X	X	X	X	1	1	0	1
1	X	X	X	X	X	X	X	1	1	1	1

其中，$D7$ 为最高优先级，$D0$ 为最低优先级；$Y2$、$Y1$、$Y0$ 为输出编码；V 为有效位，当输入全为 0 时，$V = 0$，输出无效。

逻辑表达式 根据真值表，可以推导出输出的逻辑表达式：

$$
\begin{aligned}
Y2 &= D4 + D5 + D6 + D7 \\
Y1 &= D2 + D3 + D6 + D7 \\
Y0 &= D1 + D3 + D5 + D7 \\
V &= D0 + D1 + D2 + D3 + D4 + D5 + D6 + D7
\end{aligned}
$$

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-11)：

```verilog
module priority_encoder_8to3 (
    input [7:0] d,
    output reg [2:0] y,
    output reg valid
);
    always @(*) begin
        valid = 1'b0; // 默认输出为无效状态
        y = 3'b000;
        if (d[7]) begin // 检查优先级，从最高位到最低位
            y = 3'b111;
```

[5] 出于排版美观的考量，8-3 编码器的仿真波形（图 6）放在文章的最后

```
            valid = 1'b1;
        end
        else if (d[6]) begin
            y = 3'b110;
            valid = 1'b1;
        end
        else if (d[5]) begin
            y = 3'b101;
            valid = 1'b1;
        end
        else if (d[4]) begin
            y = 3'b100;
            valid = 1'b1;
        end
        else if (d[3]) begin
            y = 3'b011;
            valid = 1'b1;
        end
        else if (d[2]) begin
            y = 3'b010;
            valid = 1'b1;
        end
        else if (d[1]) begin
            y = 3'b001;
            valid = 1'b1;
        end
        else if (d[0]) begin
            y = 3'b000;
            valid = 1'b1;
        end
    end
endmodule
```

Code 11. 8-3 优先编码器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-12)：

```
module tb_priority_encoder_8to3;
    reg [7:0] d;
    wire [2:0] y;
    wire valid;
    priority_encoder_8to3 uut (
        .d(d),
        .y(y),
        .valid(valid)
    );

    integer i;
    initial begin
        for (i = 0; i < 256; i = i + 1) begin
            d = i;
            #10;
        end

        $finish;
    end
    initial begin
        $monitor("At time %t: d = %b -> y = %b, valid = %b", $time, d, y, valid);
    end
endmodule
```

Code 12. 8-3 优先编码器 testbench 代码.

在创建的 Testbench 文件中，我编写了测试激励代码，每隔 10ns 更改一次输入，遍历不同的输入组合，以确保覆盖每一种优先级输入。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation，选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 7 所示。[6]

4.3. 3-8 译码器

根据实验指导的要求，设计一个 3-8 译码器。3-8 译码器是将 3 位二进制输入转换为 8 个输出信号，其中只有一个输出为高电平，其余输出为低电平。

真值表

A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

电路图 根据上述真值表，设计的 3-8 译码器电路使用逻辑门实现输入到输出的映射。输入信号为 A_2、A_1、A_0，输出信号为 Y_0 到 Y_7。

Verilog 代码 在 Vivado 平台设计如下的 Verilog 代码 (Code-13)：

```
module decoder_3to8 (
    input [2:0] a,
    output reg [7:0] y
);
    always @(*) begin
        // 根据输入信号 a 选择输出信号 y
        case (a)
            3'b000: y = 8'b00000001;
            3'b001: y = 8'b00000010;
            3'b010: y = 8'b00000100;
            3'b011: y = 8'b00001000;
            3'b100: y = 8'b00010000;
            3'b101: y = 8'b00100000;
            3'b110: y = 8'b01000000;
            3'b111: y = 8'b10000000;
            default: y = 8'b00000000;
        endcase
    end
endmodule
```

Code 13. 3-8 译码器设计代码.

仿真 Testbench 代码 为了验证电路设计的正确性，我编写了用于 Behavioral Simulation 的 Testbench。(Code-14)：

```
module tb_decoder_3to8;
    reg [2:0] a;
    wire [7:0] y;
    decoder_3to8 uut (
        .a(a),
```

[6] 出于排版美观的考量，8-3 优先编码器的仿真波形（图 7）放在文章的最后

```
        .y(y)
    );

    integer i;
    initial begin

    // 使用for循环遍历所有输入组合. 每隔 10ns 改变一次输入
    for (i = 0; i < 8; i = i + 1) begin
        a = i;
        #10;
    end

        $finish;
    end

    // 监控信号变化
    initial begin
        $monitor("At time %t: a = %b -> y = %b", $time, a, y);
    end

endmodule
```

Code 14. 3-8 译码器 testbench 代码.

在创建的 TestBench 文件中，我编写了测试激励代码，代码每隔 10ns 更改一次输入，遍历所有可能的输入组合（共 2^3 个不同的三元组），以确保覆盖每一种输入情况。

Behavioral Simulation 输出波形 在 Vivado 的"Flow Navigator" 窗口中，找到 Simulation 部分，选择 Run Simulation，选择 Run Behavioral Simulation（行为仿真）。Vivado 编译设计源代码和 Testbench，生成仿真文件，波形如图 8 所示。[7]

5. 调试和心得体会

在本次实验中，我全面掌握了 Vivado 软件的使用方法，并深入学习了 Verilog 硬件描述语言的编程技巧。在设计和实现各种组合逻辑电路的过程中，经历了多次调试和优化，加深了对数字电路设计的理解。

在多输入门电路的设计中，初始的 Verilog 代码由于对逻辑表达式理解不透彻，导致仿真结果与预期不符。通过重新审视逻辑表达式，手动推导真值表，并对代码中的逻辑运算符进行检查，最终纠正了错误，确保了电路的功能正确。

在复用器的设计中，特别是八路复用器，由于输入信号和选择信号较多，测试用例的数量呈指数增长。为了提高测试效率，我编写了自动生成测试激励的脚本，使得仿真可以覆盖更多的输入组合。同时，在仿真波形分析中，通过关注关键信号的变化，验证了复用器的选择功能是否正确，实现了对设计的全面验证。

在优先编码器的设计中，需要考虑输入信号的优先级，这增加了设计的复杂性。通过使用条件语句和优先级判断，我成功地实现了优先编码器的功能。在调试过程中，发现当多个输入信号同时为高电平时，输出不符合预期。经过分析，发现是条件判断的顺序导致了优先级的错误。通过调整代码中条件判断的顺序，确保了高优先级信号被正确编码。

在译码器的设计中，我尝试了多种不同的描述方式，包括行为级、数据流级和结构化描述。通过对比不同描述方式的代码和仿真结果，体会到了不同设计方法的优缺点。特别是在结构化描述中，通过模块的层次化设计，提高了代码的可读性和可维护性。

在整个实验过程中，充分利用了 Vivado 的仿真和调试功能。在遇到问题时，学会了通过波形分析定位错误，利用断点和信号强制等手段进行调试。本次实验让我对组合逻辑电路的设计和仿真有了更深刻的理解。熟悉了 Vivado 的仿真和调试流程，掌握了使用 Verilog 进行硬件描述的多种方法。

6. 附件

由于排版原因，本次实验报告将所有波形图文件放置在最后一页，以兼具美观和可读性。

[7] 出于排版美观的考量，3-8 译码器的仿真波形（图 8）放在文章的最后

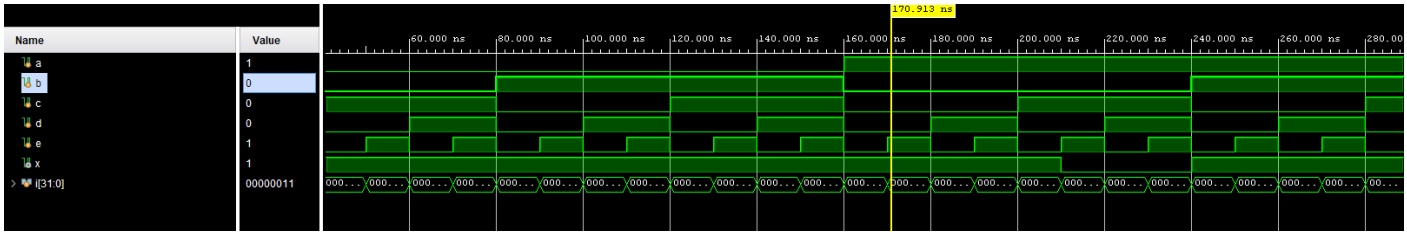

Figure 1. 多输入门电路仿真波形

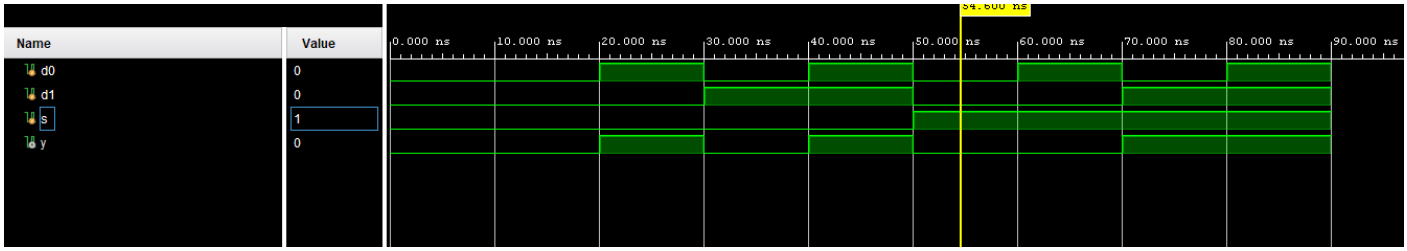

Figure 2. 二路复用器仿真波形

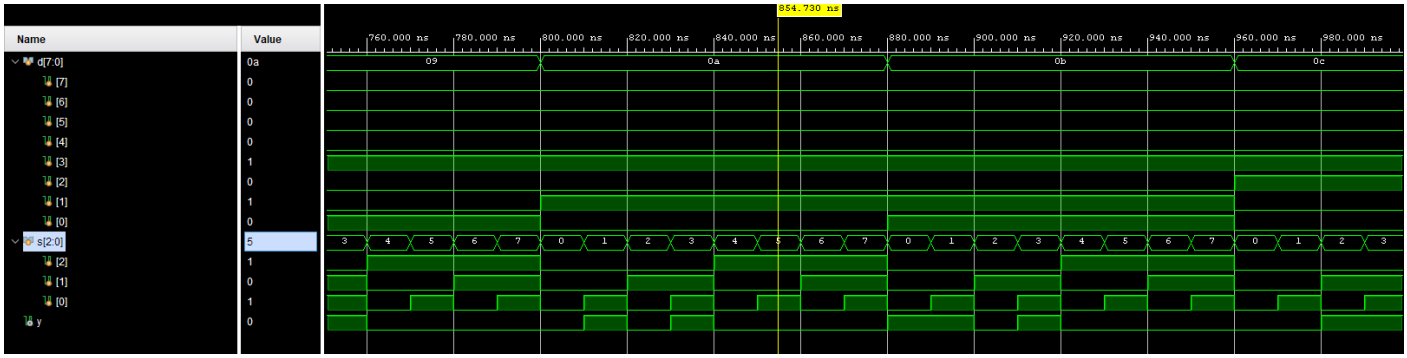

Figure 3. 八路复用器仿真波形

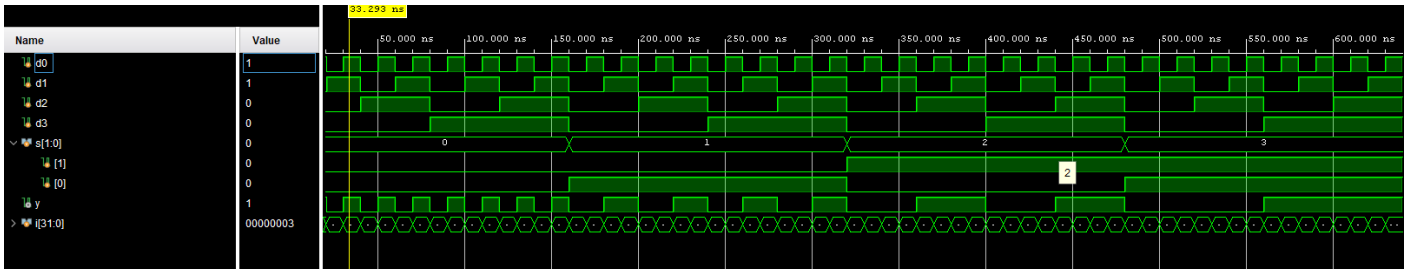

Figure 4. 基于三态缓冲器的 4 路复用器仿真波形

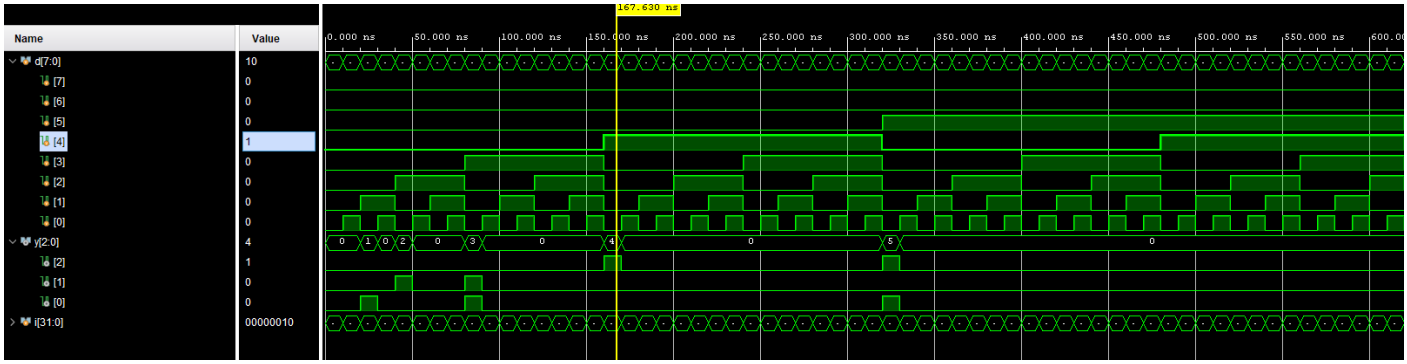

Figure 5. 8-3 编码器仿真波形

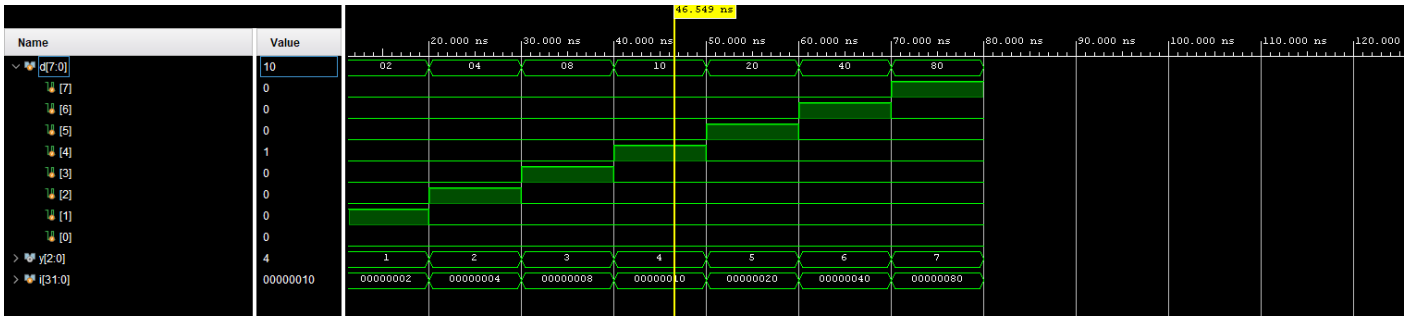

Figure 6. 8-3 编码器仿真波形

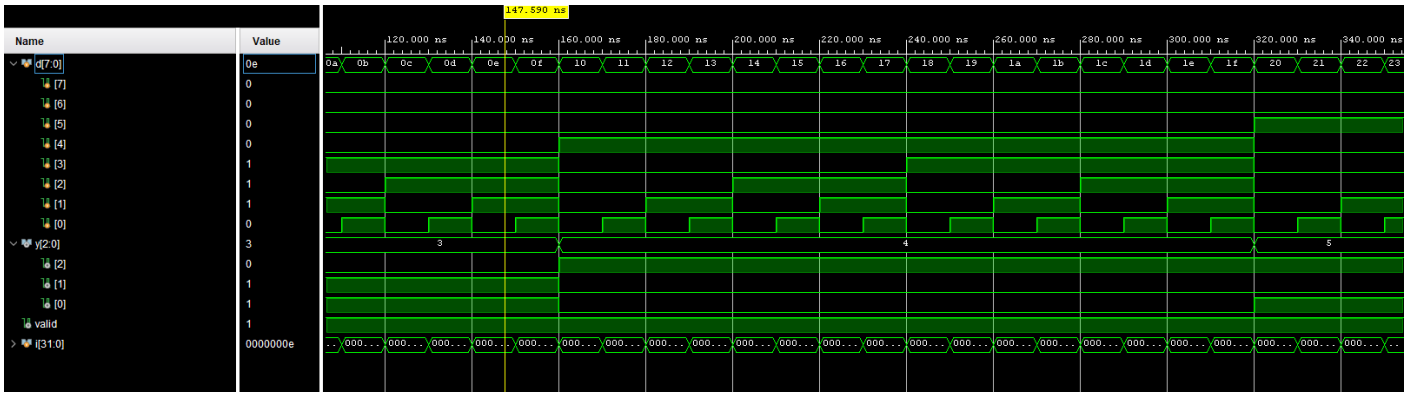

Figure 7. 8-3 优先编码器仿真波形

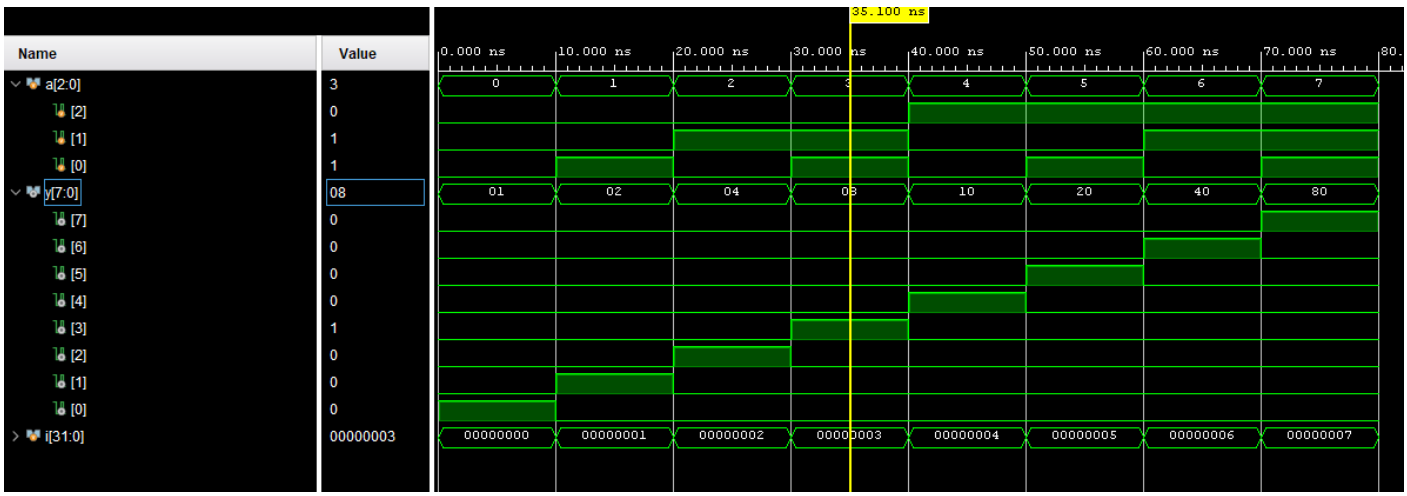

Figure 8. 3-8 译码器仿真波形