同济大学计算机系

数字逻辑课程实验报告

学　　号 2353900

姓　　名 汪嘉晨

专　　业 计算机科学与技术（精英班）

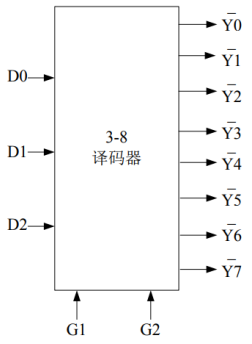
授课老师 郭玉臣

一、实验内容

在本次实验中，我们将使用 VerilogHDL 语言实现 3-8 译码器、8-3 编码器以及七段数码管的设计和仿真。

1) 3-8 译码器

实现译码功能的组合逻辑电路称为译码器，它的输入是一组二进制代码，输出是一组高低电平信号。所要建模的 3-8 译码器及真值表如图 6.3.1 所示，它有三个编码输入、八个输出和二个使能输入端（G1，G2）。作为译码器使用时，使能端必须满足 G1=1，G2=0。



G_1	G_2	D_2	D_1	D_0	$\bar{Y}_0$	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\bar{Y}_6$	$\bar{Y}_7$
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

图 6.3.1 3-8 译码器及其真值表

2) 七段数码管译码驱动器

图 6.3.2 为所要建模的七段数码管译码驱动原理图，它由译码驱动器和荧光数码管组成。荧光数码管是分段式半导体显示器件，7 个发光二极管组成 7 个发光段，发光二极管可以将电能转换成光能，从而发出清晰悦目的光线。本实验采用的是共阳极电路，故译码器的输出 a~g 分别加到 7 个阴极上。只有在阴极上呈低电平的二极管导通发光，显示 0~9 中相应的十进制数字。表 6.3.1 所示为七段数码管译码驱动器逻辑功能真值表，四个输入和七个输出以及对应显示的字符。

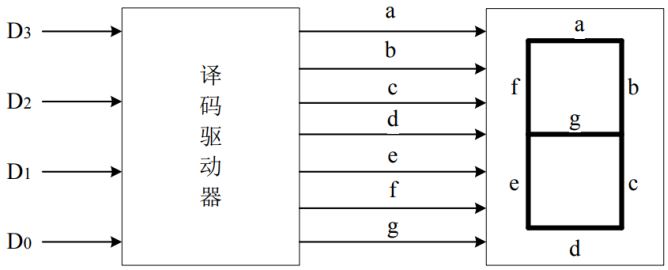


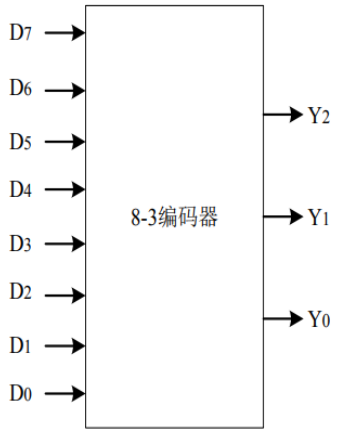
图 6.3.2 七段数码管译码驱动原理图

表 6.3.1 七段数码管译码驱动器逻辑功能表

输入				输出							显示
D_3	D_2	D_1	D_0	g	f	e	d	c	b	a	字符
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1	1
0	0	1	0	0	1	0	0	1	0	0	2
0	0	1	1	0	1	1	0	0	0	0	3
0	1	0	0	0	0	1	1	0	0	1	4
0	1	0	1	0	0	1	0	0	1	0	5
0	1	1	0	0	0	0	0	0	1	0	6
0	1	1	1	1	1	1	1	0	0	0	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	1	0	0	0	0	9

3) 普通 8-3 编码器

用来完成编码工作的电路称为编码器。它可以实现对一组输入信号的二进制编码。图 6.3.3 为所要建模的普通 8-3 编码器及其真值表。它有 8 个输入以及 3 个输出，真值表中每行只有一个输入电平有效（高电平为 1，低电平为 0）。

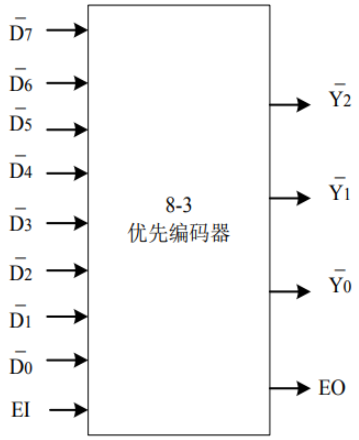


D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0

图 6.3.3 普通 8-3 编码器及其真值表

4) 具有优先级的 8-3 编码器

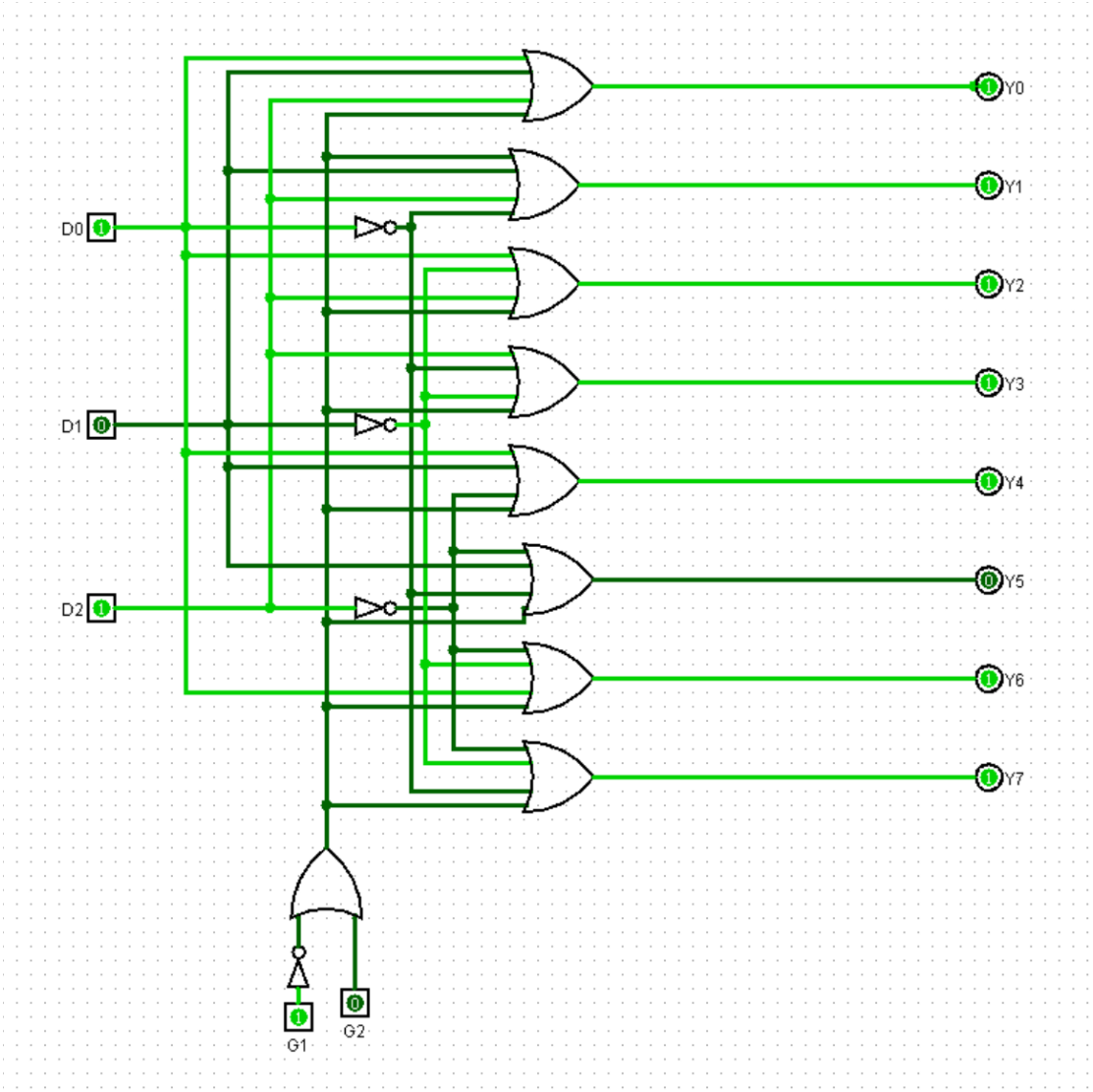
普通编码器对输入线是有限制的，即在任意一时刻所有输入线中只允许一个输入线信号有效，否则编码器将发生混乱。为解决这一问题可以采用具有优先级的编码器。图 6.3.4 为所要建模的具有优先级的 8-3 编码器及其真值表，它有八个输入端、三个输出端、一个选通输入端 EI 以及一个扩展输出端 EO。从真值表可以看出，输入输出的有效信号是低电平，在输入中，角标越大，优先级越高。图 6.3.4 具有优先级的 8-3 编码器及其真值表



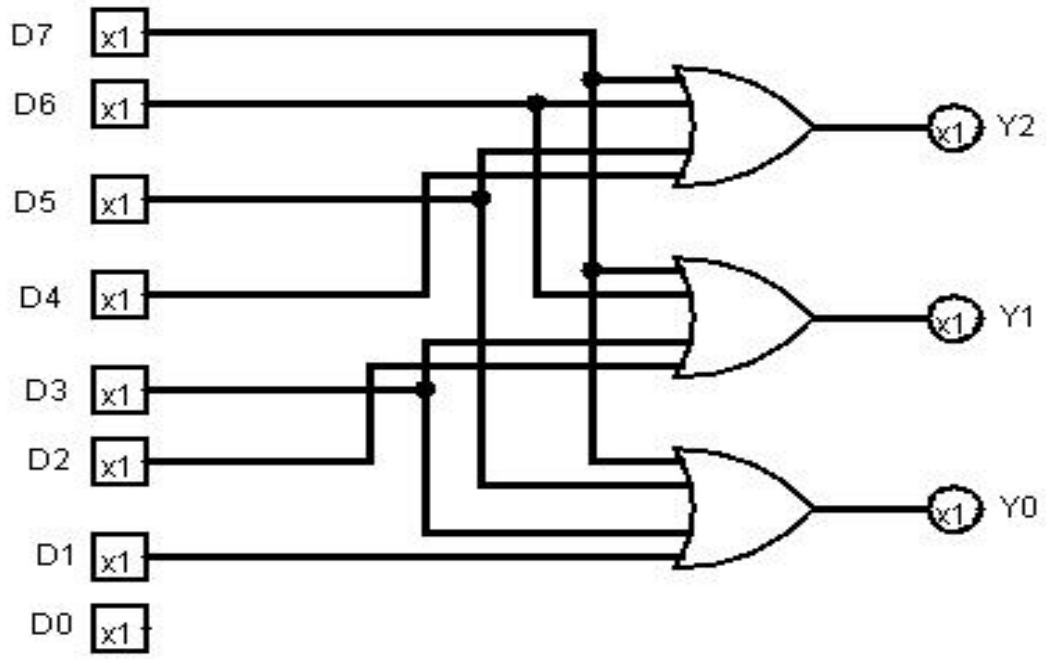
$\overline{D_0}$	$\overline{D_1}$	$\overline{D_2}$	$\overline{D_3}$	$\overline{D_4}$	$\overline{D_5}$	$\overline{D_6}$	$\overline{D_7}$	$\overline{Y_2}\overline{Y_1}\overline{Y_0}$	EI	EO
×	×	×	×	×	×	×	×	1 1 1	1	0
1	1	1	1	1	1	1	1	1 1 1	0	0
×	×	×	×	×	×	×	0	0 0 0	0	1
×	×	×	×	×	×	0	1	0 0 1	0	1
×	×	×	×	×	0	1	1	0 1 0	0	1
×	×	×	×	0	1	1	1	0 1 1	0	1
×	×	×	0	1	1	1	1	1 0 0	0	1
×	×	0	1	1	1	1	1	1 0 1	0	1
×	0	1	1	1	1	1	1	1 1 0	0	1
0	1	1	1	1	1	1	1	1 1 1	0	1

二、硬件逻辑图

3-8 译码器的硬件逻辑图如下：



普通 8-3 编码器硬件逻辑图如下：



三、模块建模

1. 3-8 译码器

```
module decoder(
    input [2:0] iData, //  三位输入  D2,D1,D0
    input [1:0] iEna,    //  使能信号  G1,G2
    output reg [7:0] oData //  八位译码输出  Y7? ~ Y0?,  低电平有效
);
always @(*) begin
    if (iEna[1] == 1 && iEna[0] == 0) begin
        case (iData)
            3'b000: oData = 8'b11111110;
            3'b001: oData = 8'b11111101;
            3'b010: oData = 8'b11111011;
            3'b011: oData = 8'b11110111;
            3'b100: oData = 8'b11101111;
            3'b101: oData = 8'b11011111;
            3'b110: oData = 8'b10111111;
            3'b111: oData = 8'b01111111;
            default: oData = 8'b11111111; //  默认高电平
        endcase
    end else begin
        oData = 8'b11111111; //  默认高电平
    end
end
endmodule
```

2. 七段数码管译码驱动器

```
module display7(
    input [3:0] iData, //  四位输入  D3~D0
    output reg [6:0] oData //  七位译码输出  g~a
);
always @(*) begin
    case (iData)
        4'b0000: oData = 7'b1000000; //  显示数字  0
        4'b0001: oData = 7'b1111001; //  显示数字  1
        4'b0010: oData = 7'b0100100; //  显示数字  2
        4'b0011: oData = 7'b0110000; //  显示数字  3
        4'b0100: oData = 7'b0011001; //  显示数字  4
        4'b0101: oData = 7'b0010010; //  显示数字  5
        4'b0110: oData = 7'b0000010; //  显示数字  6
        4'b0111: oData = 7'b1111000; //  显示数字  7
        4'b1000: oData = 7'b0000000; //  显示数字  8
        4'b1001: oData = 7'b0010000; //  显示数字  9
        default: oData = 7'b1111111; //  默认高电平
```

```
        endcase
    end
    endmodule
```

3. 普通 8-3 编码器

```
module encoder83(
    input [7:0] iData, // 八位输入
    output reg [2:0] oData // 三位输出
);
always @(*) begin
    case (iData)
        8'b00000001: oData = 3'b000; // 输入 00000001，输出 000
        8'b00000010: oData = 3'b001; // 输入 00000010，输出 001
        8'b00000100: oData = 3'b010; // 输入 00000100，输出 010
        8'b00001000: oData = 3'b011; // 输入 00001000，输出 011
        8'b00010000: oData = 3'b100; // 输入 00010000，输出 100
        8'b00100000: oData = 3'b101; // 输入 00100000，输出 101
        8'b01000000: oData = 3'b110; // 输入 01000000，输出 110
        8'b10000000: oData = 3'b111; // 输入 10000000，输出 111
        default: oData = 3'bxxx; // 默认输出
    endcase
end
endmodule
```

4. 具有优先级的 8-3 编码器

```
module encoder83_Pri(
    input [7:0] iData,
    input iEI,
    output reg [2:0] oData,
    output reg oEO
    );
    always @(*)
    begin
        case (iEI)
            1'b1:
            begin
                oData = 3'b111;
                oEO = 1'b1;
            end
            1'b0:
            begin
                oEO = 1'b1;
                if (iData[7] == 1'b0)
                    oData = 3'b000;
```

```
                else if (iData[7:6] == 2'b10)
                    oData = 3'b001;
                else if (iData[7:5] == 3'b110)
                    oData = 3'b010;
                else if (iData[7:4] == 4'b1110)
                    oData = 3'b011;
                else if (iData[7:3] == 5'b11110)
                    oData = 3'b100;
                else if (iData[7:2] == 6'b111110)
                    oData = 3'b101;
                else if (iData[7:1] == 7'b1111110)
                    oData = 3'b110;
                else if (iData[7:0] == 8'b11111110)
                    oData = 3'b111;
                else if (iData[7:0] == 8'b11111111)
                begin
                    oData = 3'b111;
                    oEO = 1'b0;
                end
            end
        endcase
    end
endmodule
```

四、测试模块建模

1. 3-8 译码器

```
module decoder_tb;
    // 输入信号
    reg [2:0] iData;
    reg [1:0] iEna;
    // 输出信号
    wire [7:0] oData;
    // 实例化被测试的译码器模块
    decoder uut (
        .iData(iData),
        .iEna(iEna),
        .oData(oData)
    );
    // 初始化和测试向量
    initial begin
        // 初始化信号
        iData = 3'b000;
```

```
        iEna = 2'b00;
        // 监视信号变化
        $monitor("Time=%0t, iData=%b, iEna=%b, oData=%b", $time, iData, iEna, oData);
        // 测试用例
        #10 iEna = 2'b10; iData = 3'b000; // G1=1, G2=0, iData=000
        #10 iData = 3'b001; // iData=001
        #10 iData = 3'b010; // iData=010
        #10 iData = 3'b011; // iData=011
        #10 iData = 3'b100; // iData=100
        #10 iData = 3'b101; // iData=101
        #10 iData = 3'b110; // iData=110
        #10 iData = 3'b111; // iData=111
        // 测试使能信号无效的情况
        #10 iEna = 2'b00; iData = 3'b000; // G1=0, G2=0
        #10 iEna = 2'b01; iData = 3'b000; // G1=0, G2=1
        #10 iEna = 2'b11; iData = 3'b000; // G1=1, G2=1
        // 结束仿真
        #10 $finish;
    end
endmodule
```

2. 七段数码管译码驱动器

```
module tb_display7;
    // 输入信号
    reg [3:0] iData;
    // 输出信号
    wire [6:0] oData;
    // 实例化被测试的七段数码管译码驱动器模块
    display7 uut (
        .iData(iData),
        .oData(oData)
    );
    // 初始化和测试向量
    initial begin
        // 初始化信号
        iData = 4'b0000;
        // 监视信号变化
        $monitor("Time=%0t, iData=%b, oData=%b", $time, iData, oData);
        // 测试用例
        #10 iData = 4'b0000; // 显示数字 0
        #10 iData = 4'b0001; // 显示数字 1
        #10 iData = 4'b0010; // 显示数字 2
        #10 iData = 4'b0011; // 显示数字 3
        #10 iData = 4'b0100; // 显示数字 4
```

```
        #10 iData = 4'b0101; //  显示数字  5
        #10 iData = 4'b0110; //  显示数字  6
        #10 iData = 4'b0111; //  显示数字  7
        #10 iData = 4'b1000; //  显示数字  8
        #10 iData = 4'b1001; //  显示数字  9
        // 结束仿真
        #10 $finish;
    end
endmodule
```

3. 普通 8-3 编码器

```
module tb_encoder83;
    // 输入信号
    reg [7:0] iData;
    // 输出信号
    wire [2:0] oData;
  // 实例化被测试的 8-3 编码器模块
    encoder83 uut (
        .iData(iData),
        .oData(oData)
    );
   // 初始化和测试向量
    initial begin
        // 初始化信号
        iData = 8'b00000000;
        // 监视信号变化
        $monitor("Time=%0t, iData=%b, oData=%b", $time, iData, oData);
        // 测试用例
        #10 iData = 8'b00000001; //  输入 00000001
        #10 iData = 8'b00000010; //  输入 00000010
        #10 iData = 8'b00000100; //  输入 00000100
        #10 iData = 8'b00001000; //  输入 00001000
        #10 iData = 8'b00010000; //  输入 00010000
        #10 iData = 8'b00100000; //  输入 00100000
        #10 iData = 8'b01000000; //  输入 01000000
        #10 iData = 8'b10000000; //  输入 10000000
        // 结束仿真
        #10 $finish;
    end
endmodule
```

4. 具有优先级的 8-3 编码器

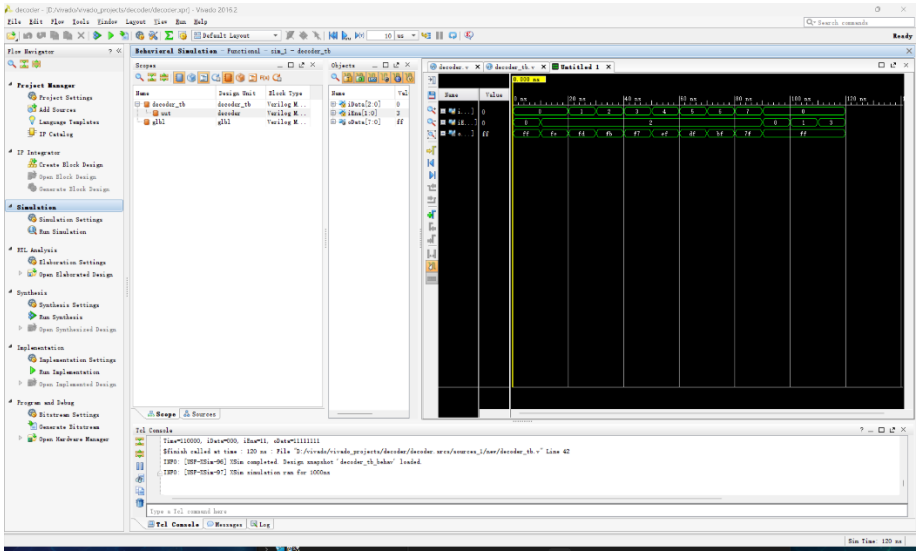
```
module tb_encoder83_Pri;
    // 输入信号
```

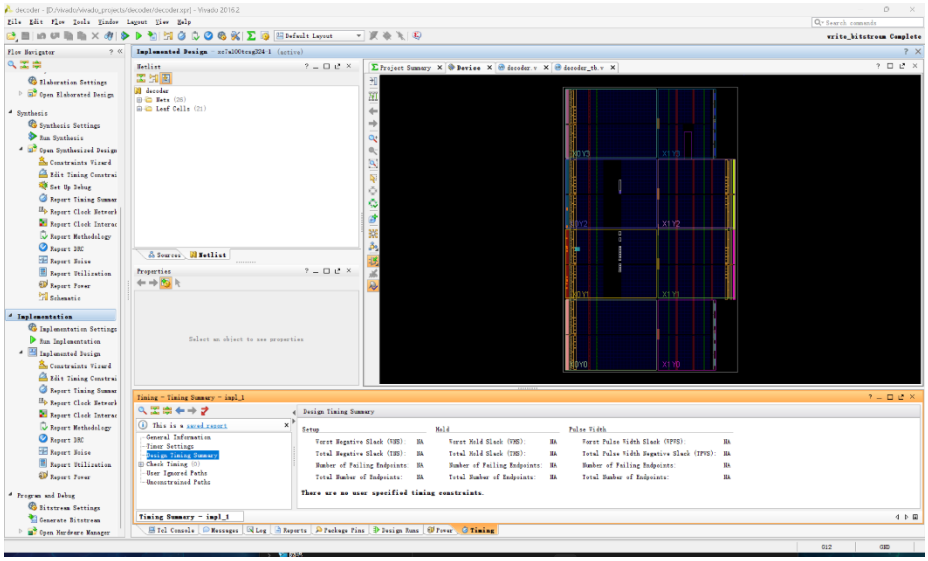
```
    reg [7:0] iData;
    reg iEI;
    // 输出信号
    wire [2:0] oData;
    wire oEO;
    // 实例化被测试模块
    encoder83_Pri uut (
        .iData(iData),
        .iEI(iEI),
        .oData(oData),
        .oEO(oEO)
    );
    initial begin
        // 初始化输入信号
        iData = 8'b00000000;
        iEI = 1'b0;
        // 监视信号变化
        $monitor("Time = %0t, iData = %b, iEI = %b, oData = %b, oEO = %b", $time, iData, iEI, oData, oEO);
        // 测试用例
        #10 iData = 8'b00000001; iEI = 1'b0;
        #10 iData = 8'b00000010; iEI = 1'b0;
        #10 iData = 8'b00000100; iEI = 1'b0;
        #10 iData = 8'b00001000; iEI = 1'b0;
        #10 iData = 8'b00010000; iEI = 1'b0;
        #10 iData = 8'b00100000; iEI = 1'b0;
        #10 iData = 8'b01000000; iEI = 1'b0;
        #10 iData = 8'b10000000; iEI = 1'b0;
        #10 iData = 8'b11111111; iEI = 1'b0;
        #10 iEI = 1'b1; // 测试 iEI 为 1 的情况
        #10 $finish; // 结束仿真
    end

endmodule
```

五、实验结果

1. 3-8 译码器

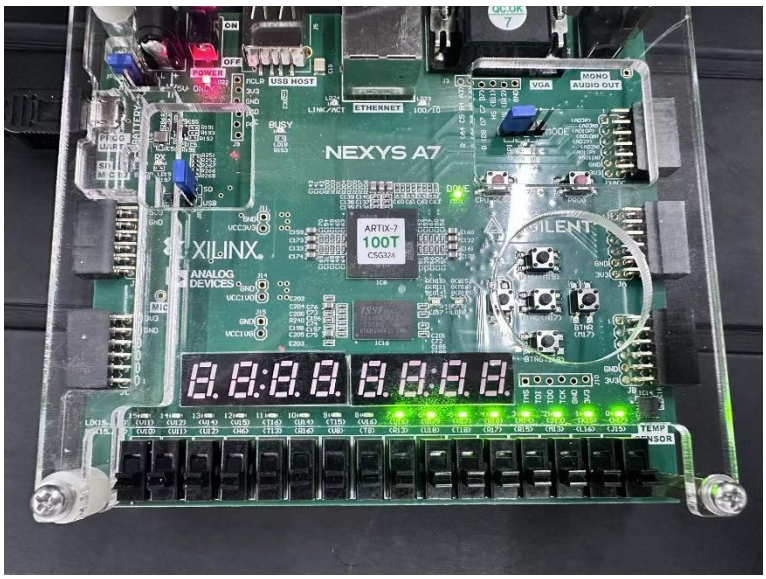
   ModelSim 仿真测试该模块如下：

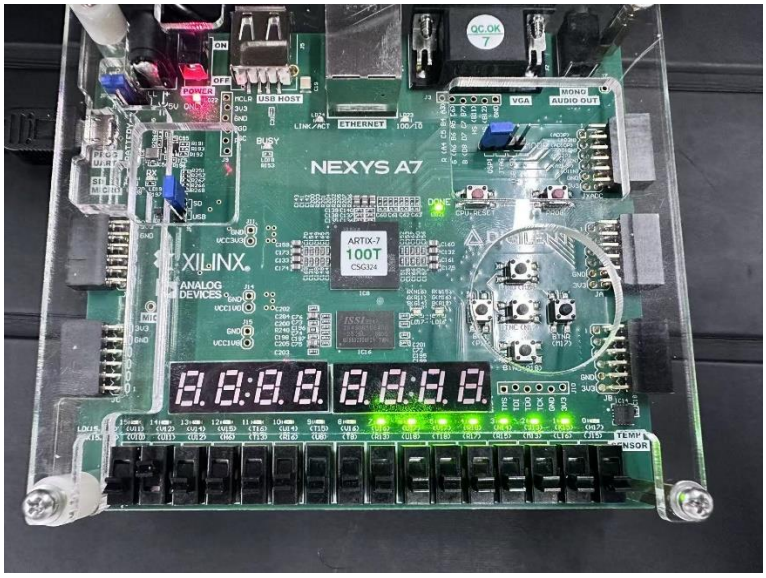


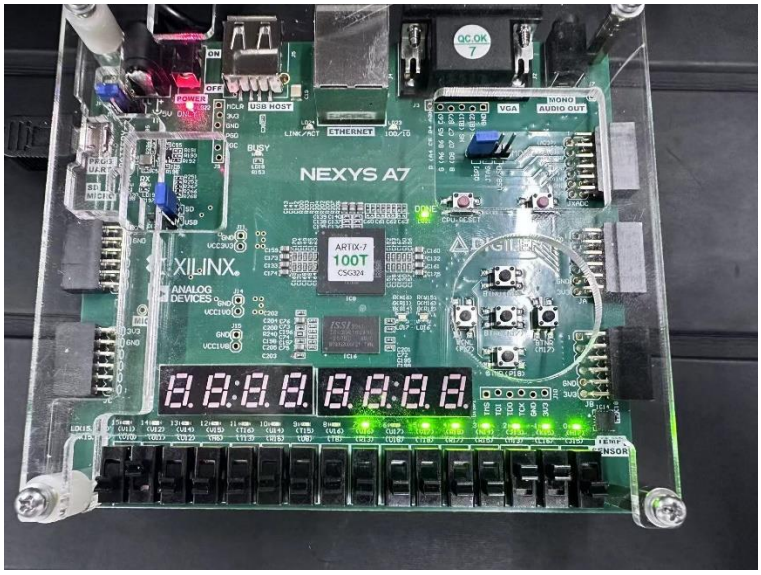


综合下板，结果与真值表相同：

通过控制 3 个开关，实现 8 个 LED 灯的开合

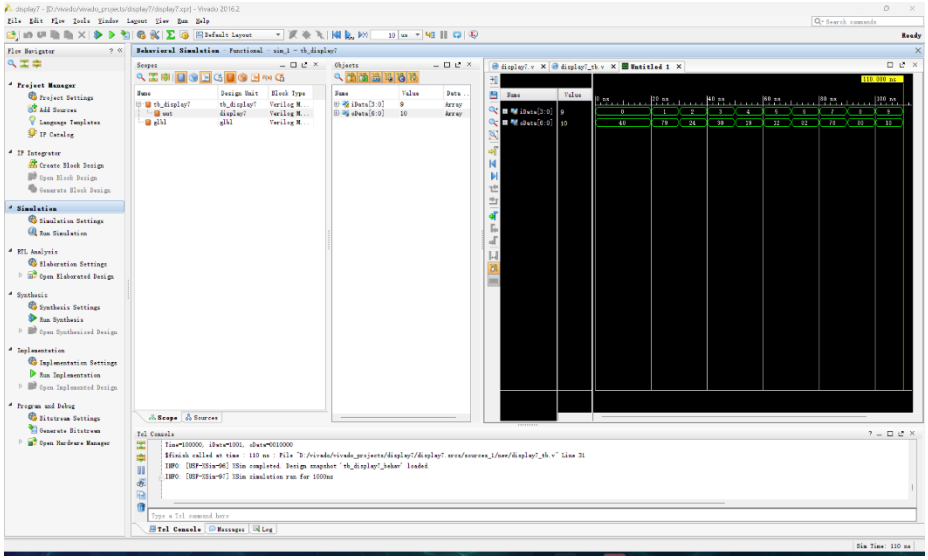


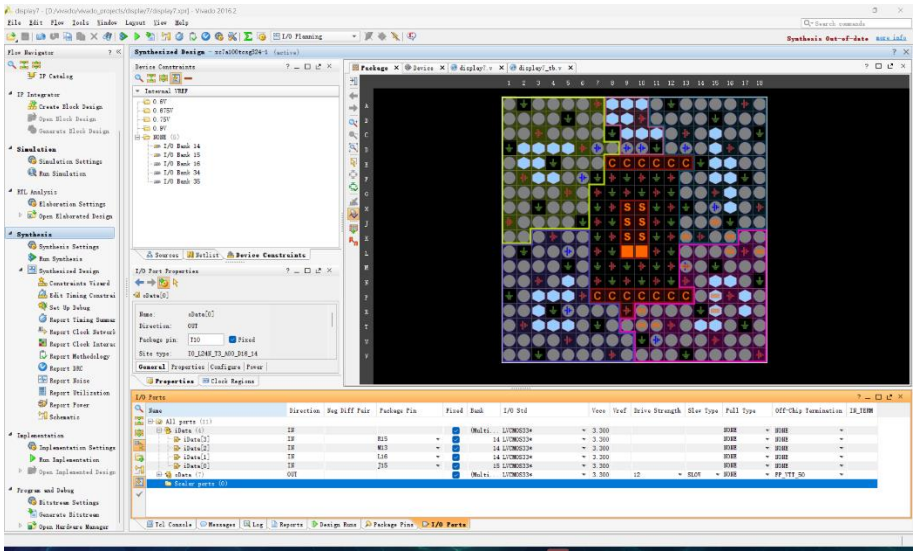


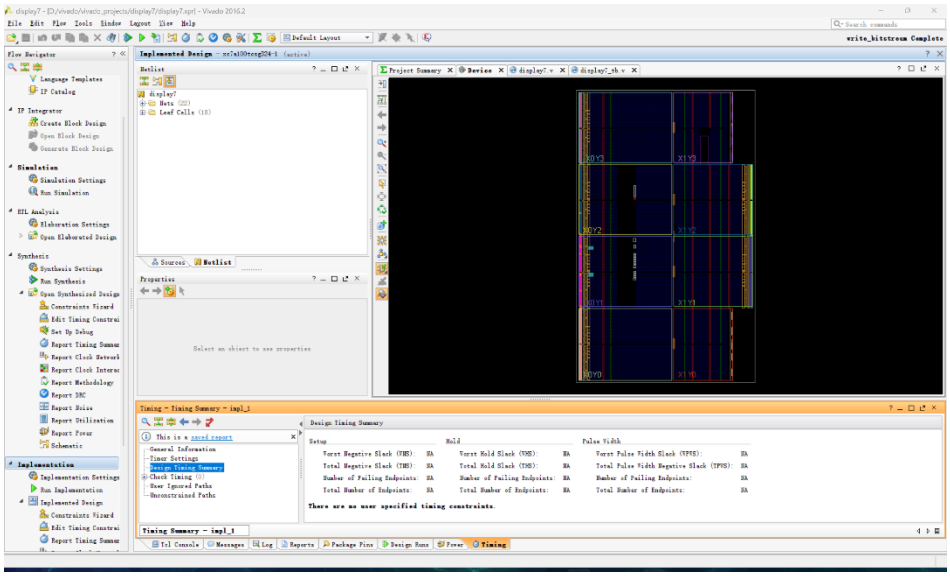


2. 七段数码管译码驱动器

ModelSim 仿真测试该模块如下：

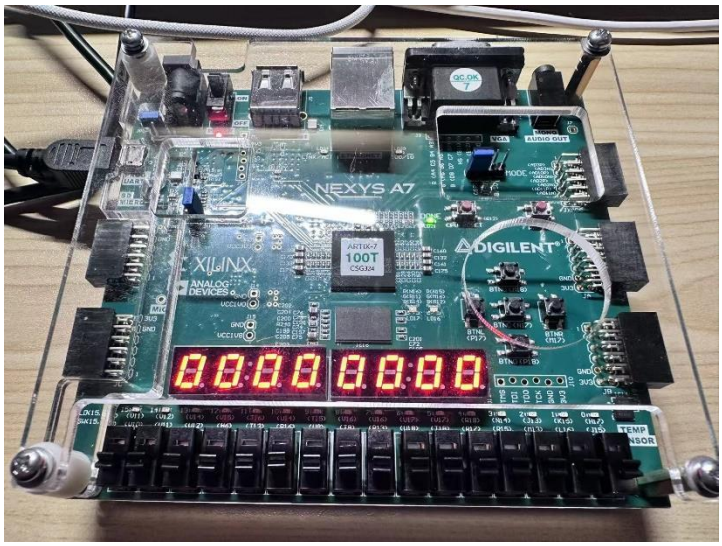


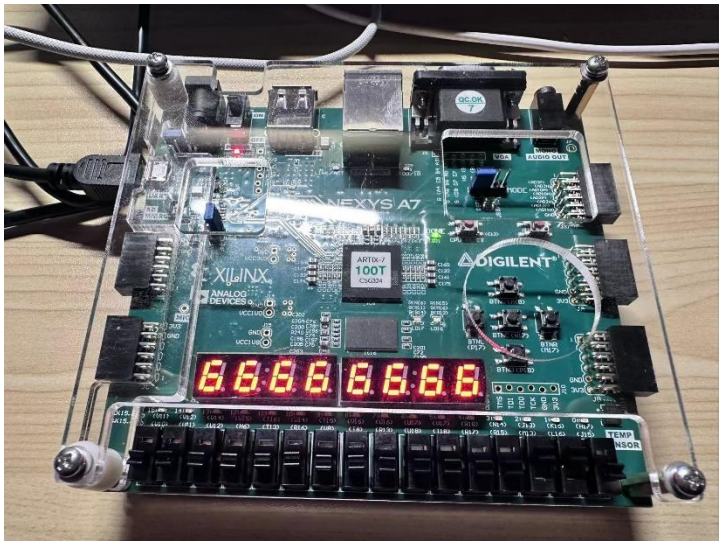


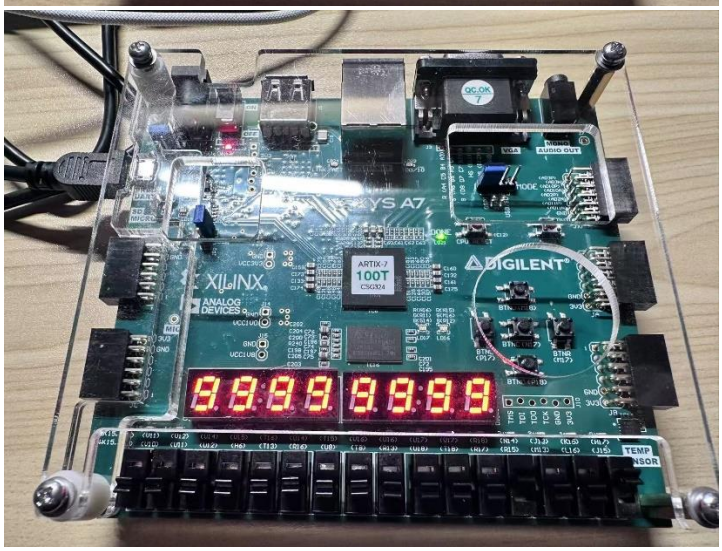


综合下板，结果与真值表相同：

控制从右到左 0-9 这几个开关，显示不同的数字

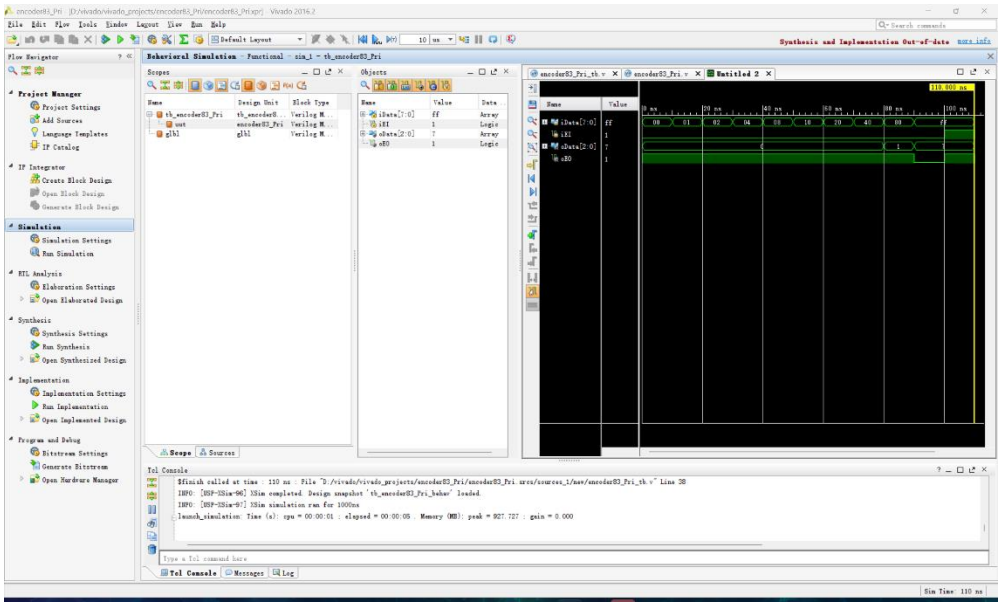


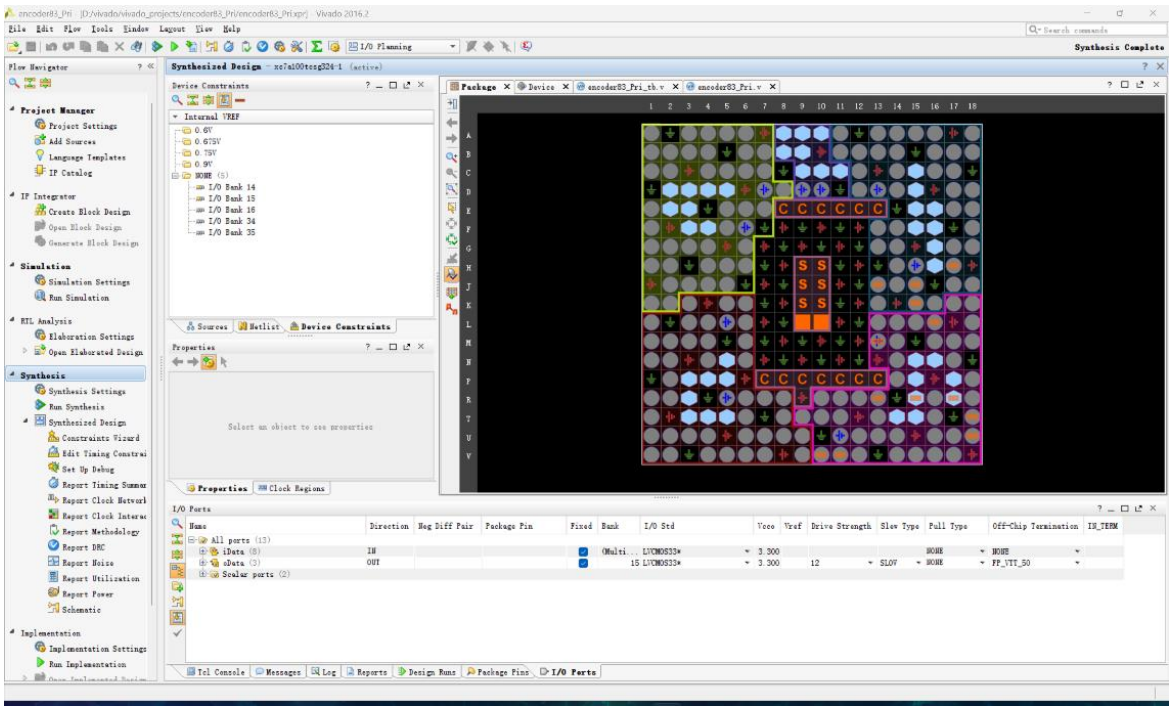


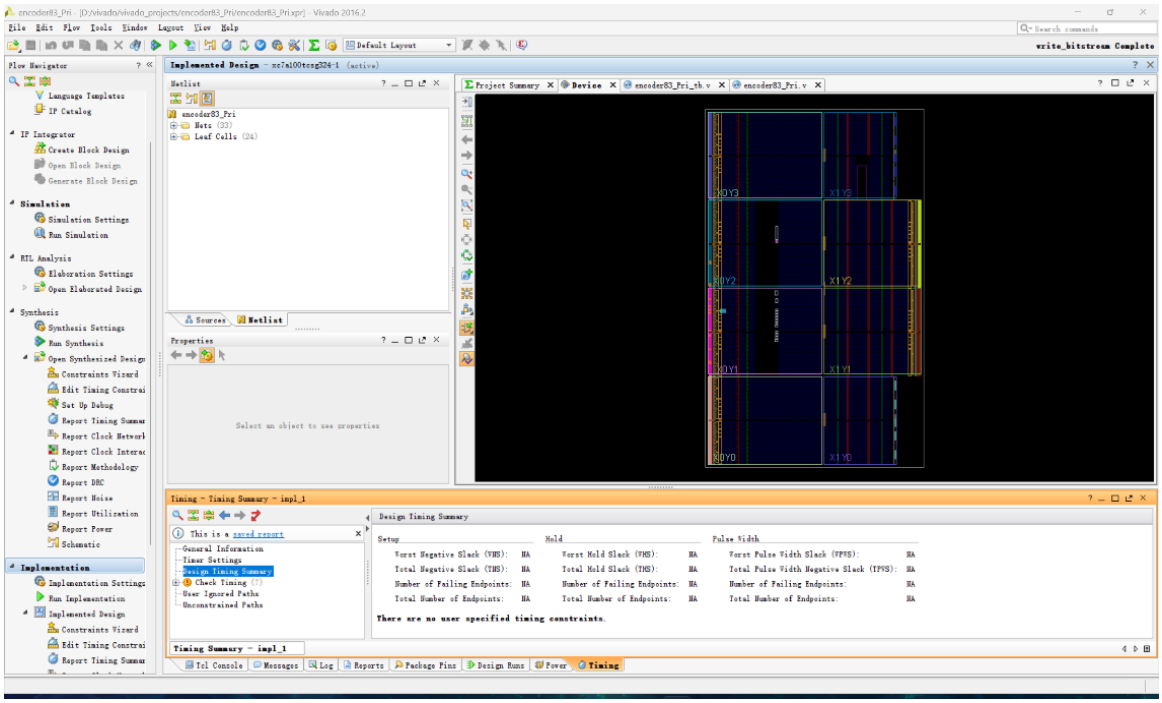


3. 普通 8-3 编码器

ModelSim 仿真测试该模块如下：

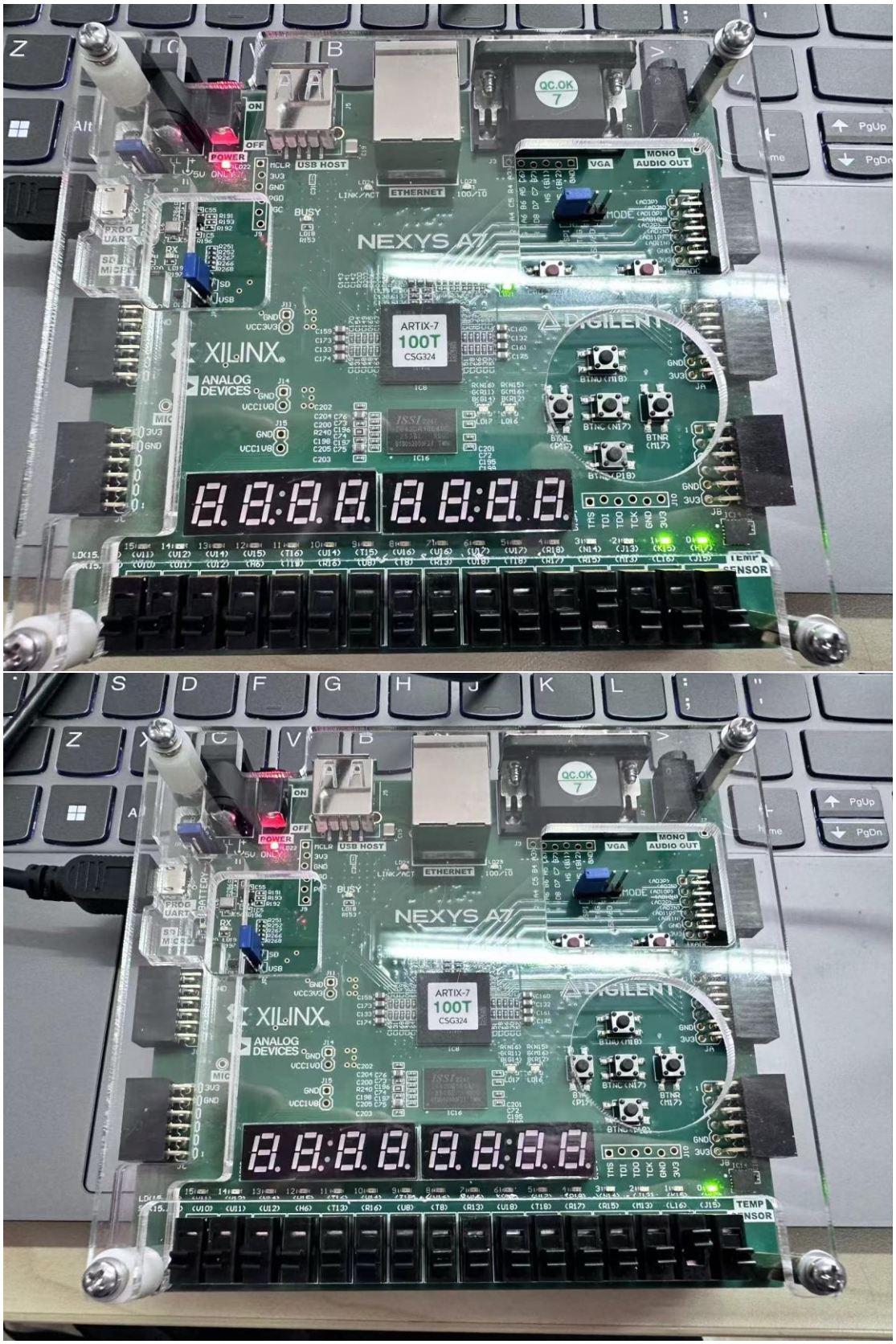






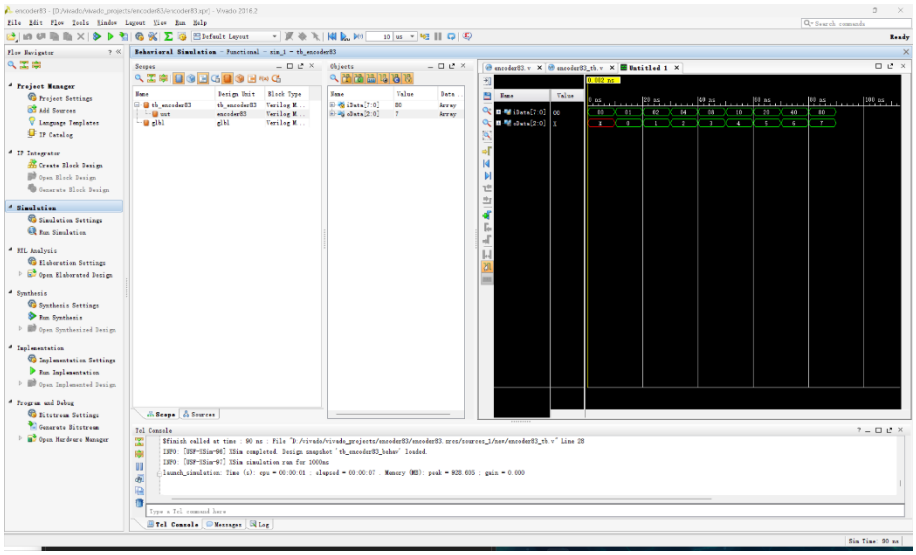
综合下板，结果与真值表相同：

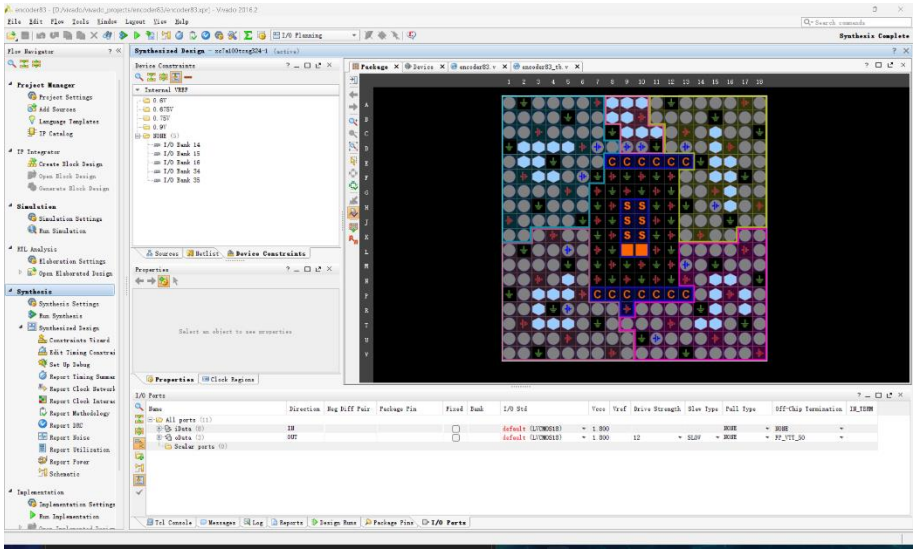
8 个输入以及 3 个输出，真值表中每行只有一个输入电平有效

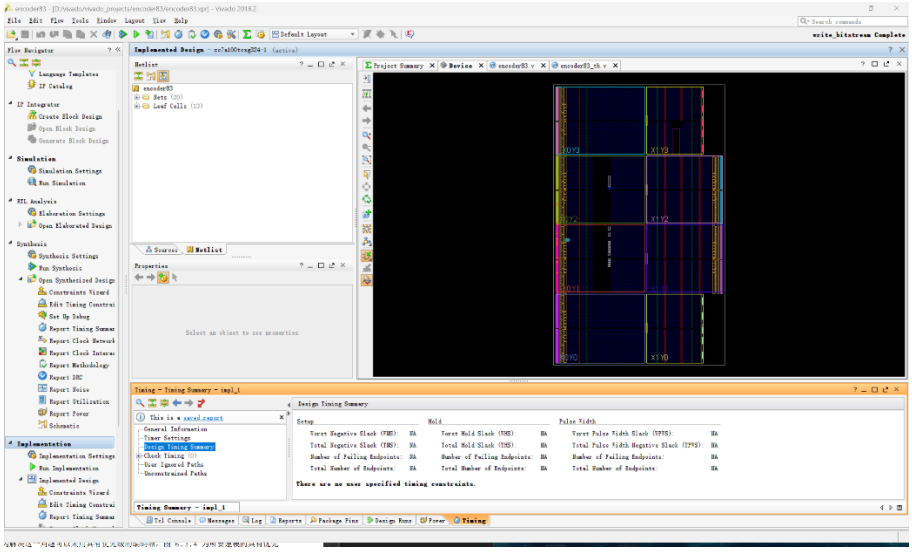


4. 具有优先级的 8-3 编码器

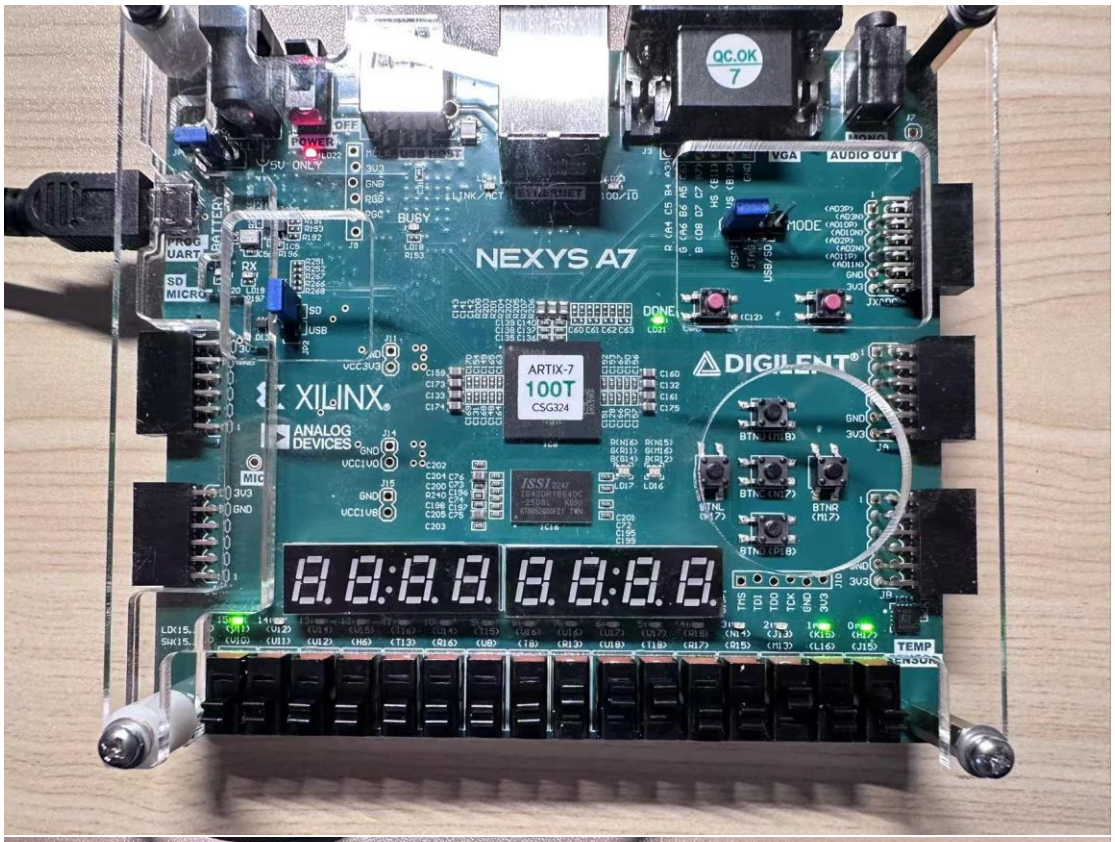
ModelSim 仿真测试该模块如下：

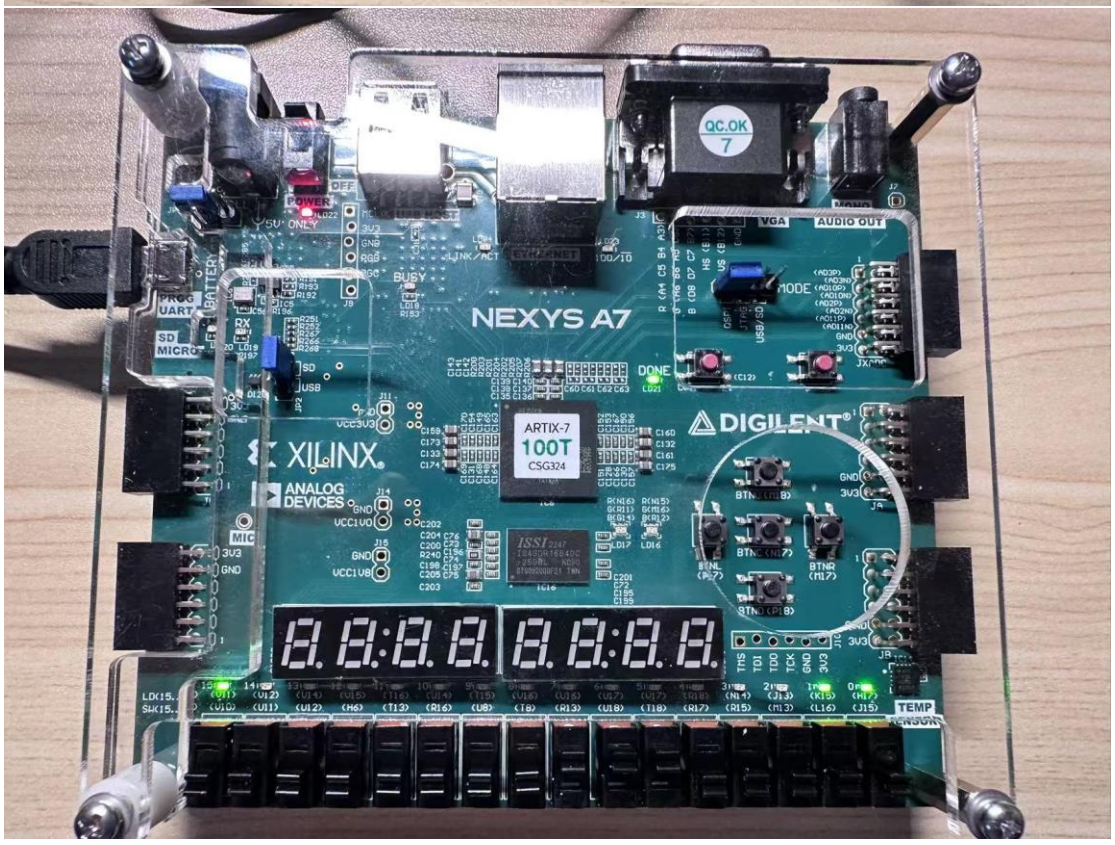


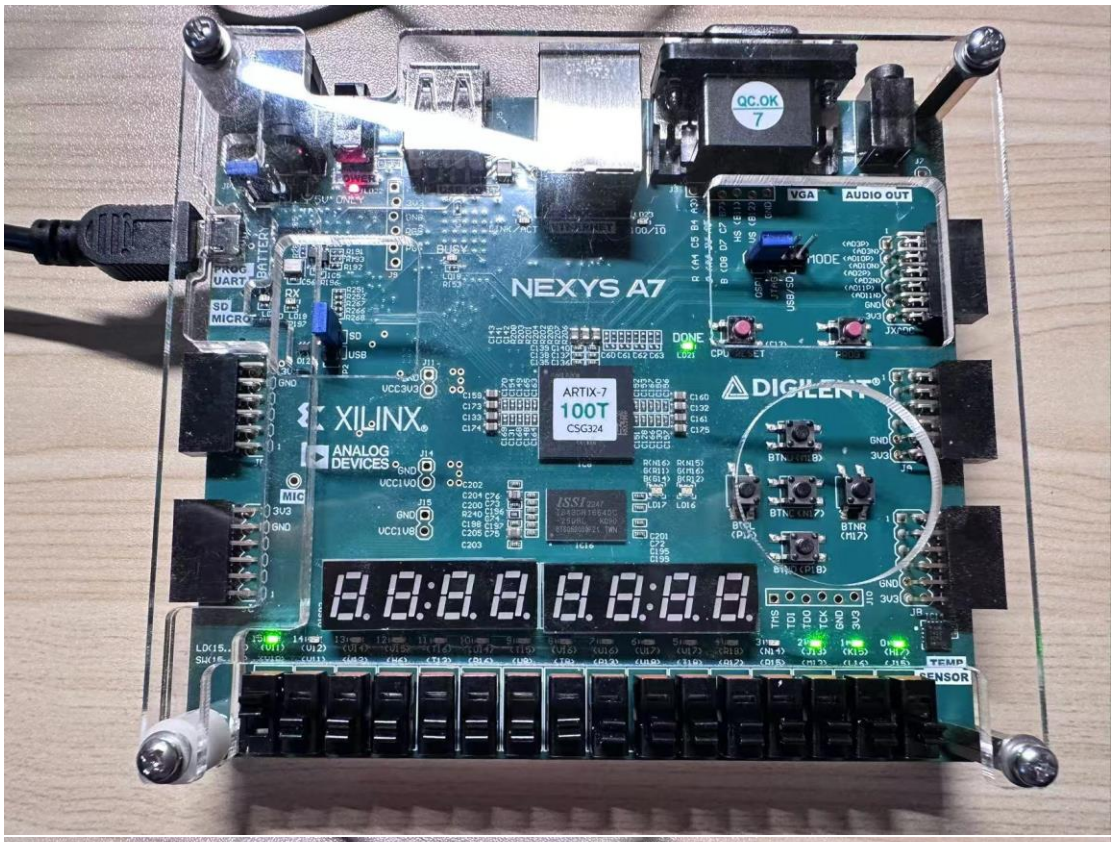


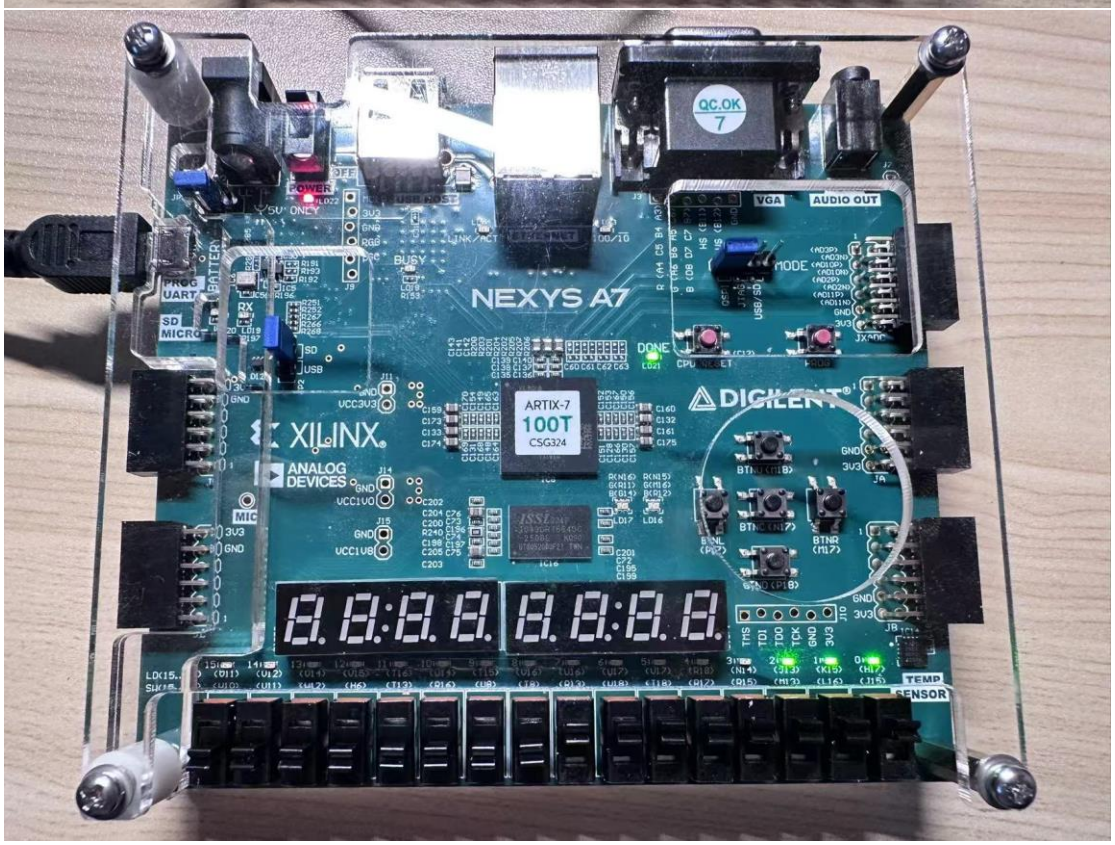


综合下板，结果与真值表相同：


QC.OK
7
NEXYS A7
ARTIX-7
100T
CSG324
XILINX
DIGILENT
ANALOG DEVICES
VGA
AUDIO OUT
MODE
POWER
PROG UART
SD MICRO
BTNU
BTNL
BTNR
BTND
TEMP


QC.OK
7
NEXYS A7
ARTIX-7
100T
CSG324
XILINX
DIGILENT
ANALOG DEVICES
VGA
AUDIO OUT
MODE
POWER
PROG UART
SD MICRO
BTNU
BTNL
BTNR
BTND
TEMP


QC.OK
7
NEXYS A7
VGA
AUDIO OUT
XILINX
ANALOG DEVICES
DIGILENT
ARTIX-7
100T
CSG324
PROG UART
SD MICRO
MIC
TEMP SENSOR


QC.OK
7
NEXYS A7
VGA
AUDIO OUT
XILINX
ANALOG DEVICES
DIGILENT
ARTIX-7
100T
CSG324
PROG UART
SD MICRO
MIC
TEMP SENSOR