实验二

译码器和编码器的设计

实验报告　　日期：2020 年 9 月 18 日

姓名： **朱嘉琦**

学号： **191220185**

班级： **数电实验一班**

邮箱： **1477194584@qq.com**

实验二报告——译码器和编码器的设计

191220185 朱嘉琦

一、实验目的

学习绍常用的译码器和编码器的设计方法以及七段数码管的使用，学习Verilog语言中for循环的使用，熟悉verilog语言中的case语句、casez语句和casex语句的使用等，查找8-3优先编码器相关原理和实现方法，完成一个8-3优先编码器，完成8-3编码器的设计、功能仿真和硬件实现。

二、实验原理

译码器

译码器是将某一输入信息转换为某一特定输出的逻辑电路，通常是多路输入/ 输出电路，它将n 位的输入编码转换为m 位的编码输出，一般情况下n < m，输入编码和输出编码之间存在着一一对应的映射关系，每个输入编码产生唯一的不同于 其他的输出编码。

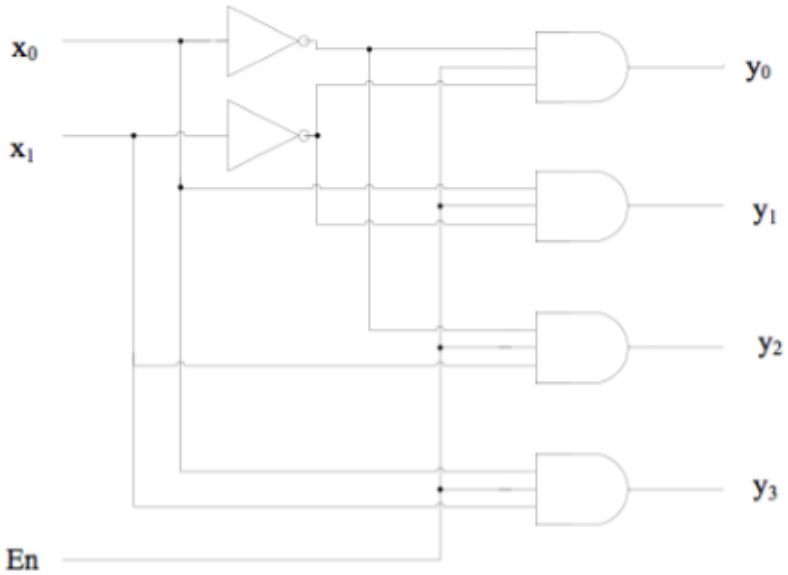


图 2-3: -4 译码器逻辑电路

编码器

常用的二进制编码器把来自于2n条输入线的信息编码转换成n 位二进制码。二进制编码器每次输入的2n位信号中只能有一位为1，其余均为0（即独热码），编码器的输出端为一个二进制数，用来指示对应的哪一个位输入为1。

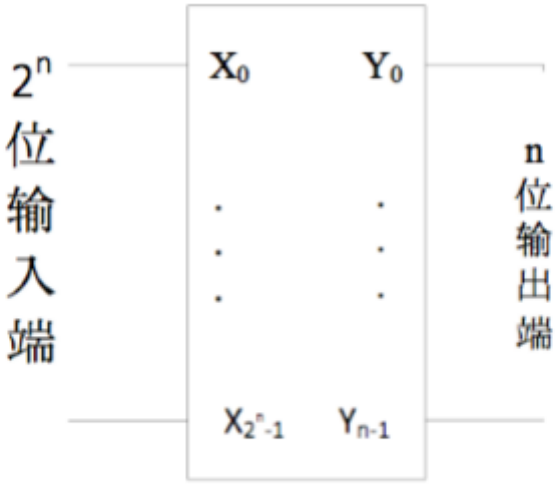


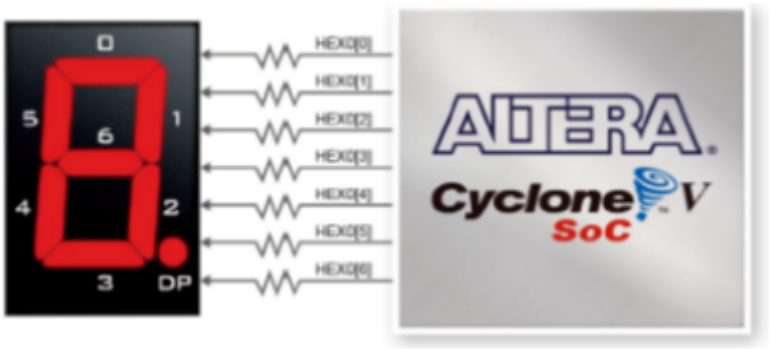
图 2-5: 2^n 位输入，n 位输出的编码器

优先编码器

优先编码器允许同时在几个输入端有输入信号，即输入不止一个“1”，编码器按输入信号排定的优先顺序，只对同时输入的几个信号中优先权最高的一个进行编码。

七段数码管

七段LED数码管是一种常用的显示元件，常应用于手表、计算器等仪器中，用 于显示数值。数码管分为共阴极和共阳极两种类型，共阴极就是将七个LED的阴 极连在一起，让其接低电平。这样给任何一个LED的另一端高电平，它便能点亮。而共阳极就是将七个LED的阳极连在一起，让其接高电平。这样，给任何一个LED的另一端低电平它就能点亮。 DE10-Standard 开发板上的数码管就是七段共阳极的。每个数码管的七段LED 的 一端连接在一个共同的阳极上，另一端和开发板上Cyclone V SoC FPGA的某一个一个引脚连接在一起，如果从这个引脚输出一个逻辑0，则此段数码管被点亮。



三、实验环境/器材

实验环境是Quarters 17.1 Lite，实验器材是DE10 开发板

四、程序代码或流程图

1.2-4译码器

```
module exp2_1(x,en,y);
   input [1:0] x;
   input en;
   output reg [3:0] y;
   always @(x or en)
   if(en)
   begin
      case(x)
         2'd0: y = 4'b0001;
         2'd1: y = 4'b0010;
         2'd2: y = 4'b0100;
         2'd3: y = 4'b1000;

         default: y = 4'b0000;
      endcase
   end
   else y = 4'b0000;

endmodule

```

2.4-2编码器

```
module exp2_2(x,en,y);
   input [3:0] x;
   input en;
   output reg [1:0] y;
   always @ (x or en)
      if(en)
      begin
         case (x)
            4'b0001: y=2'b00;
            4'b0010: y=2'b01;
            4'b0100: y=2'b10;
            4'b1000: y=2'b11;
         default: y=2'b00;
         endcase
      end
      else
         y=2'b00;

endmodule

```

3. 4-2优先编码器

```
module exp2_3(x, en, y);
    input [3:0] x;
    input en;
    output reg [1:0] y;
    integer i=0;
    always @(x or en)
        begin
            if(en)
            begin
                y=0;
                for(i=0; i<=3; i=i+1)
                    if(x[i]==1) y=i;
            end
            else y=0;
        end

endmodule

```

4. 8-3优先编码器

完成编码部分

```
module en83(x, en, y);
    input [7:0] x;
    input en;
    output reg[2:0] y;
    integer i=0;
    always @(x or en)
        begin
            if(en == 1) begin
                y=0;
                for(i=0; i<=7; i=i+1)
                    if(x[i]==1) y=i;
            end
            else y=0;

        end

endmodule

```

利用task完成到七段码的转换

```
module en83(x, en, y, z);
    input [7:0] x;
    input en;
    output reg[2:0] y;
    output reg[6:0] z;//HEX_LED
    integer i=0;
    /*task trans;
    input [3:0]A;
    output [6:0]B;
    case(A)
        0: B<=7'b1000000;       //0
        1: B<=7'b1111001;       //1
        2: B<=7'b0100100;       //2
        3: B<=7'b0110000;       //3
        4: B<=7'b0011001;       //4
        5: B<=7'b0010010;       //5
```

```
        6: B<=7'b0000010;       //6
        7: B<=7'b1111000;       //7
        8: B<=7'b0000000;       //8
        9: B<=7'b0010000;       //9
        10: B<=7'b0001000;  //a
        11: B<=7'b0000011;  //b
        12: B<=7'b1000110;  //c
        13: B<=7'b0100001;  //d
        14: B<=7'b0000110;  //e
        15: B<=7'b1001110;  //f
    default: B<=7'b1111111;
    endcase
    endtask
    */
always @(*)
    begin
        if(en == 1) begin
            y=0;
            for(i=0; i<=7; i=i+1)
                if(x[i]==1) begin
                y=i;
                //trans(y,z);
                end
        end
        else y=0;
        //trans(y,z);
case(y)
        3'b000: z<=7'b1000000;      //0
        3'b001: z<=7'b1111001;      //1
        3'b010: z<=7'b0100100;      //2
        3'b011: z<=7'b0110000;      //3
        3'b100: z<=7'b0011001;      //4
        3'b101: z<=7'b0010010;      //5
        3'b110: z<=7'b0000010;      //6
        3'b111: z<=7'b1111000;      //7
default: z<=7'b1111111;
endcase
end
endmodule
```

由于还没有专门学过task语句，导致仿真实验的时候发现所有七段码都延迟了10ns，结果不正确，于是使用常用的case转换

```
module en83(x, en, y, z);
    input [7:0] x;
    input en;
    output reg[2:0] y;
    output reg[6:0] z;//HEX_LED
    integer i=0;
    always @(*)
       begin
          if(en == 1) begin
             y=0;
             for(i=0; i<=7; i=i+1)
                if(x[i]==1) begin
                y=i;
                //trans(y,z);
                end
          end
          else y=0;
          //trans(y,z);
    case(y)
          3'b000: z<=7'b1000000;      //0
          3'b001: z<=7'b1111001;      //1
          3'b010: z<=7'b0100100;      //2
          3'b011: z<=7'b0110000;      //3
          3'b100: z<=7'b0011001;      //4
          3'b101: z<=7'b0010010;      //5
          3'b110: z<=7'b0000010;      //6
          3'b111: z<=7'b1111000;      //7
    default: z<=7'b1111111;
    endcase
    end

endmodule

```

五、实验步骤

- 根据讲义内容实现2-4译码器，4-2编码器，4-2优先编码器，并进行仿真实验验证正确性，熟悉译码器，编码器的原理，以及case语句的用法。
- 编写8-3编码器的编码部分，编写激励代码，利用仿真实验验证三位输出以及sign信号的正确性。
- 编写输出到七段码的case代码，利用仿真实验验证。
- 将八位输入，一位使能端输入，三位编码输出，一位信号输出，七段码输出的引脚分配完成并编译。
- 连接FPGA开发板，进行硬件验证，结果正确。

六、测试代码

1. 激励代码

通过在激励代码中枚举所有可能的情况，在仿真波形图中对结果进行验证

2-4译码器的激励代码

```
en=0;   x=2'b00;    #10;
        x=2'b01;    #10;
        x=2'b10;    #10;
        x=2'b11;    #10;
en=1;   x=2'b00;    #10;
        x=2'b01;    #10;
        x=2'b10;    #10;
        x=2'b11;    #10;
```

4-2编码器的激励代码

```
en=0;       x=4'b0001;  #10;
            x=4'b0010;  #10;
            x=4'b0100;  #10;
            x=4'b1000;  #10;
            x=4'b1111;  #10;
en=1;       x=4'b0001;  #10;
            x=4'b0010;  #10;
            x=4'b0100;  #10;
            x=4'b1000;  #10;
            x=4'b1111;  #10;
```

4-2优先编码器的激励代码

```
en=1;       x=4'b0001;  #10;
            x=4'b0010;  #10;
            x=4'b0011;  #10;
            x=4'b0100;  #10;
            x=4'b0101;  #10;
            x=4'b0110;  #10;
            x=4'b0111;  #10;
            x=4'b1000;  #10;
            x=4'b1001;  #10;
            x=4'b1010;  #10;
            x=4'b1011;  #10;
            x=4'b1100;  #10;
            x=4'b1101;  #10;
            x=4'b1110;  #10;
            x=4'b1111;  #10;
en=0;       x=4'b0001;  #10;
            x=4'b0010;  #10;
            x=4'b0011;  #10;
            x=4'b0100;  #10;
            x=4'b0101;  #10;
            x=4'b0110;  #10;
            x=4'b0111;  #10;
            x=4'b1000;  #10;
            x=4'b1001;  #10;
            x=4'b1010;  #10;
            x=4'b1011;  #10;
            x=4'b1100;  #10;
            x=4'b1101;  #10;
            x=4'b1110;  #10;
            x=4'b1111;  #10;
```

8-3优先编码器的激励代码

```
en=1;       x=8'b00000000;  #10;
            x=8'b00000001;  #10;
            x=8'b00000011;  #10;
            x=8'b00000111;  #10;
            x=8'b00001111;  #10;
            x=8'b00011111;  #10;
            x=8'b00111111;  #10;
            x=8'b01111111;  #10;
            x=8'b11111111;  #10;
en=0;       x=8'b00000000;  #10;
            x=8'b00000001;  #10;
            x=8'b00000011;  #10;
            x=8'b00000111;  #10;
            x=8'b00001111;  #10;
            x=8'b00011111;  #10;
            x=8'b00111111;  #10;
            x=8'b01111111;  #10;
            x=8'b11111111;  #10;
```

8-3优先编码器的引脚分配

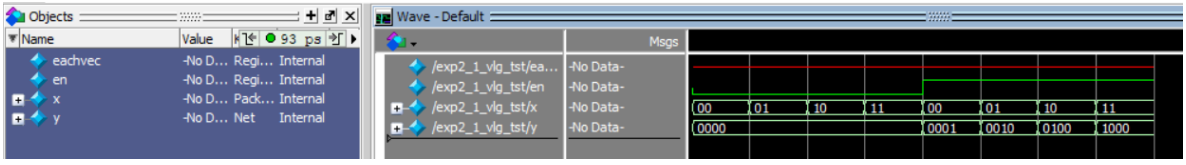
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	urrent Streng	Slew Rate
en	Input	PIN_AC29	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
sign	Output	PIN_AD24	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
x[7]	Input	PIN_AD30	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[6]	Input	PIN_AC28	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[5]	Input	PIN_V25	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[4]	Input	PIN_W25	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[3]	Input	PIN_AC30	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[2]	Input	PIN_AB28	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[1]	Input	PIN_Y27	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
x[0]	Input	PIN_AB30	5B	B5B_N0	2.5 V ...fault)		12mA ...ault)	
y[2]	Output	PIN_AC23	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
y[1]	Output	PIN_AB23	5A	B5A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
y[0]	Output	PIN_AA24	5A	B5A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[6]	Output	PIN_AH18	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[5]	Output	PIN_AG18	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[4]	Output	PIN_AH17	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[3]	Output	PIN_AG16	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[2]	Output	PIN_AG17	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[1]	Output	PIN_V18	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)
z[0]	Output	PIN_W17	4A	B4A_N0	2.5 V ...fault)		12mA ...ault)	1 (default)

2. 硬件测试

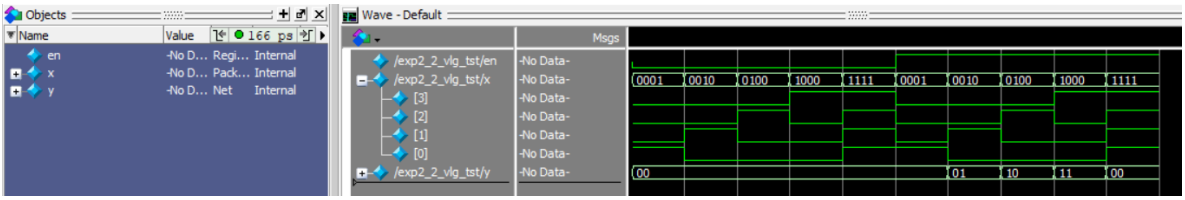
在开发板上验证所有可能的输入，观察输出结果与输出逻辑是否一致来进行验证

七、实验结果

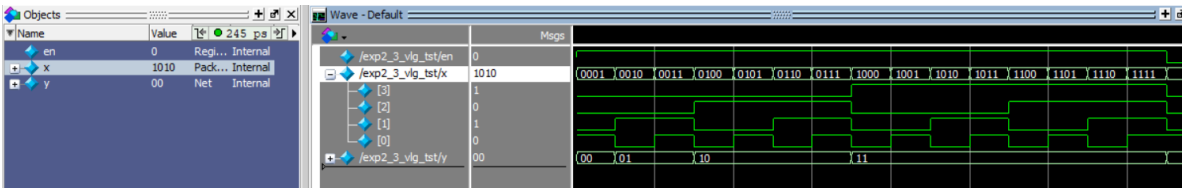
1. 2-4译码器



2. 4-2编码器



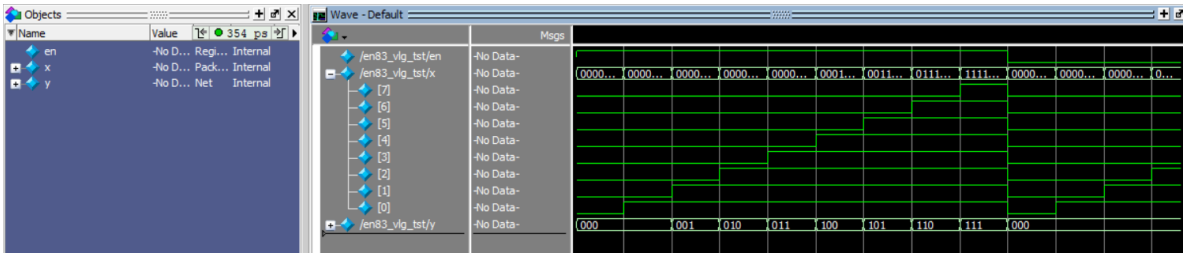
3. 4-2优先编码器



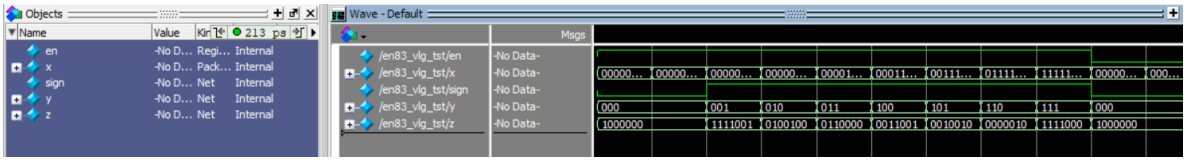
（后续en=0的时候，不在截图中，输出均为00）

4. 8-3优先编码器

仅编码部分



编码转换七段码结果



八、思考

如何利用casez/casex来实现8-3优先编码器?

在casez语句中，如果分支表达式某些位的值为高阻z，那么对这些位的比较就会忽略，不予考虑，而只关注其他位的比较结果。

在casex语句中，则把这种处理方式进一步扩展到对x的处理，即如果比较双方有一方的某些位的值是z或x，那么这些位的比较就不予考虑。以下是通过casex实 现的优先编码器：

```
8      always @(*)
9          begin
10             if(en == 1) begin
11                 y=0;
12                 //for(i=0; i<=7; i=i+1)
13                     //if(x[i]==1) begin
14                     //y=i;
15                     //trans(y,z);
16                     //end
17                 casex(x)
18                     8'b00000001: y=0;
19                     8'b0000001x: y=1;
20                     8'b000001xx: y=2;
21                     8'b00001xxx: y=3;
22                     8'b0001xxxx: y=4;
23                     8'b001xxxxx: y=5;
24                     8'b01xxxxxx: y=6;
25                     8'b1xxxxxxx: y=7;
26                 default y=0;
27                 endcase
28             end
29             else y=0;
30             //trans(y,z);
31             if(y) sign=1;
32             else sign=0;
```

九、实验中遇到的问题及解决方法

在实验中本来想将三位二进制码转到七段码的过程模块化，利用task完成，但是使用了几种不同方式都不能得到正确答案，所以我选择了更常用的case语句，但我依旧认为应该将其函数化，因为当前只有8个case，当case数量上升时，函数的价值才能体现。

在做思考题的时候不太理解casez和casex的用法，去网络上搜索了相关资料，尝试写了一个8-3优先编码器，结果是正确的。

十、实验得到的启示

前期准备要做好，无论是新语法的概念，还是功能的模块化，都要在编写代码前准备好，前期花费的时间可能较多，但能使代码显得更简洁，提高效率。

十一、意见和建议

无。