《数字电路与数字系统实验》实验报告

第 3 次实验： 4-2 编码器及 8-3 优先编码器的实现

姓名： 张逸凯

学号： 171840708

院系: 物理 学院

邮箱： 645064582@qq.com

电话： 18051988316

实验时间: 2019 年 3 月 18 日

〇.预习部分

可以采用硬件描述语言来实现 2-4 译码器电路。表 2-1是一个带有使能端的 2-4 译码器的代码：

表 2-1: 2-4 译码器代码

```
module decode24(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;
       endcase
    end
    else  y = 4'b0000;

endmodule
```

表 2-2: 2-4 译码器测试代码

```
`timescale 10 ns/ 1 ps
module test_decode24();
  reg  [1:0] x;
  reg  en;
  wire [3:0]y;
  decode24  i1 (
              . x (x),
              . en (en),
              .y(y) );
  initial
  begin
    en = 1'b0; x = 2'b00;  #10;
               x = 2'b01;  #10;
               x = 2'b10;  #10;
               x = 2'b11;  #10;
    en = 1'b1; x = 2'b00;  #10;
               x = 2'b01;  #10;
               x = 2'b10;  #10;
               x = 2'b11;  #10;
  end
endmodule
```

通过预习了解了 for 循环语句可以用来实现译码器， 但译码器具体的实现方式还得视情况而定，*比如 2，4 译码器可以用 case 语句来实现，而 3，8*

或者 4，16 等更高级的译码器用 case 语句显得繁琐，所以利用 for 循环此时可以比较简洁. 在别人的博客里有这样一种说法:”硬件设计和软件编程不同，在 C 语言里 if…else 和 for 循环满天飞，可以说用这两个语句打天下都是不成问题的，但是 HDL 设计中这是万万不可的”，以后还是要多留心注意.

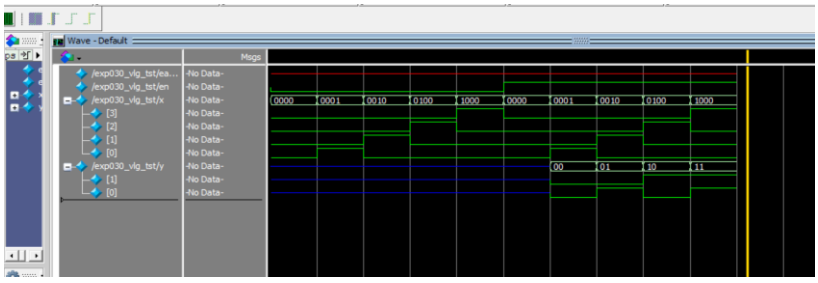
预习了后半部分后发现编码器也可以用 case 语句和 for 循环两种格式来实现，普通编码器用 for 循环实现的时候要*加一个整型变量用于计数有多少个 1（用一个变量 flag 标记第一个 1 出现的位置）*，如果超过 1 个或者 0 个 1，那么此时输出编码为 zzzzz，否则正常编码. 此时用 case 语句代码量会大一些，但是思路更清晰. 但是在编写优先编码器的时候使用 for 循环语句是比较方便的，从左到右找到第一个 1 后就可以直接编码了. 就是优先级有差异的原因.

➢ 预习部分之 4-2 编码器

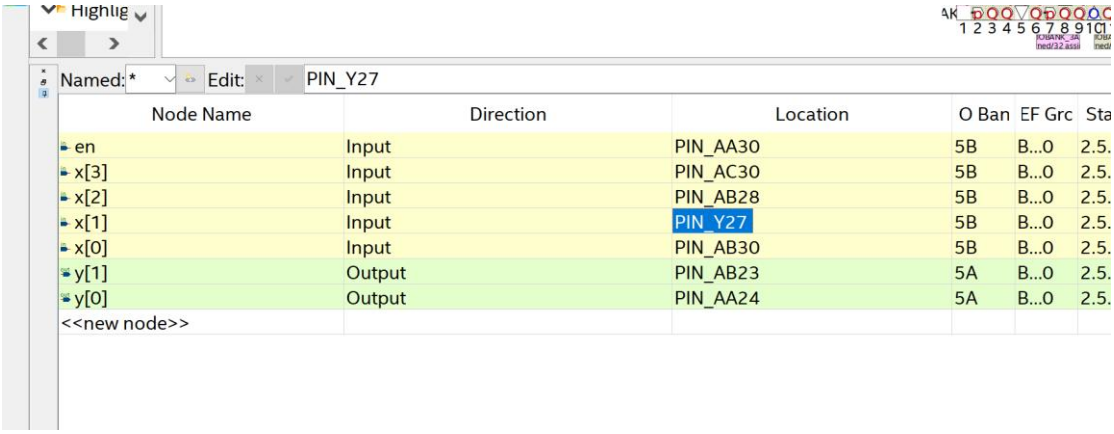
4-2 编码器即为 4 位输入 2 位输出的编码器，它的输入信号是 x0、x1、x2 和 x3，输出是 y0 和 y1。我们采用独热码，每次输入中只有一位为 1，对于有两位或者两位以上为 1 的情况，则将输出置为高阻态。

4-2 编码器 PDF 中的代码已经基本给出了 8-3 优先编码器的一种结构了，这样子的代码还是很简洁漂亮的. *设计代码和激励代码都是用 PDF 中的(但是在测试代码中直接把 x 的值赋上而不是通过 s).*

4-2 编码器仿真波形:



4-2 编码器引脚分配:



具体的实现图像忘记拍照了呜呜呜，但是助教哥检查过了的

下面开始 8-3 优先编码器

一. 实验目的

预习译码器、编码器以及优先编码器的实现原理并且掌握这几种器件的设计方法， 学会用 verilog 语言设计这三种器件. 学会利用 FPGA 开发板上的七段显示管来显示数字.

二. 实验原理（ 知识背景，结合理论课总结 ）

- **基本定义：**
  - **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， 每个输入编码产生唯一的一个不同于其他的输出编码.

常用的二进制译码器是一个 n 路输入和 2n 路输出的逻辑电路. 在 2n 路

输出信号中，只有一条输出信号有效. 译码器有一个使能信号 En，当 En=0 时，无论输入为什么，译码器没有任何有效值输出；当 En=1 时，输入的值决定了输出信号的值.

编码器:

编码器是一种与译码器功能相反的逻辑电路，编码器的输出编码比其输入编码位数少. *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，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.

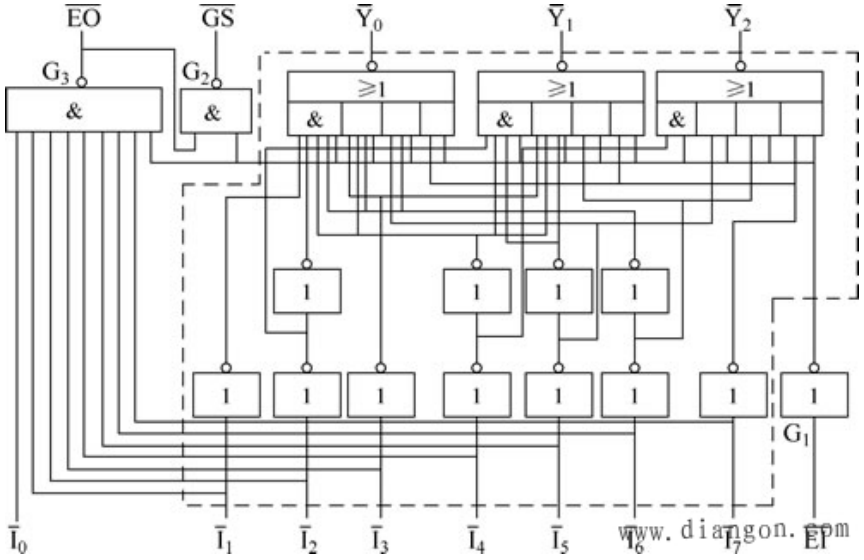
七段显示器采用七段译码，一般情况下，其输入为 4 位的 BCD 码，输出为 7 位的编码，用于驱动七段显示器的不同位，以显示出不同的数字.

4-2 编码器真值表:

x_3	x_2	x_1	x_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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，*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.



上图为 8-3 优先编码器的电路实现图.

对应的真值表: (Excel 输入，累死了)

EI	I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
1	X	X	X	X	X	X	X	X	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1
0	X	X	X	X	X	X	X	0	0	0	0
0	X	X	X	X	X	X	0	1	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0
0	X	X	X	X	0	1	1	1	0	1	1
0	X	X	X	0	1	1	1	1	1	0	0
0	X	X	0	1	1	1	1	1	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1

从该表可以很直观地看出优先编码器其实就是用*电路实现的用优先级进行编码的器件*，其中 I7 优先级最高，然后 I6…

三.　实验设备环境

硬件器材：FPGA 开发板.
软件平台：Qaurtus 开发平台.

四.实验步骤 / 过程（设计思路、设计代码、测试代码、仿真结果和硬件实现等的截图代码等）

- **设计思路:**

基于 PPT 例子

表 2-6为该 4-2 编码器的测试代码。

表 2-6: 4-2 编码器测试代码

```
module test_encode4_2();
    reg en;
    reg [3:0] s;
    wire [1:0] y;

    encode42 t1(
     .en(en),
     .x(s),
     .y(y));
    initial
    begin
      en=1'b0;  s =4'b0000; #10;
                s =4'b0001; #10;
                s =4'b0010; #10;
                s =4'b0100; #10;
                s =4'b1000; #10;
      en=1'b1;  s =4'b0000; #10;
                s =4'b0001; #10;
                s =4'b0010; #10;
                s =4'b0100; #10;
                s =4'b1000; #10;
    end
endmodule
```

模仿 PDF 给出的例子写出代码，同样的利用 for 循环就很容易找到优先级了，但是注意***要添加一个一位的输出，***当输入的 8 位二进制值有效时输出为 1，当输入的 8 位 2 进制数全 0 时输出为 0.

至于在七段数码管上显示，在代码中***使用一个 case 语句***就可以显示数字了．见下面代码展示:

- **设计代码:**

```
1  module exp03(x, en, y, flag, seg);
2     input [7:0] x;
3     input en;
4     output reg[2:0] y;
5     output reg flag;
6     output reg [6:0] seg;
7     integer i;
8     always @ (x or en)
9     begin
10       if(x == 0) flag = 0;
11       else flag = 1;
12
13       if(en) begin
14          y = 0;
15          for(i = 0; i <= 7; i = i + 1)
16          if(x[i]==1) y = i;
17       end
18       else y = 0;
19       case(y)
20       0: seg=7'b1000000;
21       1: seg=7'b1111001;
22       2: seg=7'b0100100;
23       3: seg=7'b0110000;
24       4: seg=7'b0011001;
25       5: seg=7'b0010010;
26       6: seg=7'b0000010;
27       7: seg=7'b1111000;
28       endcase
29    end
30 endmodule
31
```

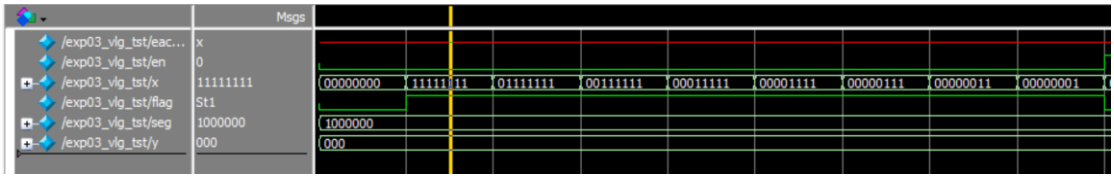
图1　设计代码

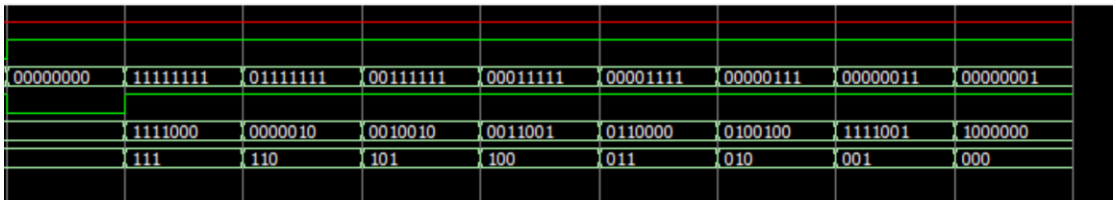
激励代码:

```
// constants
// general purpose registers
reg eachvec;
// test vector input registers
reg en;
reg [7:0] x;
// wires
wire flag;
wire [6:0]  seg;
wire [2:0]  y;

// assign statements (if any)
exp03 i1 (
// port map - connection between master ports and sig
   .en(en),
   .flag(flag),
   .seg(seg),
   .x(x),
   .y(y)
);
initial
begin
// code that executes only once
// insert code here --> begin
   en=1'b0; x=8'b00000000;#10;
            x=8'b11111111;#10;
            x=8'b01111111;#10;
            x=8'b00111111;#10;
            x=8'b00011111;#10;
            x=8'b00001111;#10;
            x=8'b00000111;#10;
            x=8'b00000011;#10;
            x=8'b00000001;#10;
   en=1'b1; x=8'b00000000;#10;
            x=8'b11111111;#10;
            x=8'b01111111;#10;
            x=8'b00111111;#10;
            x=8'b00011111;#10;
            x=8'b00001111;#10;
            x=8'b00000111;#10;
            x=8'b00000011;#10;
            x=8'b00000001;#10;
end
endmodule
```

ModelSim 仿真波形:

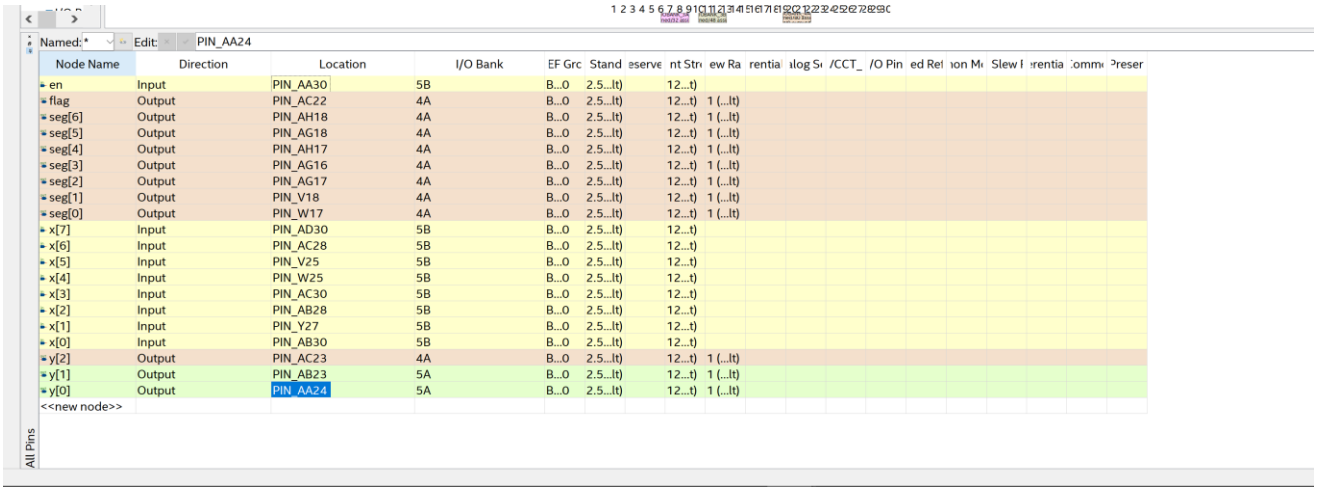




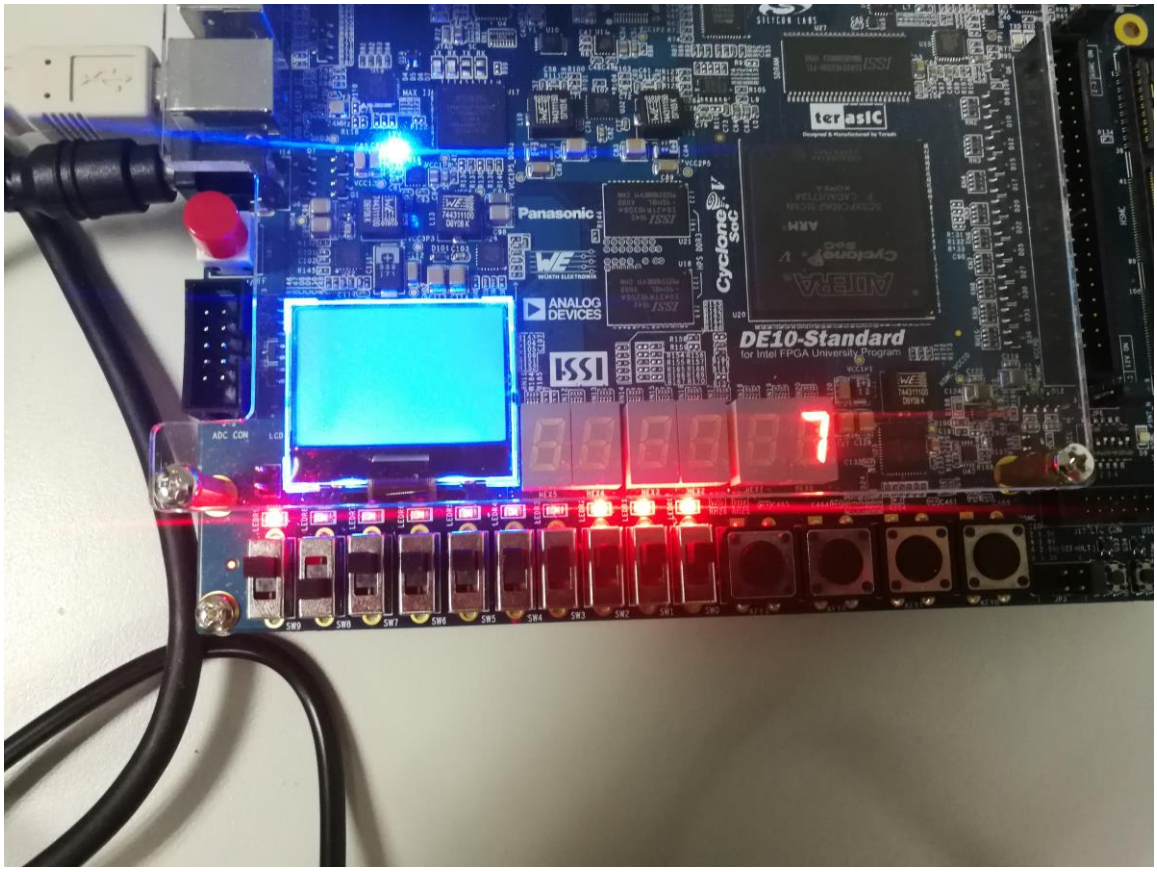
仿真结果和预期一样，当 en 为 0 时，y 为 0，当 en 为 1 时为正常的 83 优先编码器．当输入的 x 为 0 时，输出 flag 值为 0，否则输出 flag 值为：1.

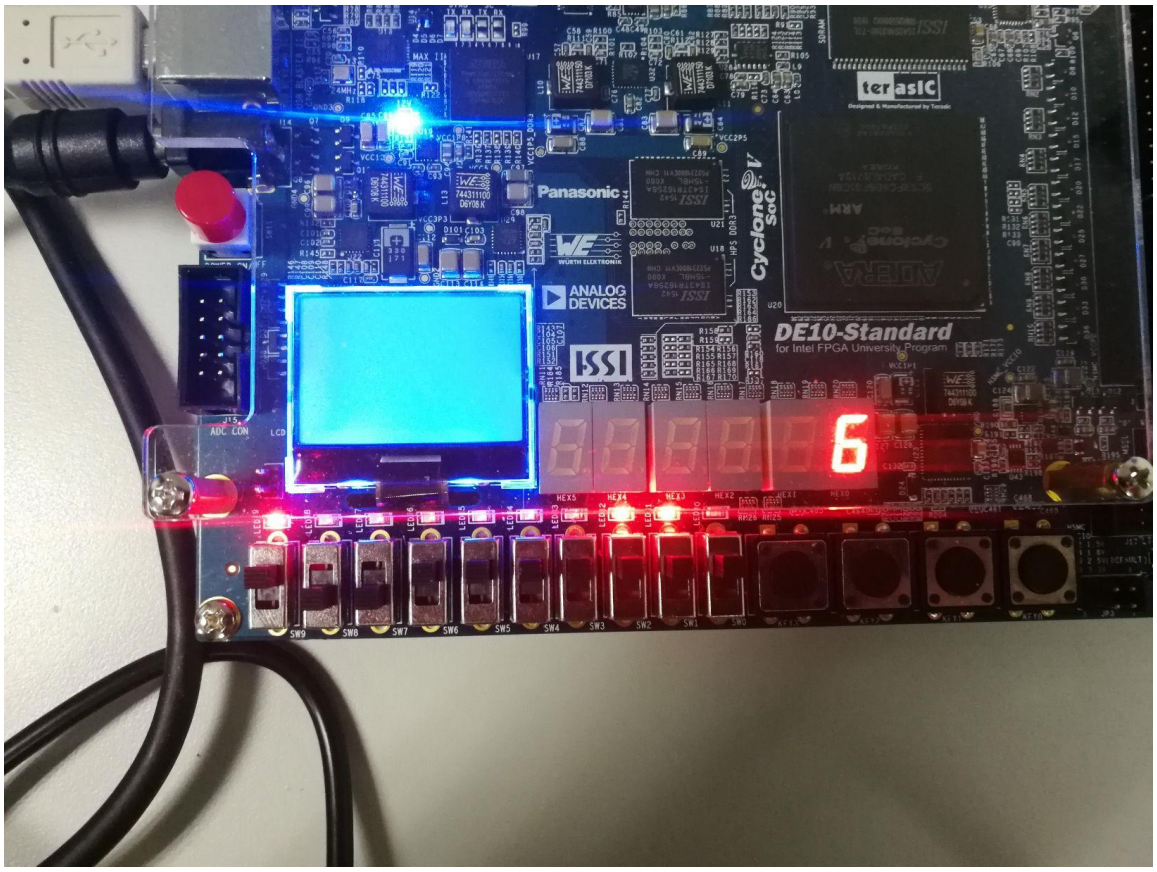
➢ 硬件实现:

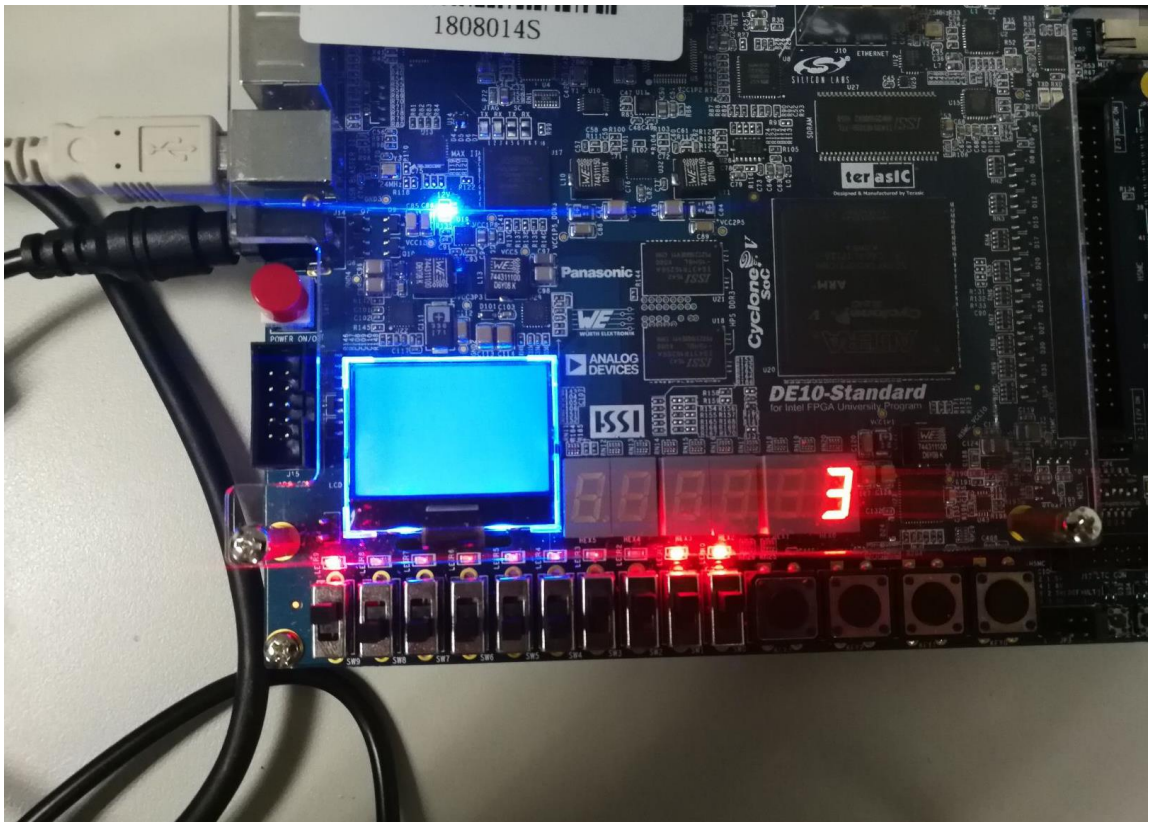
引脚分配:

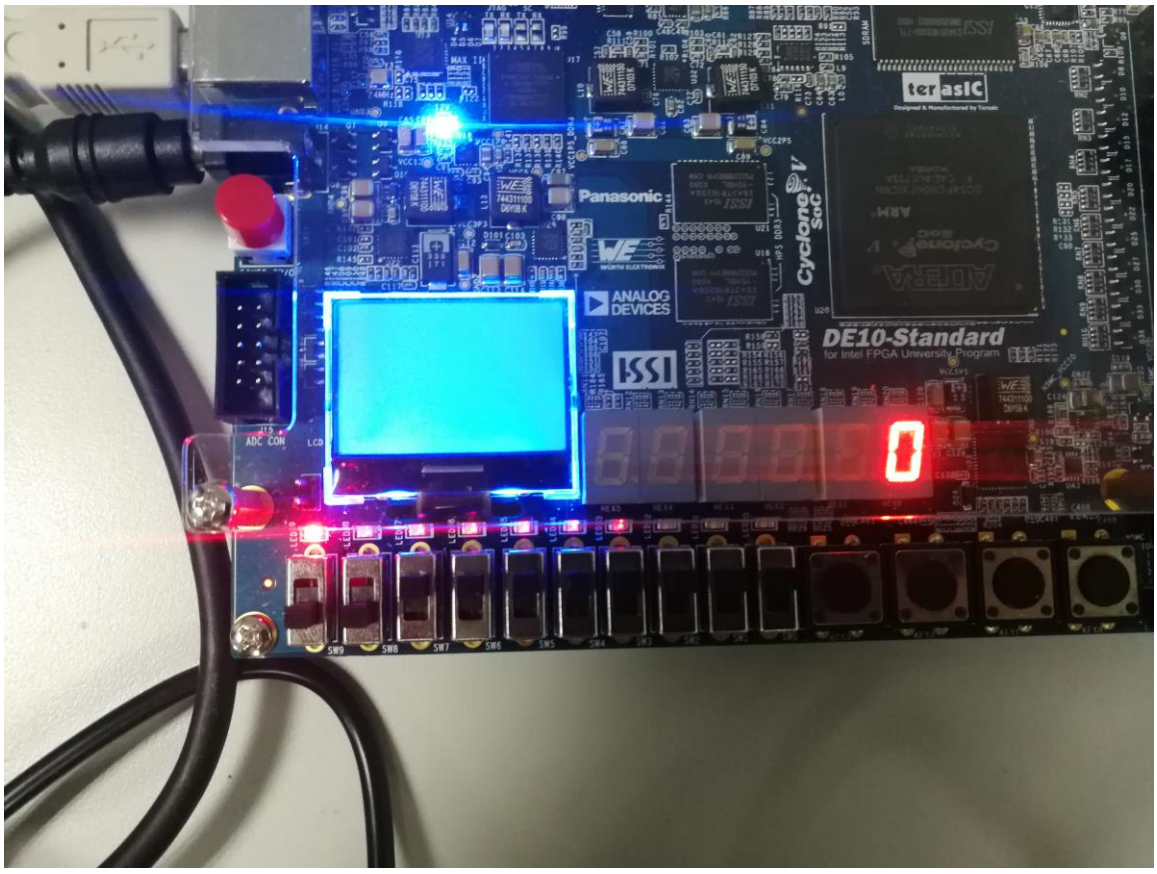


- 开发板实现


terasIC
Panasonic
ANALOG DEVICES
ISSI
DE10-Standard
for Intel FPGA University Program
ADC CON
LCD

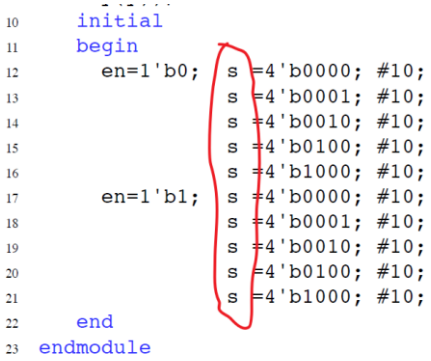

terasIC
Panasonic
ANALOG DEVICES
ISSI
DE10-Standard
for Intel FPGA University Program
ADC CON
LCD


1808014S
terasIC
Panasonic
ANALOG DEVICES
ISSI
DE10-Standard
for Intel FPGA University Program
3


terasIC
Panasonic
ANALOG DEVICES
ISSI
DE10-Standard
for Intel FPGA University Program
0

五.实验中遇到的问题及解决方案（请具体的描述问题和解决方法）

4-2 编码器 PDF 里面的代码是错的(orz)，虽然是很小的错误:



```
10      initial
11      begin
12        en=1'b0;  s =4'b0000; #10;
13                  s =4'b0001; #10;
14                  s =4'b0010; #10;
15                  s =4'b0100; #10;
16                  s =4'b1000; #10;
17        en=1'b1;  s =4'b0000; #10;
18                  s =4'b0001; #10;
19                  s =4'b0010; #10;
20                  s =4'b0100; #10;
21                  s =4'b1000; #10;
22      end
23  endmodule
```

这里的 s 未定义，估计是手误？直接 x 上就好了呀为什么要把 s 赋给 x 再多来一遍？仿真的时候就出现了 undefined 的错误. 直接把 s 改成 x 就可以了.

实验的时候开始没有将 flag 表示为 reg 型变量，后来经过错误提示才发现因为 flag 在 always 语句中有赋值，所以必须要声明为 reg 型变量.

六.实验得到的启示（积极思考）

在 https://blog.csdn.net/hanghang121/article/details/23449467 这样的网站中提到了 casez 和 casex，查找资料学习这两个东西:

- 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.
- 同时还要注意在 casez 和 casex 语句中如果同时找到了两个匹配的选择，那么要以第一个为准. 这就相当于是优先编码器了.

简单写一下如何用 casex 实现优先编码器:

```
If(en)
   casex(x)
         8'b1??????? : begin y=3'b111; seg=7'b1111000; end
         8'b01?????? : begin y=3'b110; seg=7'b0000010; end
         8'b001????? : begin y=3'b101; seg=7'b0010010; end
         8'b0001???? : begin y=3'b100; seg=7'b0011001; end
         8'b00001??? : begin y=3'b011; seg=7'b0110000: end
         8'b000001??: begin y=3'b010; seg=7'b0100100; end
         8'b0000001?; begin y=3'b001; seg=7'b1111001; end
         8'b00000001: begin y=3'b000; seg=7'b1000000; end
         default: begin y=3'b000; seg=7'b1000000: end
   endcase
else begin y=3'b000; seg=7'b1000000; end
```

七.意见和建议等

老实说这次实验还是花了很大功夫，上网找资料之类，然后对于之前的知识掌握得不是很牢固，也是花了很多时间在复习上．不过老师有给几个例子，慢慢学还是挺好的.

这次实验也是比较后面才开始做，一些同学已经把 PDF 欠缺的问题都说了，比如使能端为 0 时，七段显示管怎么弄，也省了很多时间.

做实验时出现错误不要慌,要慢慢去寻找，这样能节约很多的时间，同时做实验要看清实验要求，这样也会少花一些功夫.