2020-9-15

数字电路实验二 8-3 优先编码器

秦嘉余

191220088

1348288404@qq.com

目录

1.实验目的

本次实验的目的是熟悉 8-3 优先级编码器的内部结构与 7 段显示管的构成，并在 quartus 环境下利用 verilog 语言编写相关模拟程序，模拟仿真测试后写入 DE-10 开发板上。

2.实验原理

编码器是一种组合逻辑电路的重要器件，编码器是一种将特定输入转化为某一信息输出的逻辑电路，通常为 2^n 输入，n 输出。由于其选择的特性，可以使用 verilog 中的 case 语句来完成，case 的结构功能与编码器的性质最为吻合。当 n 较大时，枚举每一种情况显然不是最好的解决方式，同时我们需要完成的是一种优先编码器，可能同时有多个有效输入，所以显然可以的情况无法全部模拟完全。所以此时可以利用 verilog 中的 for 循环语句，通过 n 到 0 或者 0 到 n 顺序，可以改变优先级。7 段显示管由 7 个 LED 灯管组成，通过 case 语句，把编码的每种情况对于的 8 进制表示数需要亮的灯管输入 0（低电平有效），即可达到效果。

3.实验环境/器材

本次实验的环境为 Quartus17.1 版本
本次实验的器材为 DE10 Standard 开发板

4.程序代码

本次实验需要实现 8-3 优先编码器，同时将编码结果的 8 进制表示引入到 7 段显示器中。输入有：8 路输入端 x，使能端 en；输出端有：3 路输出 y，是否有输入判断端 is_input，7 段显示管 HEX

```
module encode83(x,en,y,is_input,HEX);
    input [7:0] x;
    input en;
    output reg [2:0] y;
    output reg is_input;
    output reg [6:0] HEX;
    integer i;
    always @(x or en)
        begin
            if(en)
                begin
                    is_input=0;
                    y=0;
                    for(i=0;i<=7;i=i+1)
                        if(x[i]==1)
                            begin
                                y=i;
                                is_input=1;
                            end
                end
            else
                begin
                    y=0;
                    is_input=0;
                end
            case (y)
                0: HEX=7'b1000000;
                1: HEX=7'b1111001;
                2: HEX=7'b0100100;
                3: HEX=7'b0110000;
                4: HEX=7'b0011001;
                5: HEX=7'b0010010;
                6: HEX=7'b0000010;A
                7: HEX=7'b1111000;
                default: HEX=7'b1111111;
            endcase
        end
endmodule
```

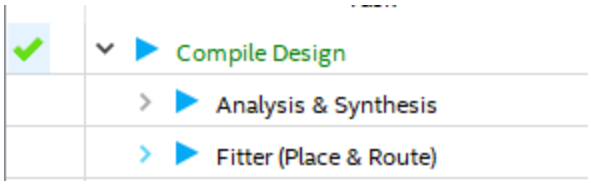
红框框选的第一部分是编码部分，根据输入进行编码，同时遵循优先级，同时如果有输入，将 is_input 赋值为 1

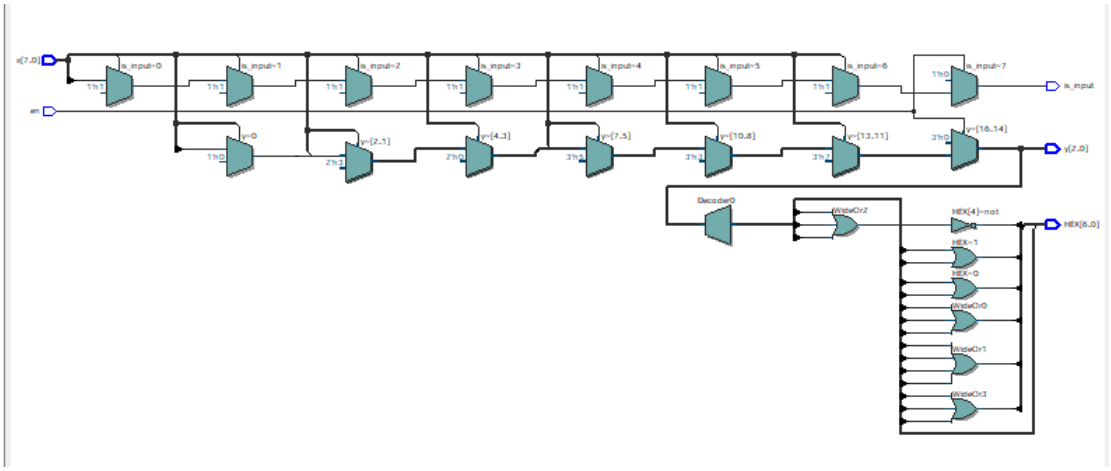
红框框选的第二部分是七段显示管的部分，根据最后输出 y 的值，给 HEX 不同的排列输出，使其显示不同的数字

5.实验步骤

该部分主要展现是实验中各步骤

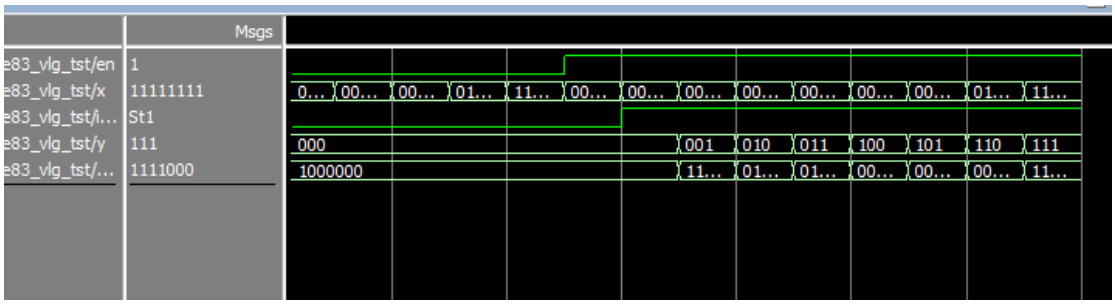
第一步：根据器件建立合适的项目文件，新建实验代码文件，键入相关代码，并且分析与综合，查看电路原理图，展示如下：



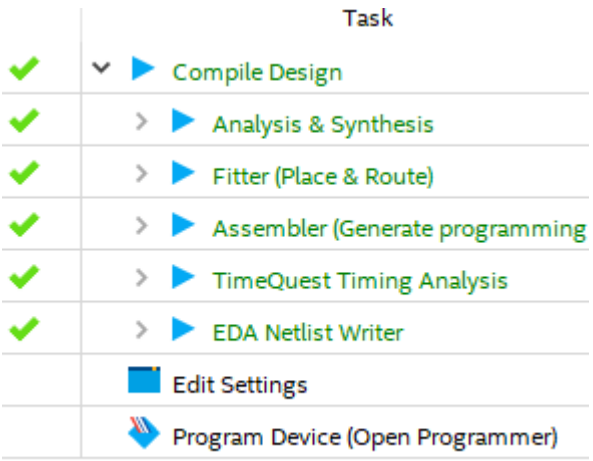


第二步：编写模拟测试代码，在 Test Benches 中链接相关模拟测试文件，最后利用 RTL Simualtion 查看波形图，测试程序是否正确，展示如下：

```
);
 initial
begin
 // code that executes only once
 // insert code here --> begin
    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;
    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;

 // --> end
 //$display("Running testbench");
end
```



第三步：第二步正确后，将引脚对应到开发板的相应接口上，再编译全部项目，连接 DE10 Standard 开发板到电脑，将程序写入开发板上，展示如下：



out HEX[6]	Output	PIN_AH18	4A	B4A_N0	PIN_AH18
out HEX[5]	Output	PIN_AG18	4A	B4A_N0	PIN_AG18
out HEX[4]	Output	PIN_AH17	4A	B4A_N0	PIN_AH17
out HEX[3]	Output	PIN_AG16	4A	B4A_N0	PIN_AG16
out HEX[2]	Output	PIN_AG17	4A	B4A_N0	PIN_AG17
out HEX[1]	Output	PIN_V18	4A	B4A_N0	PIN_V18

in x[1]	Input	PIN_AB28	5B	B5B_N0	PIN_AB28
in x[0]	Input	PIN_Y27	5B	B5B_N0	PIN_Y27
out y[2]	Output	PIN_AD24	4A	B4A_N0	PIN_AD24
out y[1]	Output	PIN_AC23	4A	B4A_N0	PIN_AC23

第四步：写入开发板后，测试开发板输入输出是否正确，展示如下：

自此，该实验完成

6.测试方法

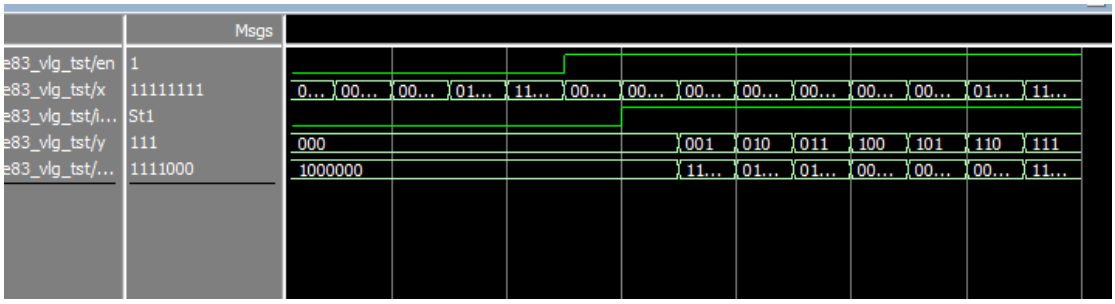
本次实验采用测仿真仿真的方法， 编写的测试仿真代码如下：

```
`timescale 10 ns/ 1 ps
module encode83_vlg_tst();
reg en;
reg [7:0] x;
wire is_input;
wire [2:0]  y;
wire [6:0] HEX;

encode83 i1 (
    .en(en),
    .is_input(is_input),
    .x(x),
    .HEX(HEX),
    .y(y)
);
initial
begin

    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;
    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;


end

endmodule
```

分为 en=1 与 en=0 的部分分别测试，由于为优先编码器，所以在输入选择时没有选择 00000001，00000010…的形式，而是 00000001,00000011 的形式，这样可以更好的体现优先级的特性，测试优先级的功能

7.实验结果

仿真结果如下：



下载运行结果如下：

实验验收完成，与仿真结果一致

8.实验中遇到的问题及解决办法

在写 case 的逻辑判断时，使用了传统的 case 语句，这样因为优先级编码器的特性稍显繁琐，之后了解了 casez 与 casex 语句后发现使用这种语句可以减少很多 case 前语句的判断，同时 case 语句对各种情况的实用性也更好、

9.思考题

casex 与 casez 都是为了应对 case 语句的特殊情况，我们知道 verilog 中的数据不只有 0 和 1 两种状态，还有 z 和 x 两种未知状态，但 case 中的状态都是确定状态不能含有 x 与 z，当我们 case 语句中，某些分支具有类似的特性的时候，可以用 x 或者 z 替代来指示，本题中的优先编码器则可以使用 casex 或者 casez 语句完成。需要注意，casez 中，每种情况中不确定的部分需要用？来替代，而 casex 语句不能用于可以仿真的电路文件中，所以该部分可以使用 casez 语句完成，如下

```
//use casez
// case (x)
//     8'b1???????:y=7;
//     8'b01??????:y=6;
//     8'b001?????:y=5;
//     8'b0001????:y=4;
//     8'b00001???:y=3;
//     8'b000001??:y=2;
//     8'b0000001?:y=1;
//     8'b00000001:y=0;
//     default:y=0;
// endcase
```