LAB2 译码器和编码器

一、译码器

1. 真值表

输入			输出							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

2. 布尔函数

$D_0 = x'y'z'$

$D_1 = x'y'z$

$D_2 = x'yz'$

$D_3 = x'yz$

$D_4 = xy'z'$

$D_5 = xy'z$

$D_6 = xyz'$

$D_7 = xyz$

3. 实验设计

(1)实验的**输入、输出**和对应的**管脚**

变量名	类型	描述	宽度	管脚
in	input	三位输入	3	J15,L16,M13
out	output	八位输出	8	H17,K15,J13,N14, R18,V17,U17,U16

备注：以上表中的管脚顺序对应于相应变量的二进制表示的顺序为：从权重较小的位到权重较大的位（如： in[0]对应的是 J15，in[1]对应的是 L16）

(2)根据真值表，可以看出输出的结果中只有下标与输入的值相等的位值为 1，其他都为 0，因此可以用循环来实现：

```
module decoder(
    input[2:0] in,
    output reg[7:0] out
    );

    integer i;
    always @(in) begin
        for (i = 0; i < 8; i = i+1) begin
            if (in == i)
                out[i] <= 1;
            else
                out[i] <= 0;
        end
    end
endmodule
```

4. 仿真

(1)测试代码：

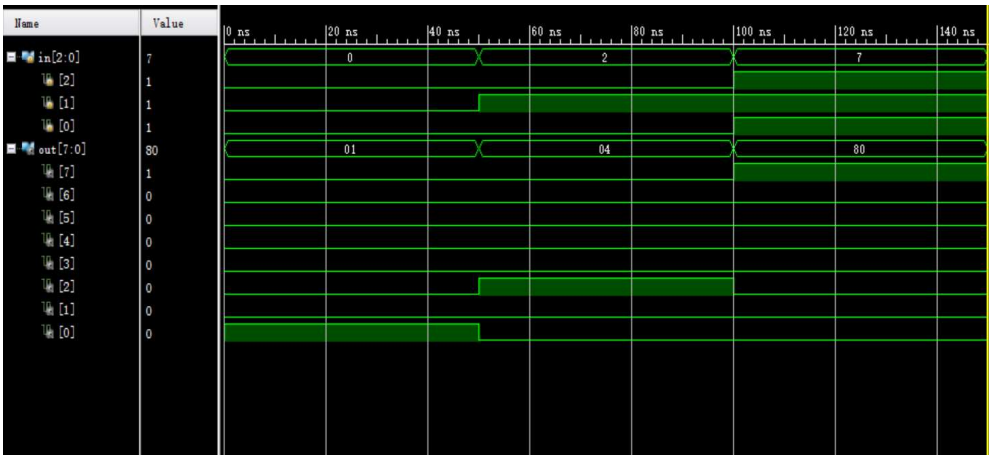
```
module test_decoder();
    reg [2:0]in;
    wire [7:0]out;
    decoder test(in, out);
    initial begin
        in = 3'b000;
        #50 in = 3'b010;
        #50 in = 3'b111;
     end
    initial begin
        $monitor($time,,, "%b = %b", in, out);
        #150
        $finish;
    end


endmodule
```

(2)从 000-111 我选取了三个输入测试，测试数据在控制台的输出如下(格式为：时间 输入=输出)：

```
  0  000 = 00000001
 50  010 = 00000100
100  111 = 10000000
```

(3)仿真波形



二、编码器

1. 真值表

输入				输出		
D_0	D_1	D_2	D_3	x	y	v
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

2. 布尔函数

$x = D_2 + D_3$

$y = D_3 + D_1D_2'$

$v = D_0 + D_1 + D_2 + D_3$

3. 实验设计

(1)实验的**输入、输出**以及对应的**管脚**

变量名	类型	描述	宽度	管脚
in	input	四位输入	4	J15，L16,M13,R15
out	output	两位结果+一位有效位	3	H17,K15,J13

备注：管脚的顺序与译码器相同

(2)根据真值表，对五种不同的输入，对应五种输出，因此**代码实现**如下，其中 out[0]是有效位，等于 1 表示结果有效，否则无效

```
module encoder(
    input[3:0] in,
    output [2:0] out
    );

    reg [2:0]tempout;
    integer i;
    always @(in) begin
        casex(in)
            4'b1000: tempout = 3'b100;
            4'bx100: tempout = 3'b101;
            4'bxx10: tempout = 3'b110;
            4'bxxx1: tempout = 3'b111;
            default: tempout = 3'b000;
        endcase
    end

    assign out = tempout;

endmodule
```

4. **仿真波形**

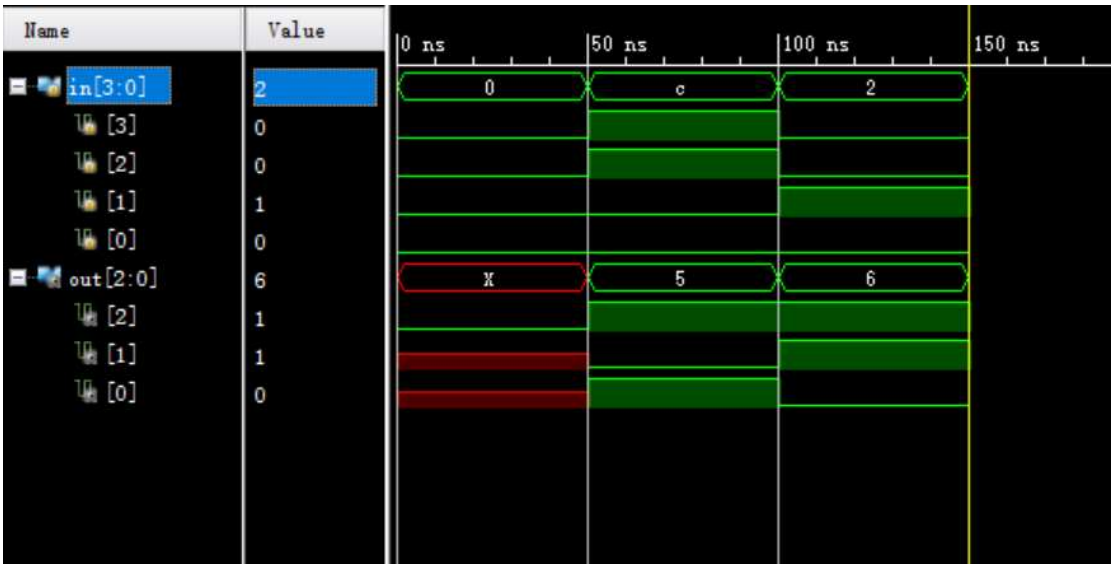
(1)**测试代码**

```
module test_encoder();
    reg [3:0]in;
    wire [2:0]out;
    encoder test(in, out);
    initial begin
        in = 4'b0000;
    end
    initial begin
            #50 in = 4'b1100;
            #50 in = 4'b0010;
     end
    initial begin
        $monitor($time,,, "%b = %b", in, out);
        #150
        $finish;
    end
endmodule
```

(2)**我选取了三中不同的输入进行测试，分别代表了无效输入、包含 don’t-care 的有效输入和不包含 don’t-care 的有效输入，对应在控制台的输出如下（格式：时间 输入=输出）：**

0 0000 = 0xx
50 1100 = 101
100 0010 = 110

(3)**仿真波形**



三、实验中遇到的问题和解决方案

1. Error: Syntax error near ‘for’

   原因：在包含改 for 循环的外层 always 语句的最后添加了多余的分号

2. 当包含不确定值 x 时，case 语句需要使用其拓展，也就是 casex 语句

四、实验中的体会与感想

因为有了 LAB1 的经验，这次做 LAB 时对软件的使用和整体设计的流程以及 verilog 的语法都熟悉了很多，总体感觉，这次的实验相比第一次更为简单。