

《EDA技术》试验

# 实验六

## 有限状态机设计与 实现

专业： 计算机科学与技术

姓名： 高星杰

学号： 2021307220712

班级： 计科 2102

报告上交时间：2023年 5月 7日

## 一、实验目的

- 1、理解有限状态机设计的基本原理；
- 2、掌握采用有限状态机设计电路的方法；
- 3、掌握有限状态机的 VerilogHDL 实现

## 二、实验任务及要求

注意：本次实验都用有限状态机做，要用 `parameter` 进行状态编码，程序中要有状态之间的转换。每道题最好设置一个复位端，最开始给系统一个复位信号。

**任务一、8 路彩灯控制器设计与实现**（只用开发板调试，不仿真；调试不正确又找不到错误时可仿真）彩灯花型自己定义，请在实验报告中介绍清楚自己所定义的彩灯花型。提示：DE2-115 开发板上的时钟频率是 50Mhz，频率太高，周期太短，人眼识别不了变化，故需要设计分频器，得到 4hz 的时钟来使用。

**任务二、1101 序列检测器设计与仿真**（先仿真，然后在开发板上调试）具体要求如下：

有一个总的同步使能开关 SW1，开关打开时才正常工作；通过开关 SW2 进行串型信号输入；每 0.25 秒检测一次输入信号；当完整检测到“111”序列时 LEDR0 灯点亮。已知开关管脚 SW1 为 AB27，SW2 为 AC27，发光二极管管脚 LEDR0 为 G25，50mhz 时钟信号管脚为 Y2。实验报告要求先画出状态转移图，然后根据状态

转移图写程序。

### 任务三、实现循环扫描显示的七段数码管译码控制电路的设计

要求：在 8 个数码管上循环显示 1~8，每个数字显示时间为 1s；

要求必须用有限状态机来做。

提示：1) DE2-115 开发板上的时钟频率是 50Mhz，频率太高，周期太短，人眼识别不了变化，故需要设计分频器，得到 4hz 的时钟来使用。

2) 每种状态都要分别给 8 个数码管赋值，不显示的数码管可以用十六进制数常量 7' h7F 赋值。

学有余力的同学可选做

**任务四：在 DE2-155 开发板上实现一个路口交通灯 30 秒倒计时器，采用两个七段数码管显示倒计时结果。**

提示：该题目的核心是做一个 BCD 码减法计数器和一个 BCD 码-7 段数码管 4-7 译码器，不需要用有限状态机，主要目的是对第三次实验复习和提升。

注意事项：（1）在用 reg 定义计数器位宽和其它变量位宽时，一定要根据变量的最大值给出足够大的位宽，否则调试易出现不易觉察的错误。（2）调试过程中出现错误又无法分析时，建议对程序进行仿真，根据仿真波形分析错误相对比较容易。

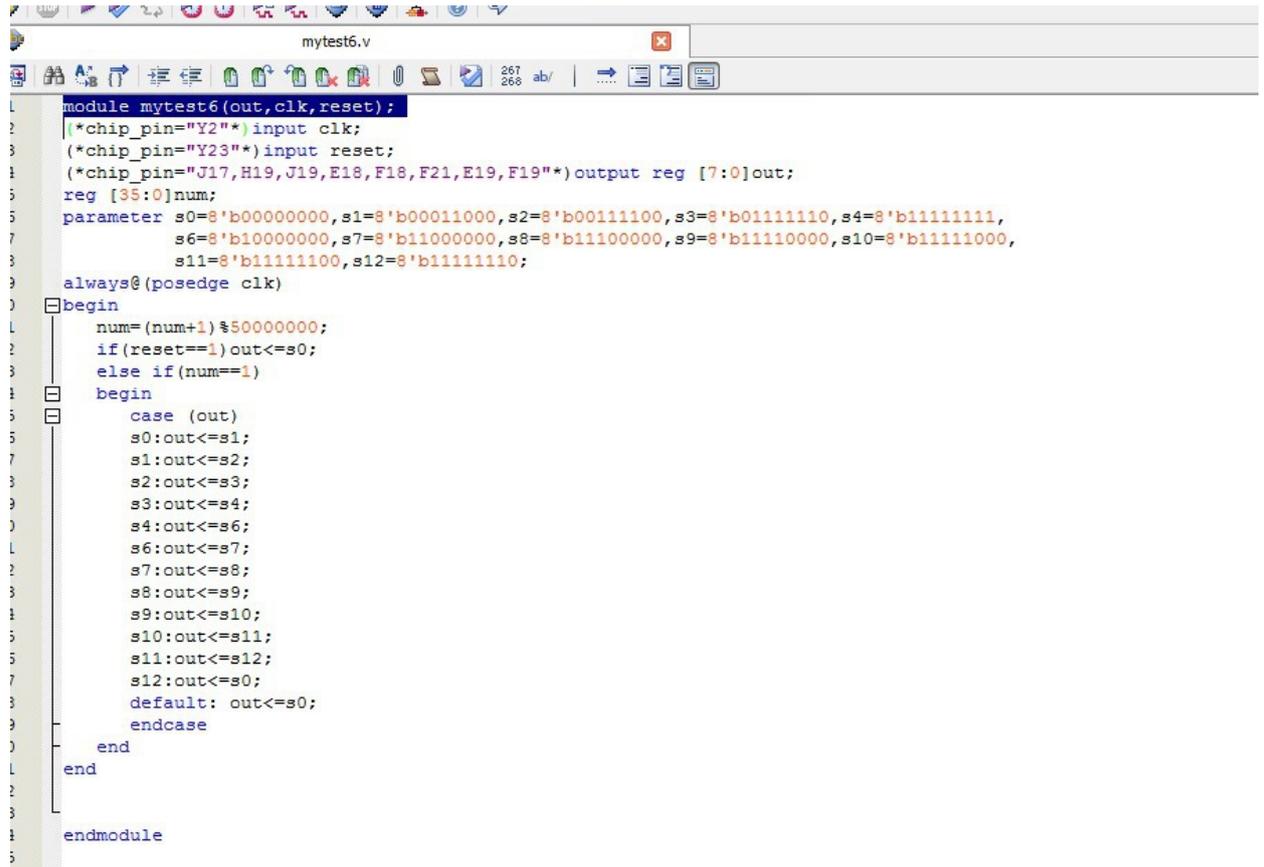
## 三、实验原理与步骤

### 任务一、8 路彩灯控制器设计与实现

原理：将 8 路彩灯的变换分为几个状态，画出状态转化图然后对这几

个状态编码，再实现有限状态机。这里我使用的是单过程实现的有限状态机，将状态机的现态、次态和输出逻辑放在一个 `always` 里进行描述。我实现的功能有从两边开始逐个亮到从头开始逐个亮然后循环往复，然后和复位信号的实现。

代码：

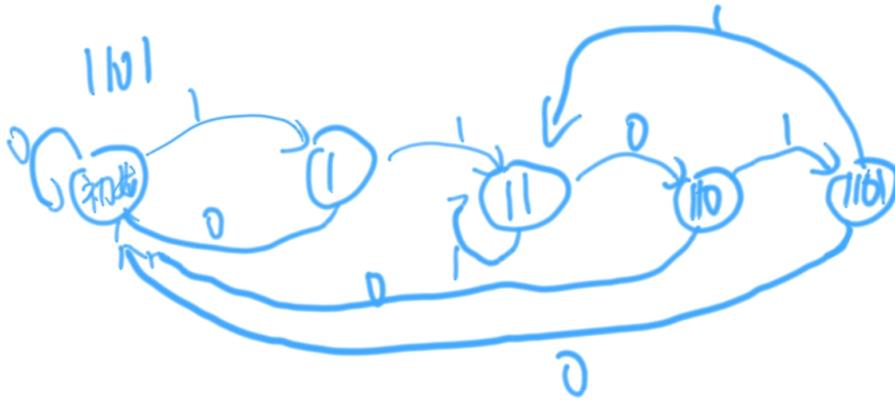


```
1 module mytest6(out,clk,reset);
2   (*chip_pin="Y2"*)input clk;
3   (*chip_pin="Y23"*)input reset;
4   (*chip_pin="J17,H19,J19,E18,F18,F21,E19,F19"*)output reg [7:0]out;
5   reg [35:0]num;
6   parameter s0=8'b00000000,s1=8'b00011000,s2=8'b00111100,s3=8'b01111110,s4=8'b11111111,
7             s6=8'b10000000,s7=8'b11000000,s8=8'b11100000,s9=8'b11110000,s10=8'b11111000,
8             s11=8'b11111100,s12=8'b11111110;
9   always@(posedge clk)
10  begin
11    num=(num+1)%50000000;
12    if(reset==1)out<=s0;
13    else if(num==1)
14    begin
15      case (out)
16        s0:out<=s1;
17        s1:out<=s2;
18        s2:out<=s3;
19        s3:out<=s4;
20        s4:out<=s6;
21        s6:out<=s7;
22        s7:out<=s8;
23        s8:out<=s9;
24        s9:out<=s10;
25        s10:out<=s11;
26        s11:out<=s12;
27        s12:out<=s0;
28        default: out<=s0;
29      endcase
30    end
31  end
32 endmodule
```

## 任务二、1101 序列检测器设计与仿真

**实验原理：**根据有限状态机设计的一般步骤来设计并且要设计一个分频器来检测输入数据，先根据状态写几个状态，然后画出状态转换图，再然后使用二进制对状态进行编码，然后可以根据这些设计出 Verilog 程序。

状态转化图：



代码:

```

module task2(in,out,work,clk);
(*chip_pin="AC27"*)input in;
(*chip_pin="Y2"*)input clk;
(*chip_pin="AB27"*)input work;
(*chip_pin="G25"*)output reg out;
reg [35:0]num;
reg [2:0]state;
parameter s0=3'b000,s1=3'b001,s2=3'b010,s3=3'b011,s4=3'b100;
always@(posedge clk)
begin
    num<=(num+1)%500000000;
    if(num==1&&work==1)
    begin
        if(in==1)
        begin
            case(state)
            s0:state<=s1;
            s1:state<=s2;
            s2:state<=s2;
            s3:state<=s4;
            s4:state<=s2;
            endcase
        end
    else
    begin
        case(state)
        s0:state<=s0;
        s1:state<=s0;
        s2:state<=s3;
        s3:state<=s0;
        s4:state<=s0;
        endcase
        end
        if(state==s4)
        out<=1;
    else out<=0;
    end
end
endmodule

```

### 任务三、实现循环扫描显示的七段数码管译码控制电路的设计

实验原理: 同样是有限状态机的设计, 由于状态之间的转换比较简单

所以没有画出状态转换图, 这里我实现的是双过程描述一个 always

描述现态和次态另一个 always 描述输出。我这里使用 show 模块来描述输出的逻辑。

代码:

```
module task3 (clk, reset, hex0, hex1, hex2, hex3, hex4, hex5, hex6, hex7);
(*chip_pin="Y2"* ) input clk;
(*chip_pin="M23"* ) input reset;
(*chip_pin="G18, F22, E17, L26, L25, J22, H22"* ) output [6:0] hex0;
(*chip_pin="M24, Y22, W21, W22, W25, U23, U24"* ) output [6:0] hex1;
(*chip_pin="AA25, AA26, Y25, W26, Y26, W27, W28"* ) output [6:0] hex2;
(*chip_pin="V21, U21, AB20, AA21, AD24, AF23, Y19"* ) output [6:0] hex3;
(*chip_pin="AB19, AA19, AG21, AH21, AE19, AF19, AE18"* ) output [6:0] hex4;
(*chip_pin="AD18, AC18, AB18, AH19, AG19, AF18, AH18"* ) output [6:0] hex5;
(*chip_pin="AA17, AB16, AA16, AB17, AB15, AA15, AC17"* ) output [6:0] hex6;
(*chip_pin="AD17, AE17, AG17, AH17, AF17, AG18, AA14"* ) output [6:0] hex7;
reg [35:0] num;
reg [3:0] state;
parameter s0=4'b0000, s1=4'b0001, s2=4'b0010, s3=4'b0011, s4=4'b0100, s5=4'b0101, s6=4'b0110, s7=4'b0111, s8=4'b1000;

always@(posedge clk)
begin
num<=(num+1) % 50000000;
if (num==1)
begin
case (state)
s0: state<=s1;
s1: state<=s2;
s2: state<=s3;
s3: state<=s4;
s4: state<=s5;
s5: state<=s6;
s6: state<=s7;
s7: state<=s8;
s8: state<=s0;
endcase
end
end
show (state, hex0, hex1, hex2, hex3, hex4, hex5, hex6, hex7);
endmodule

module show (in1, hex0, hex1, hex2, hex3, hex4, hex5, hex6, hex7);
input [3:0] in1;
output reg [6:0] hex0;
output reg [6:0] hex1;
output reg [6:0] hex2;
output reg [6:0] hex3;
output reg [6:0] hex4;
output reg [6:0] hex5;
output reg [6:0] hex6;
output reg [6:0] hex7;
always@(in1)
begin
case (in1)
4'b0000: begin hex0<=7'b0000001; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0001: begin hex0<=7'b1001111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0010: begin hex1<=7'b0010010; hex0<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0011: begin hex2<=7'b0000110; hex0<=7'b1111111; hex1<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0100: begin hex3<=7'b1001100; hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0101: begin hex4<=7'b0100100; hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0110: begin hex5<=7'b0100000; hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
4'b0111: begin hex6<=7'b0001111; hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex7<=7'b1111111;
4'b1000: begin hex7<=7'b0000000; hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111;
default: begin hex0<=7'b1111111; hex1<=7'b1111111; hex2<=7'b1111111; hex3<=7'b1111111; hex4<=7'b1111111; hex5<=7'b1111111; hex6<=7'b1111111; hex7<=7'b1111111;
endcase
end
endmodule
```

拓展题: 任务四、在 DE2-155 开发板上实现一个路口交通灯 30 秒倒计时器, 采用两个七段数码管显示倒计时结果。

实验原理: 核心就是使用一个分频器和一个显示模块, 没有什么困难

的地方。

代码：

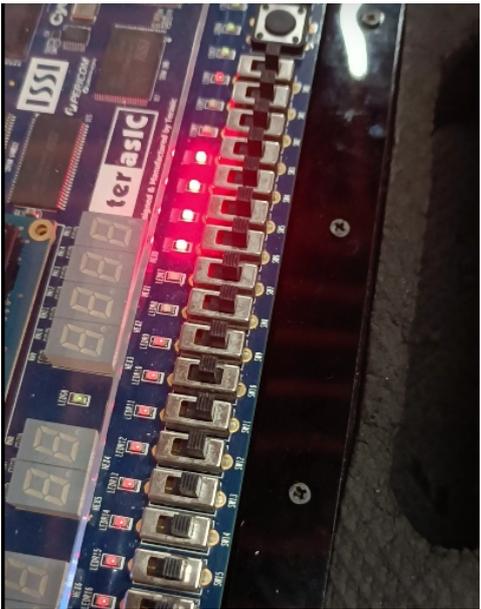
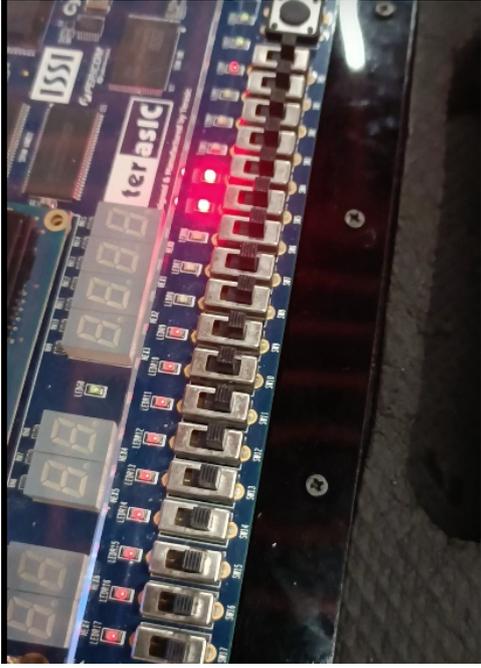
```
1 module task4(clk,hex0,hex1);
2 (*chip_pin="Y2")input clk;
3 (*chip_pin="H22,J22,L25,L26,E17,F22,G18")output [6:0]hex0;
4 (*chip_pin="U24,U23,W25,W22,W21,Y22,M24")output [6:0]hex1;
5 reg [35:0]num;
6 reg [4:0] sec;
7 always@(posedge clk)
8 begin
9     num<=(num+1)%50000000;
10    if(num==1)
11    begin
12        if(sec==0)sec<=30;
13        else sec<=sec-1;
14    end
15 end
16 show(sec%10,hex0);
17 show(sec/10,hex1);
18 endmodule
```

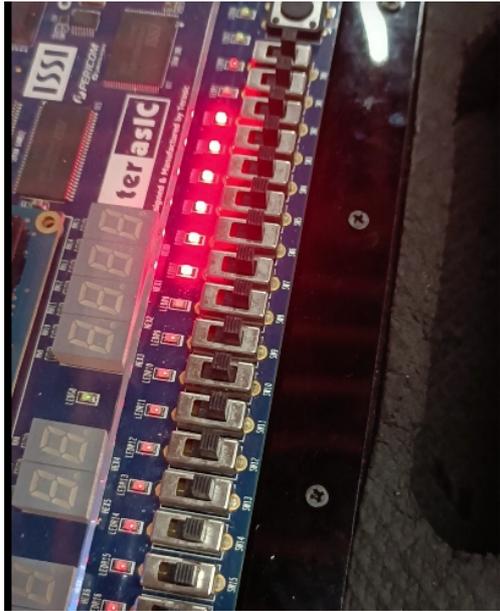
```
1 module show(in1,out);
2 input [4:0] in1;
3 output reg[6:0] out;
4 always@(in1)
5 begin
6     case(in1)
7         5'b00000:out<=7'b1000000;
8         5'b00001:out<=7'b1111001;
9         5'b00010:out<=7'b0100100;
10        5'b00011:out<=7'b0110000;
11        5'b00100:out<=7'b0011001;
12        5'b00101:out<=7'b0010010;
13        5'b00110:out<=7'b0000010;
14        5'b00111:out<=7'b1111000;
15        5'b01000:out<=7'b0000000;
16        5'b01001:out<=7'b0010000;
17        default: out<=7'b1111111;
18    endcase;
19 end
20 endmodule
```

## 四、实验结果与分析

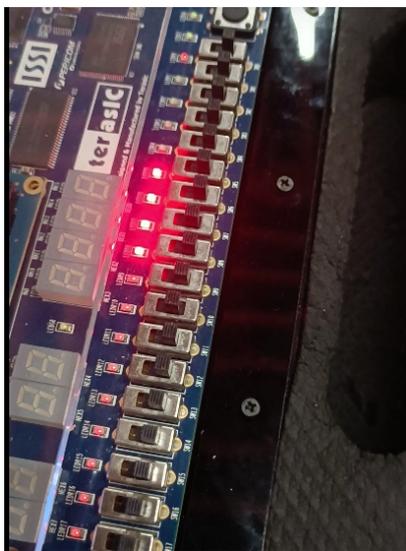
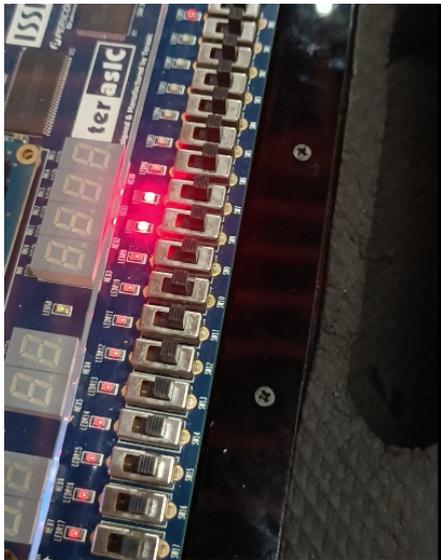
### 任务一、8 路彩灯控制器设计与实现

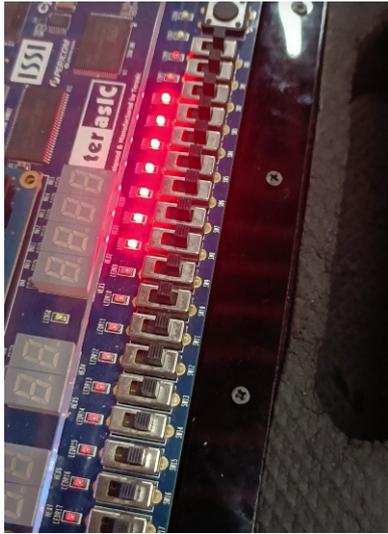
开发板调试结果：从中间开始向两边亮





从左到右逐个亮

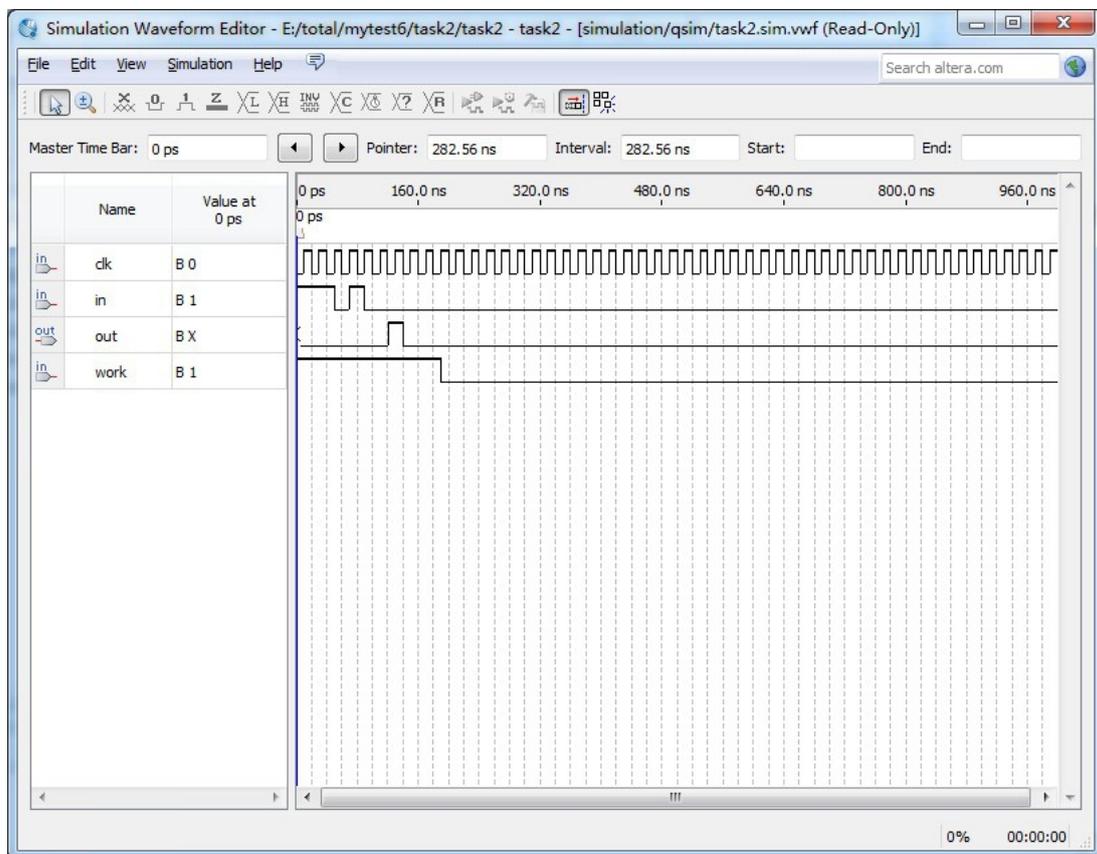




## 任务二、1101 序列检测器设计与仿真

由于设计的亮的时间有些短，很难拍到开发板上的效果，故未拍开发板，仅进行了波形仿真和 **signal tap** 仿真。

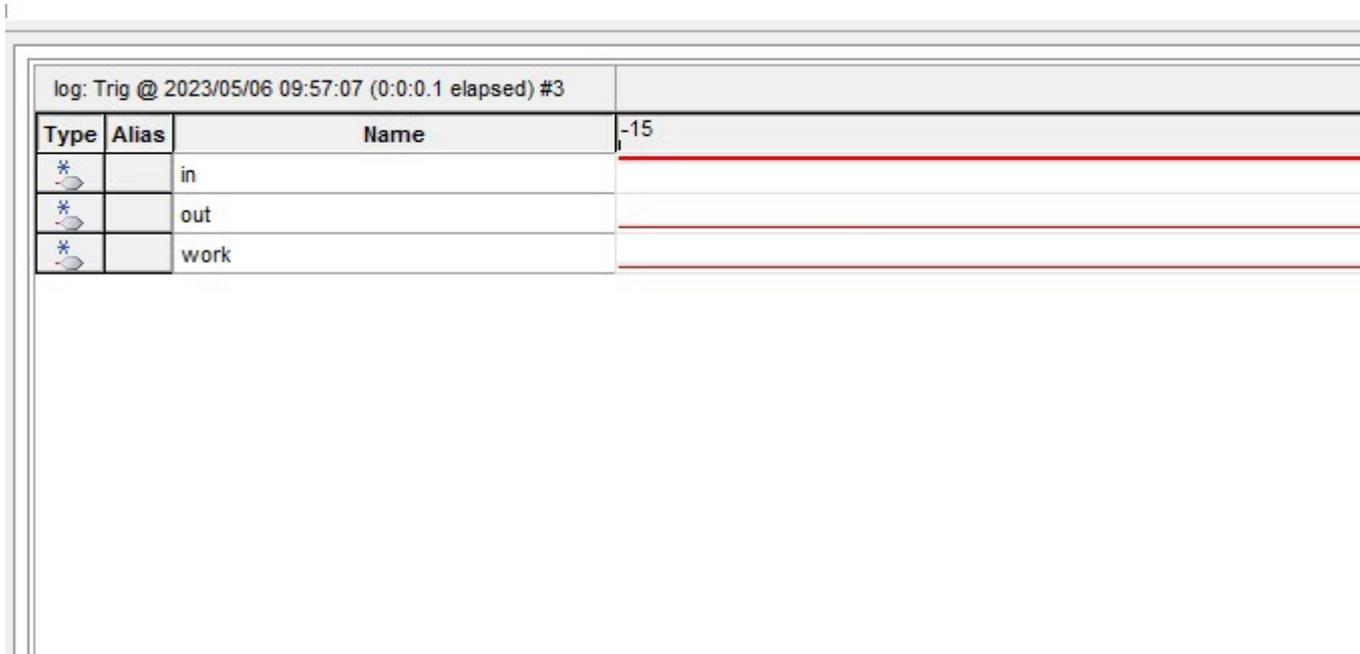
波形仿真：



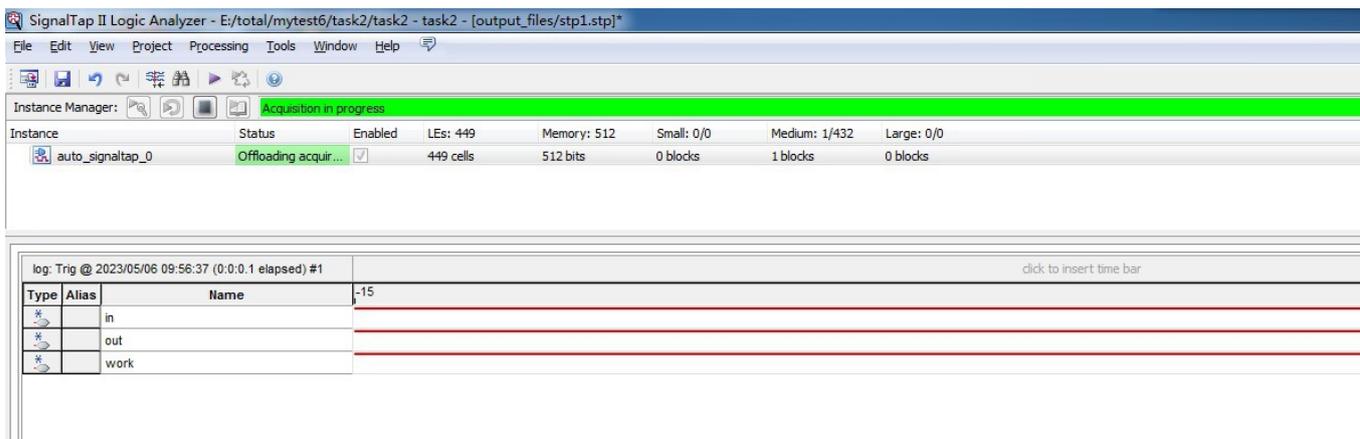
可以看出 in 在连续输入 1101 时 out 的输出变为了 1 说明有限状态机设计成功

## Signaltap 仿真:

未达到条件时:



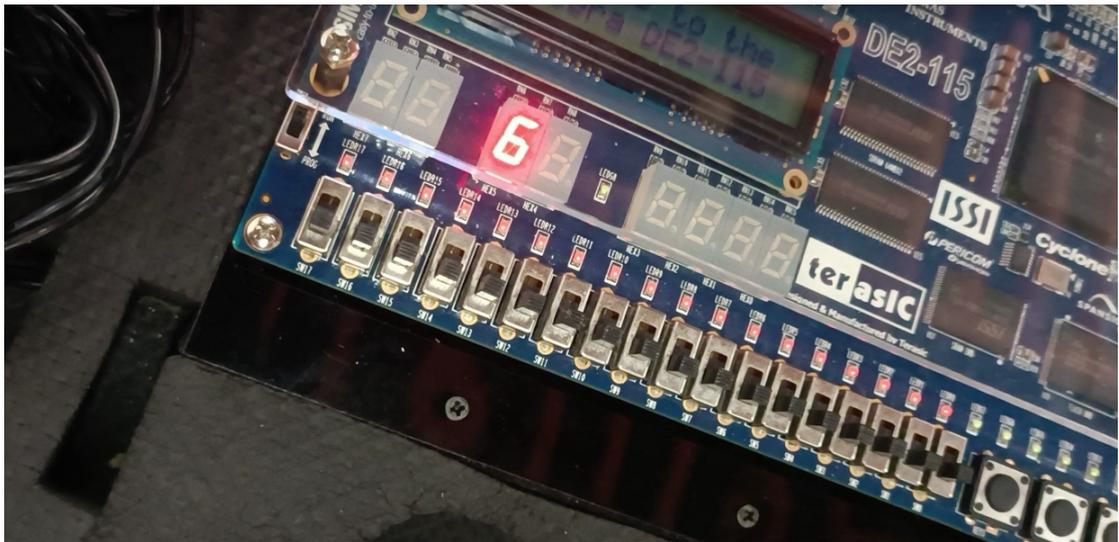
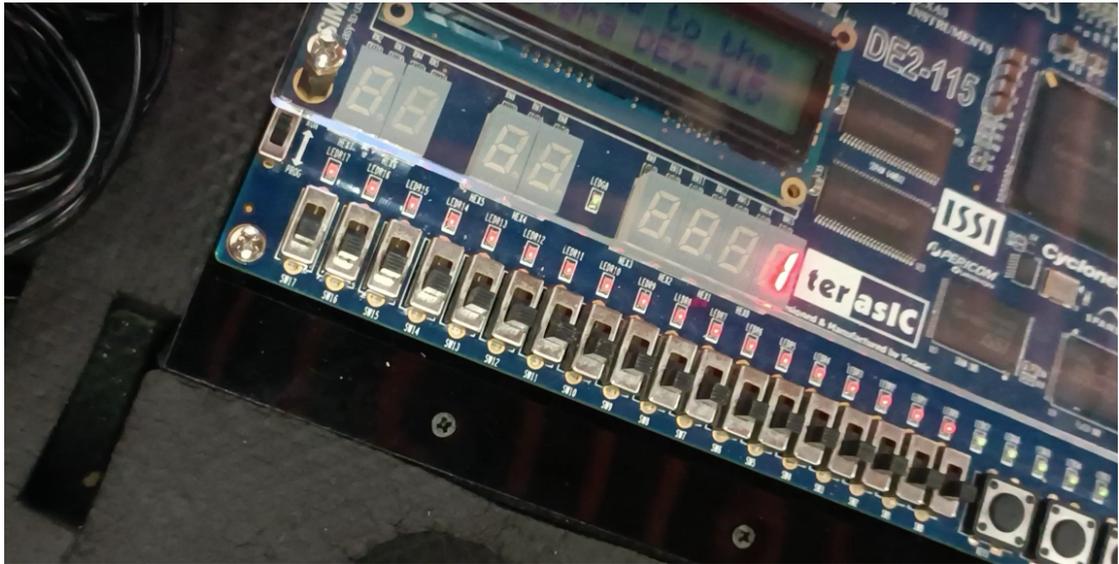
达到条件时

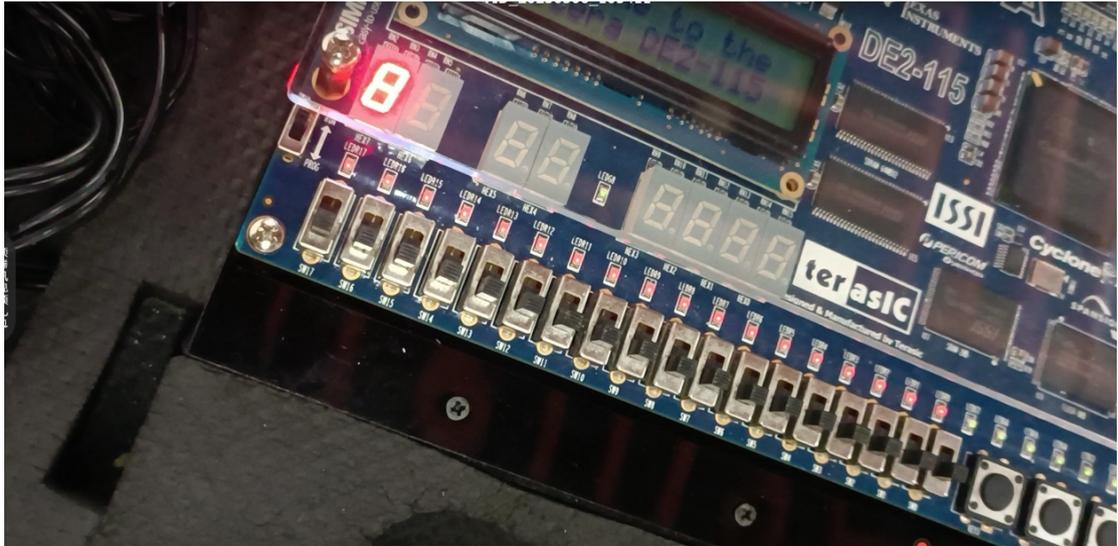


可以看得到在手动控制开关输入 1101 时从 0 变成了 1

## 任务三、实现循环扫描显示的七段数码管译码控制电路的设计

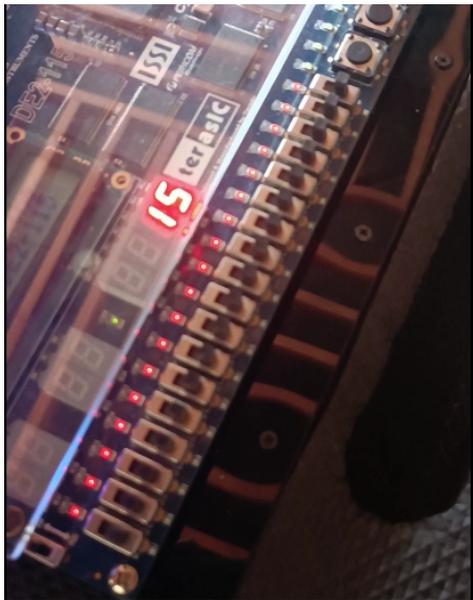
开发板调试结果:

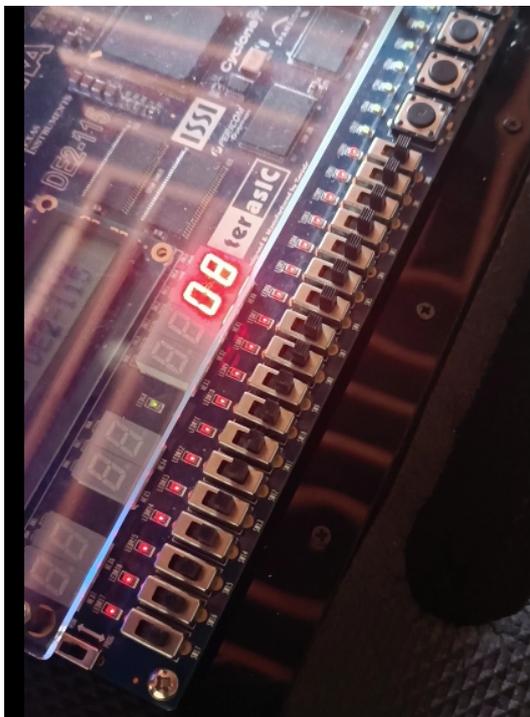
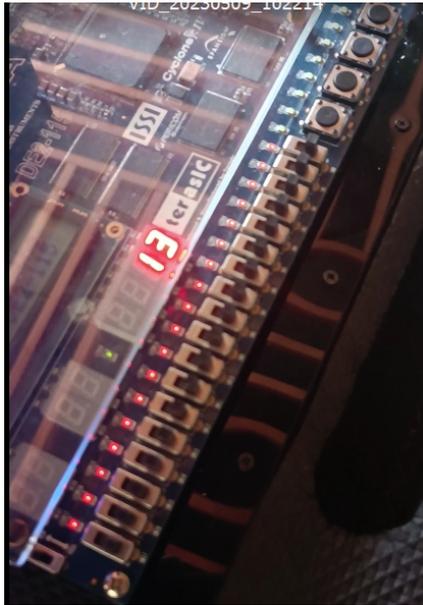




因状态太多，仅列举这几个示例，现场可以看出运行结果与预期的结果相同。故认为设计成功。

**拓展题：任务四、在 DE2-155 开发板上实现一个路口交通灯 30 秒**  
**开发板调试结果：**





可以看出与设计要求相符，从 30 秒倒数，倒数至 0 秒时从 30 再开始倒数。

## 五、实验体会与讨论

本次实验题目并不难但是也遇到了一些坑人的小错误，比如说写函数的参数时，不小心重复写了一个参数导致参数多一个，但是正常调用，编译也不会报错，下载到板子上也不会出错，但是实验结果就是有一些奇奇怪怪的问题，看了好久才发现了这个错误。

任务一的时候本来写的时双过程，总是出一些奇怪的问题，然后改未单过程描述后就不会出问题了，可能有一些问题在双过程时没有发现。所以在此后的几个任务中用的单过程描述最多，也最熟练了。

本次实验熟练了对有限状态机的设计，也回顾了分频器的设计与使用，除发生一些小错误外，整体比较流畅的写了下来，感觉自己写verilog程序的熟练度不断的提高，也是逐渐有了这个语言的思维。