电工电子实验报告

课程名称：	电工电子实验（二）
实验项目：	可编程器件的应用及数字系统设计（状态机） 小型数字系统设计（一）

学　　院：	贝尔英才学院
班　　级：	
学　　号：	
姓　　名：	
学　　期：	2022-2023学年第1学期

可编程器件的应用及数字系统设计（状态机）

一、实验目的

1.使用ISE软件完成时序逻辑电路的设计输入并仿真。
2.掌握Testbench中时序逻辑测试文件的写法。
3.下载并测试实现的逻辑功能。

二、实验仪器

ISE Design Suite 14.7

三、实验原理

Mealy型有限状态机的状态转移表

现态	次态		输出	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	1
S1	S1	S2	0	1
S2	S2	S3	0	1
S3	S3	S0	1	0

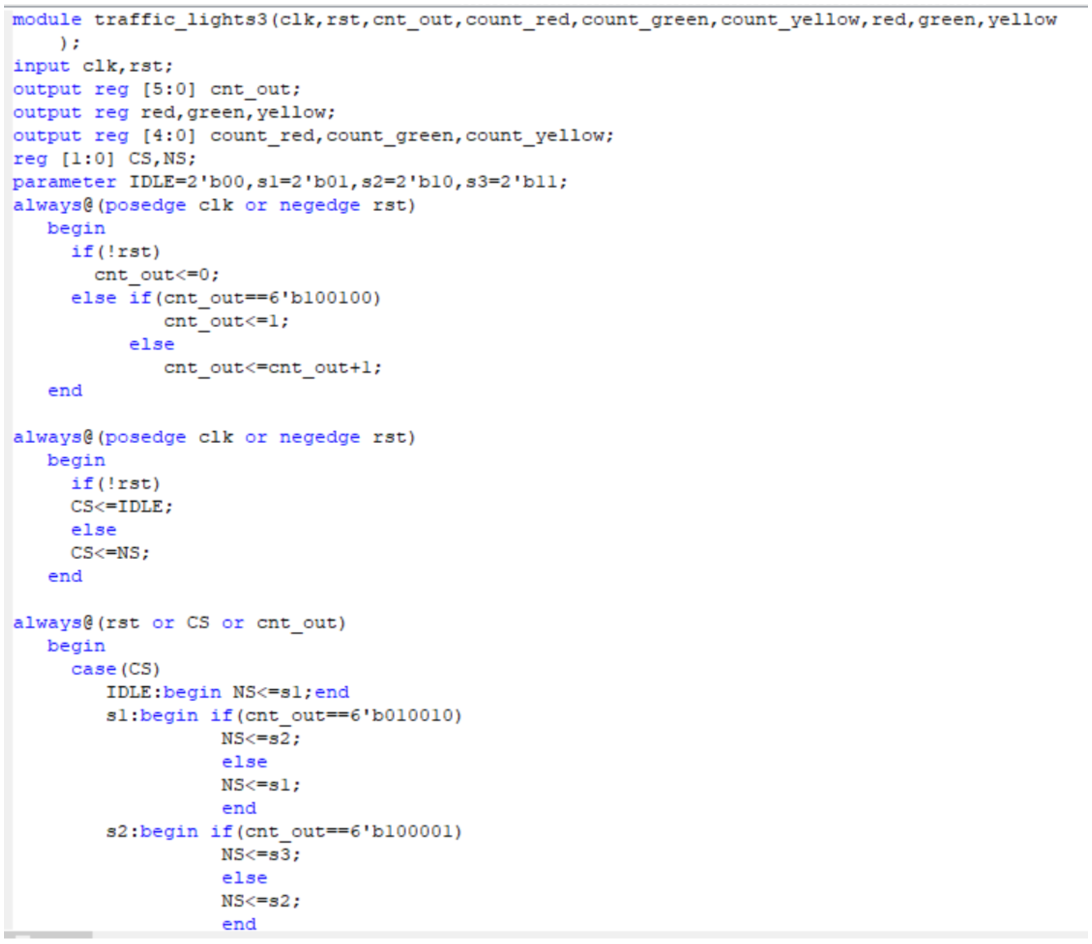
四、实验内容

1.通过状态机方法重新设计交通灯电路，并讨论有限状态机的不同实现方式和编码方式的特点(参考5.5.5节)，完成设计模块，Testbench，并保存仿真波形，Testbench能够测试到电路设计要求的所有功能，下载到FPGA，完成硬件调测并实现电路功能。
2.根据交通灯的设计方法，设计一个PWM信号方波发生器，要求如下：
（1）输入时钟脉冲频率为100kHz。
（2）产生PWM信号的周期为1ms
（3）PWM信号占空比为10%~90%可变。
（4）按键控制占空比步进，1%连续可调。
完成设计模块，Testbench，并保存仿真波形，Testbench能够测试到电路设计要求的所有功能，下载到FPGA，完成硬件调测并实现电路功能。
3.设计一组交通灯，东西南北两组灯，东西亮红灯18秒，南北亮绿灯15秒，亮黄灯3秒，然后切换东西绿灯15秒，黄灯3秒南北红灯18秒。

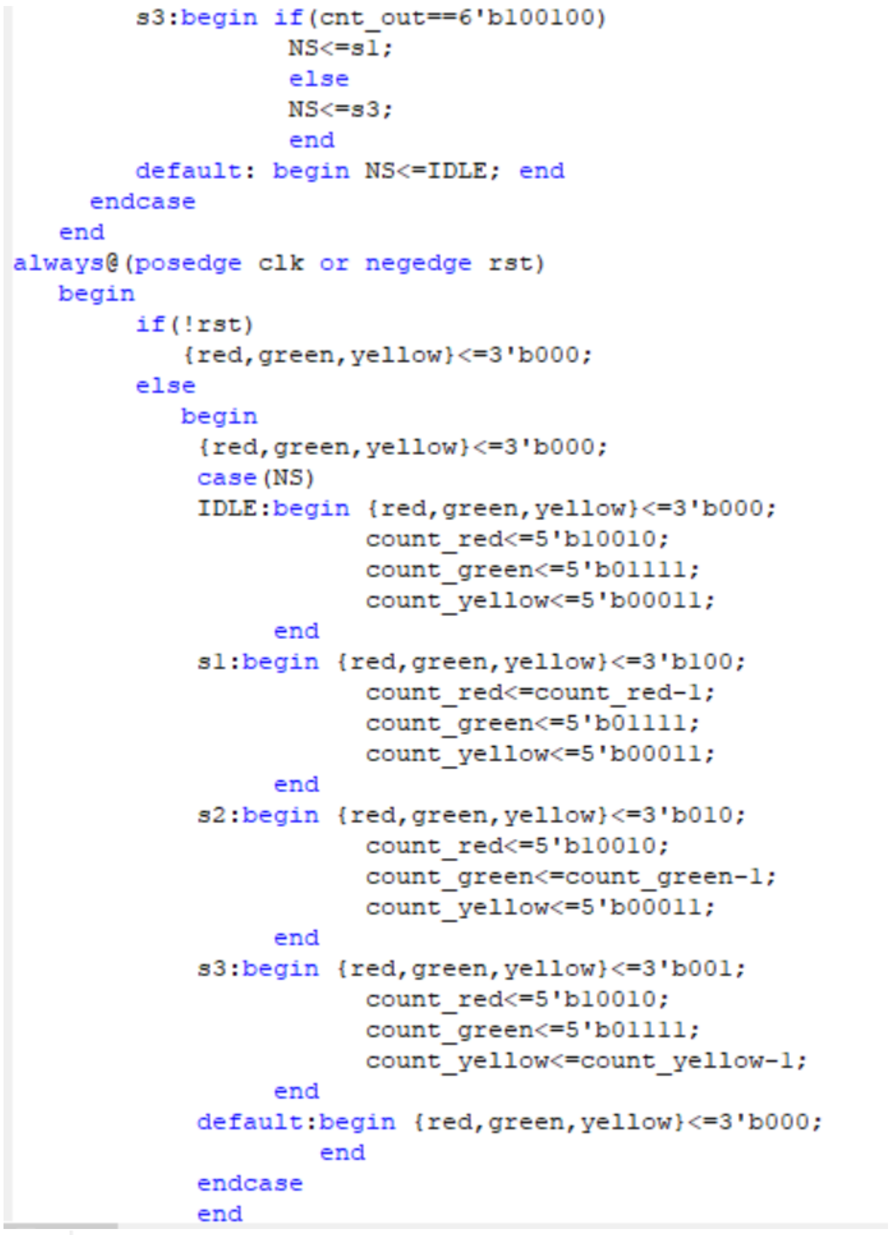
五、实验数据

1.通过状态机方法重新设计交通灯电路，并讨论有限状态机的不同实现方式和编码方式的特点(参考5.5.5节)，完成设计模块，Testbench，并保存仿真波形，Testbench能够测试到电路设计要求的所有功能，下载到FPGA，完成硬件调测并实现电路功能。

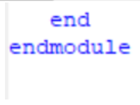
（1）设计代码



```
module traffic_lights3(clk,rst,cnt_out,count_red,count_green,count_yellow,red,green,yellow
    );
input clk,rst;
output reg [5:0] cnt_out;
output reg red,green,yellow;
output reg [4:0] count_red,count_green,count_yellow;
reg [1:0] CS,NS;
parameter IDLE=2'b00,s1=2'b01,s2=2'b10,s3=2'b11;
always@(posedge clk or negedge rst)
   begin
     if(!rst)
       cnt_out<=0;
     else if(cnt_out==6'b100100)
             cnt_out<=1;
          else
             cnt_out<=cnt_out+1;
   end

always@(posedge clk or negedge rst)
   begin
     if(!rst)
     CS<=IDLE;
     else
     CS<=NS;
   end

always@(rst or CS or cnt_out)
   begin
     case(CS)
        IDLE:begin NS<=s1;end
        s1:begin if(cnt_out==6'b010010)
                  NS<=s2;
                  else
                  NS<=s1;
                  end
        s2:begin if(cnt_out==6'b100001)
                  NS<=s3;
                  else
                  NS<=s2;
                  end
```

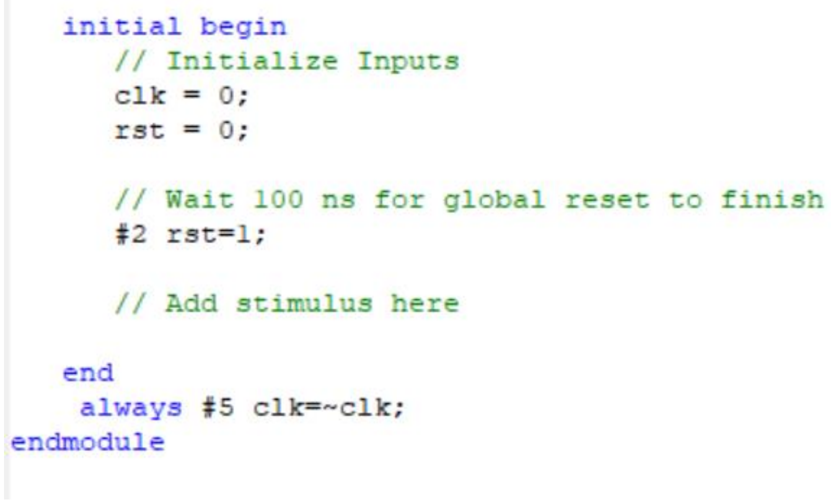


```
        s3:begin if(cnt_out==6'b100100)
                  NS<=s1;
                  else
                  NS<=s3;
                  end
        default: begin NS<=IDLE; end
      endcase
   end
always@(posedge clk or negedge rst)
   begin
        if(!rst)
           {red,green,yellow}<=3'b000;
        else
           begin
            {red,green,yellow}<=3'b000;
            case(NS)
            IDLE:begin {red,green,yellow}<=3'b000;
                       count_red<=5'b10010;
                       count_green<=5'b01111;
                       count_yellow<=5'b00011;
                 end
            s1:begin {red,green,yellow}<=3'b100;
                       count_red<=count_red-1;
                       count_green<=5'b01111;
                       count_yellow<=5'b00011;
                 end
            s2:begin {red,green,yellow}<=3'b010;
                       count_red<=5'b10010;
                       count_green<=count_green-1;
                       count_yellow<=5'b00011;
                 end
            s3:begin {red,green,yellow}<=3'b001;
                       count_red<=5'b10010;
                       count_green<=5'b01111;
                       count_yellow<=count_yellow-1;
                 end
            default:begin {red,green,yellow}<=3'b000;
                    end
            endcase
            end
```



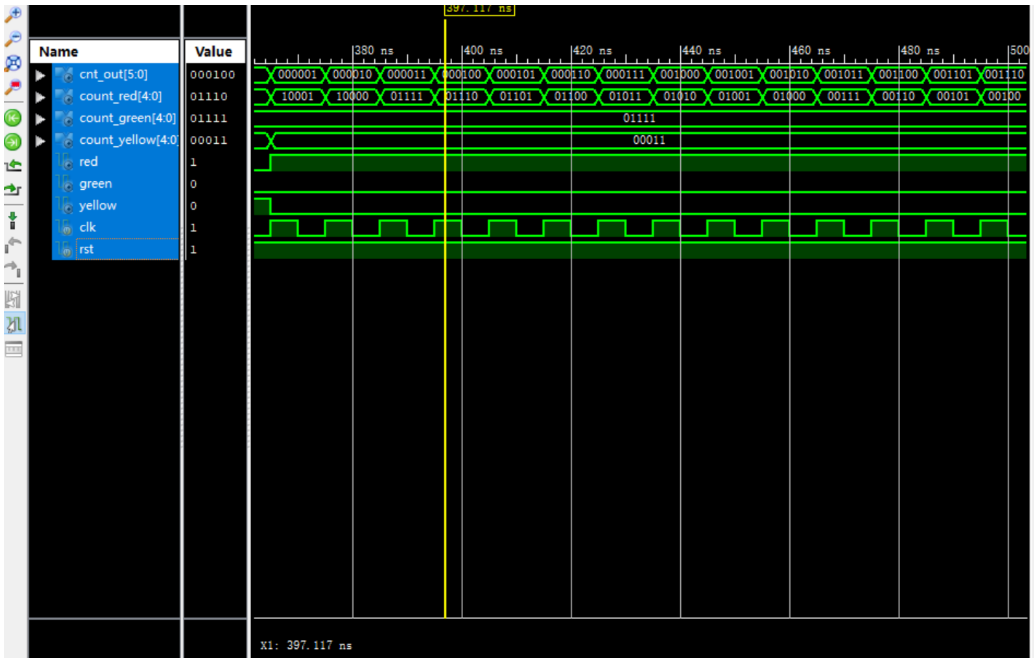
```
   end
endmodule
```

（2）测试代码

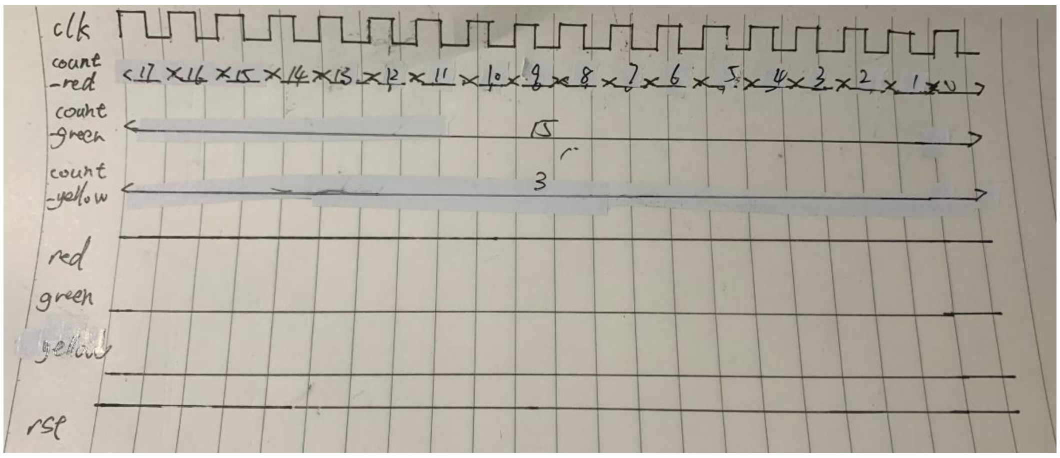


```
   initial begin
      // Initialize Inputs
      clk = 0;
      rst = 0;

      // Wait 100 ns for global reset to finish
      #2 rst=1;

      // Add stimulus here

   end
    always #5 clk=~clk;
endmodule
```

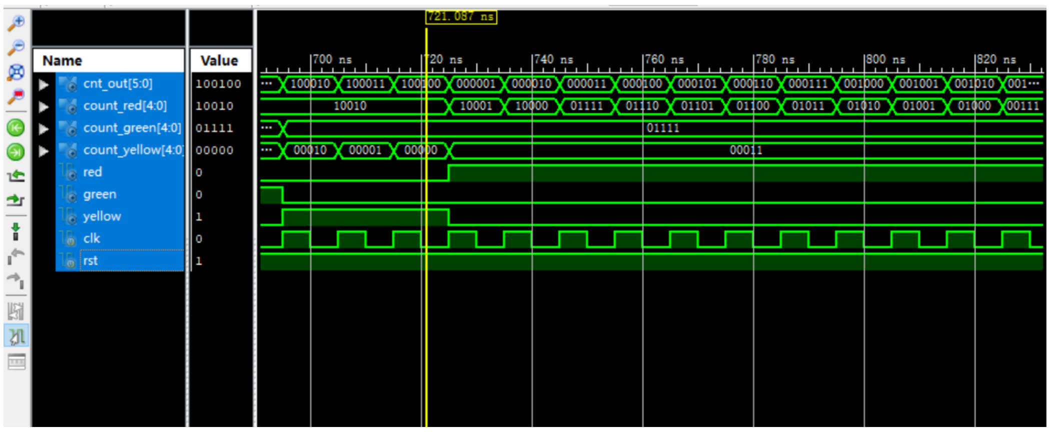
（3）仿真结果



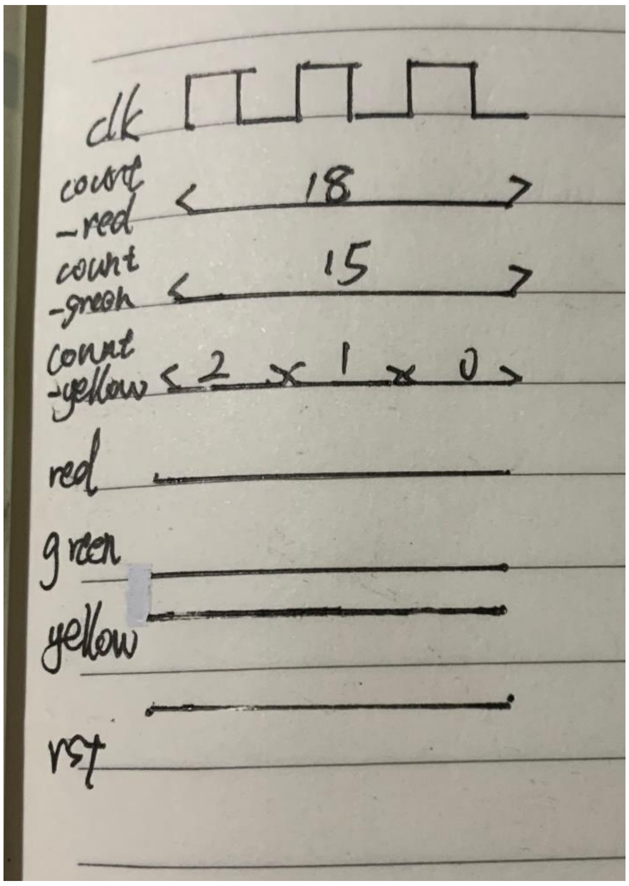
红灯倒计时仿真截图



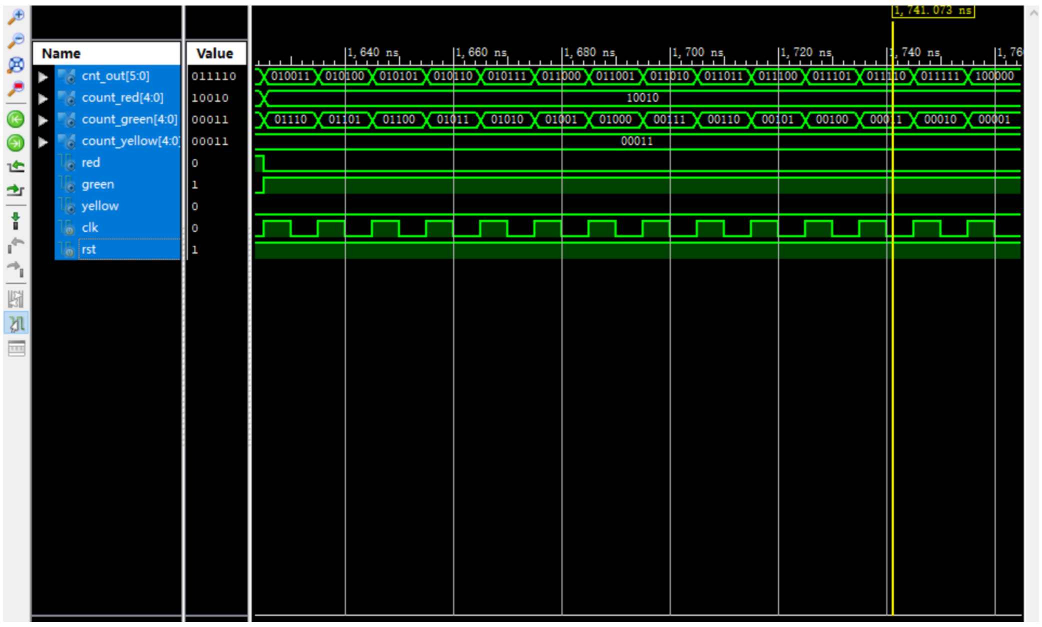
红灯倒计时手绘图



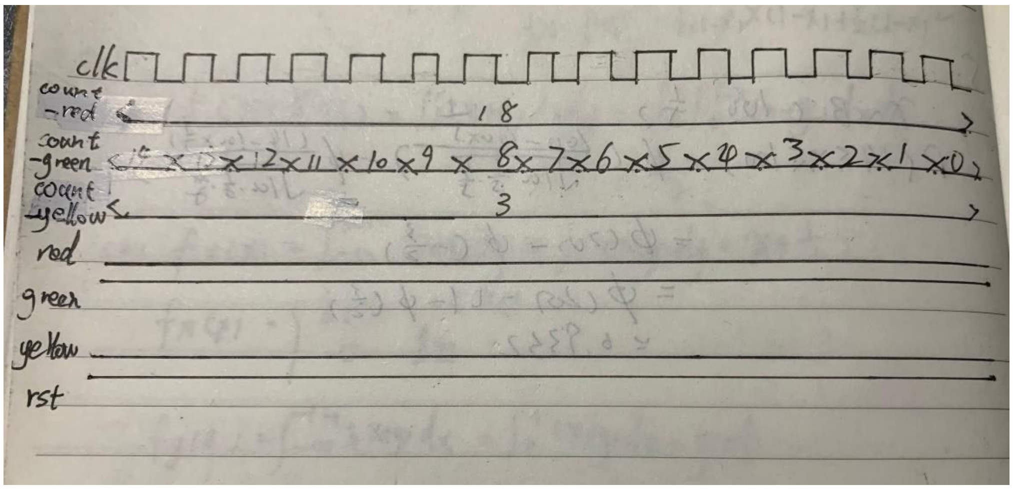
黄灯倒计时仿真截图



黄灯倒计时手绘图



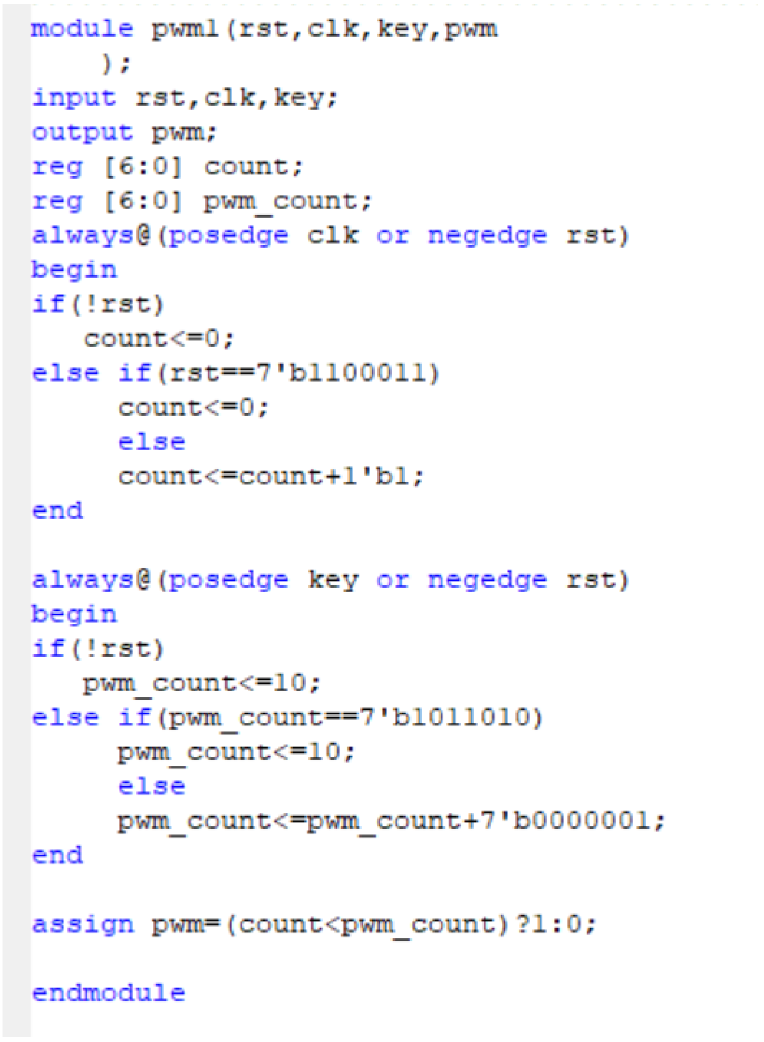
绿灯倒计时仿真截图


clk
count-red
18
count-green
14 x 13 x 12 x 11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 x 0
count-yellow
3
red
green
yellow
rst

绿灯倒计时手绘图

2.根据交通灯的设计方法，设计一个PWM信号方波发生器，要求如下：
（1）输入时钟脉冲频率为100kHz。
（2）产生PWM信号的周期为1ms
（3）PWM信号占空比为10%~90%可变。
（4）按键控制占空比步进，1%连续可调。
完成设计模块，Testbench，并保存仿真波形，Testbench能够测试到电路设计要求的所有功能，下载到FPGA，完成硬件调测并实现电路功能。

（1）设计代码

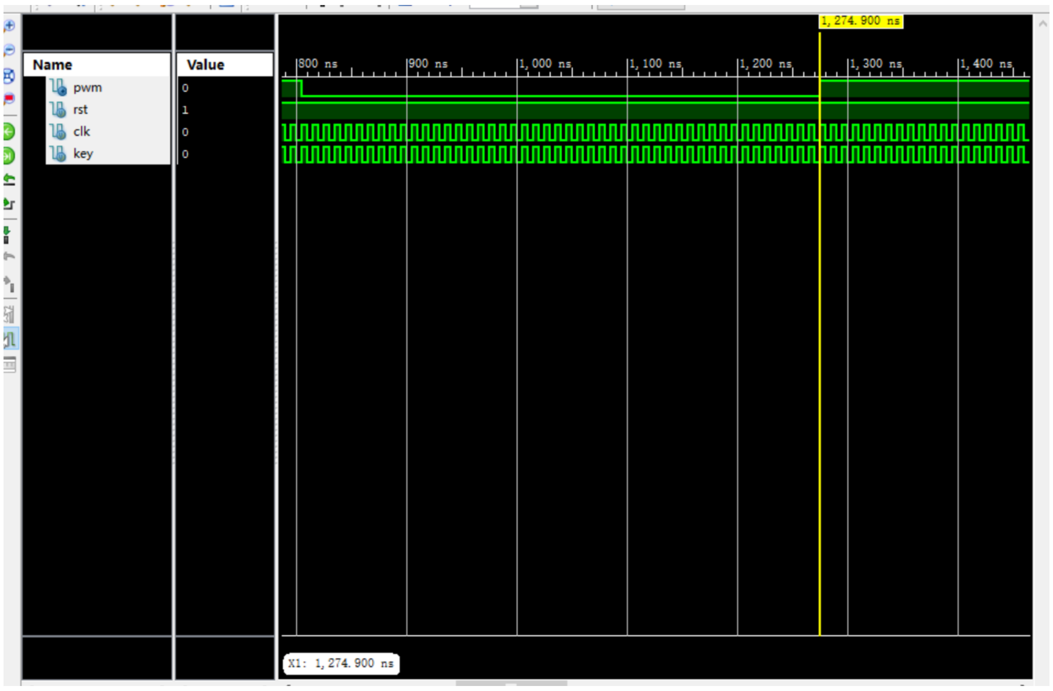


```
module pwm1(rst,clk,key,pwm
    );
input rst,clk,key;
output pwm;
reg [6:0] count;
reg [6:0] pwm_count;
always@(posedge clk or negedge rst)
begin
if(!rst)
   count<=0;
else if(rst==7'b1100011)
      count<=0;
      else
      count<=count+1'b1;
end

always@(posedge key or negedge rst)
begin
if(!rst)
   pwm_count<=10;
else if(pwm_count==7'b1011010)
      pwm_count<=10;
      else
      pwm_count<=pwm_count+7'b0000001;
end

assign pwm=(count<pwm_count)?1:0;

endmodule
```

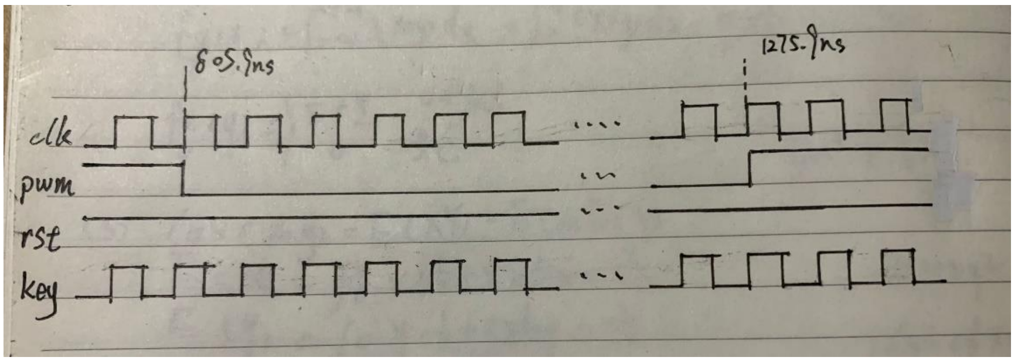
（2）测试代码

```
    initial begin
        // Initialize Inputs
        rst = 0;
        clk = 0;
        key = 0;

        // Wait 100 ns for global reset to finish
        #2 rst=1;

        // Add stimulus here

    end
        always #5 clk=~clk;
        always #5 key=~key;
endmodule
```

（3）仿真结果（截图及手绘）



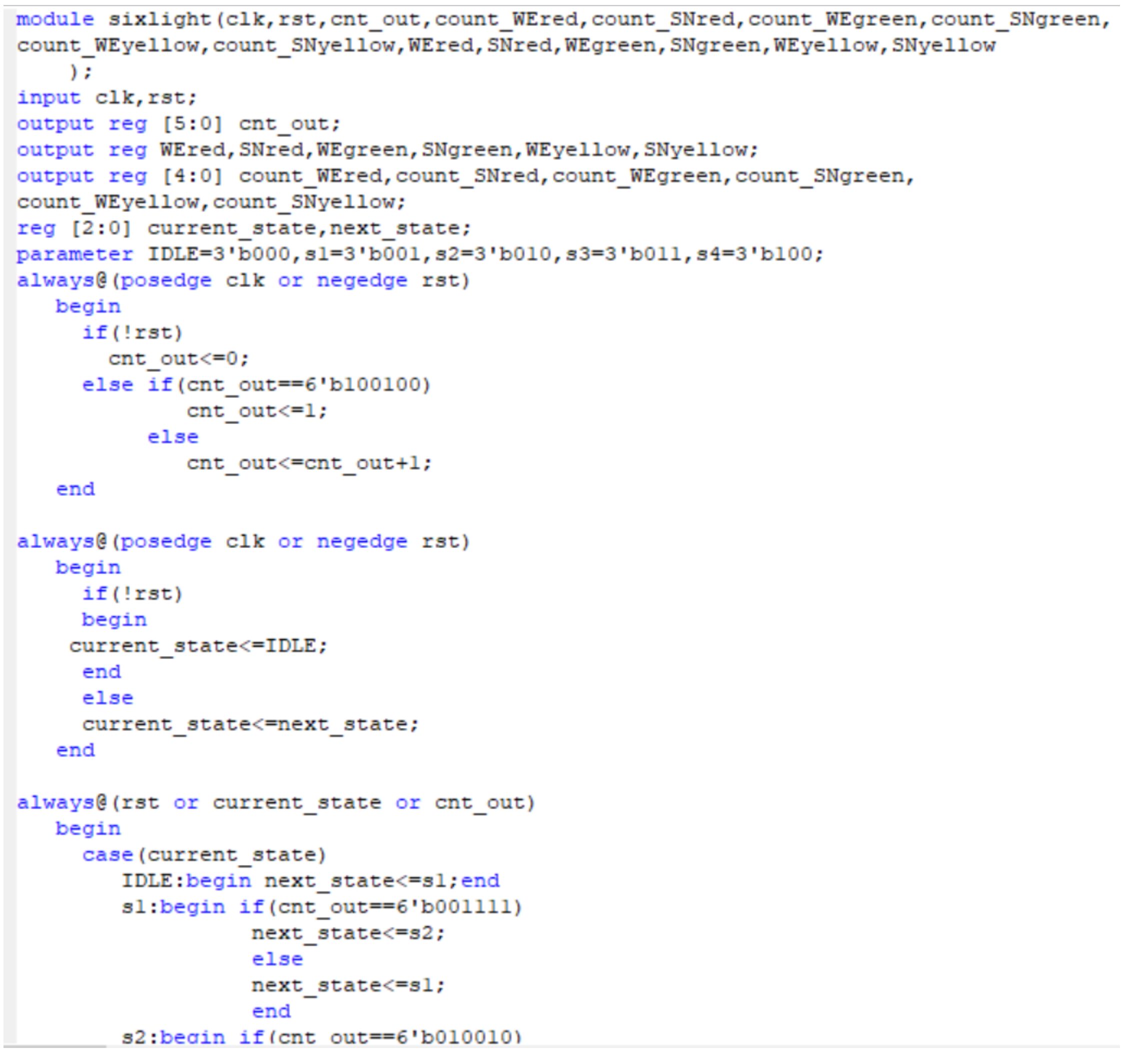
Pwm仿真截图



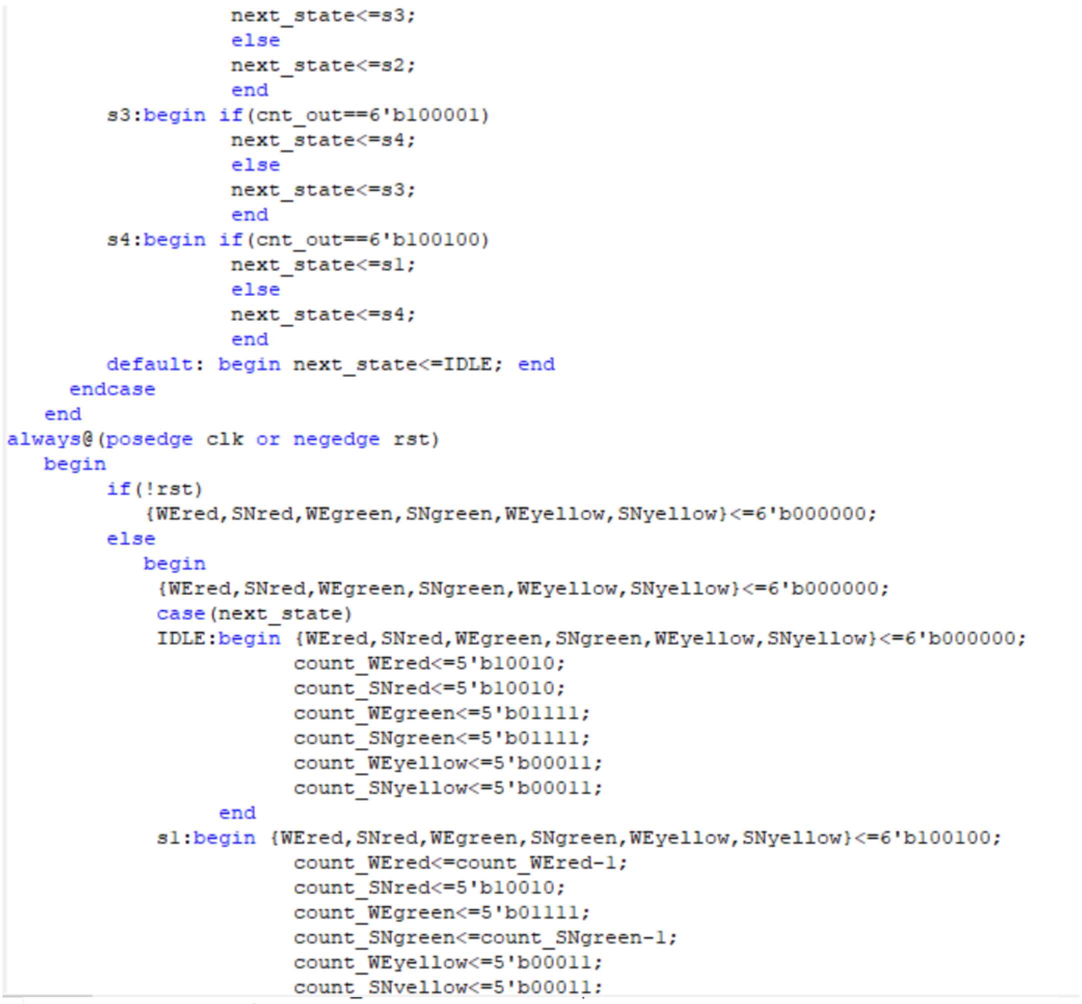
Pwm手绘波形图

3.设计一组交通灯，东西南北两组灯，东西亮红灯18秒，南北亮绿灯15秒，亮黄灯3秒，然后切换东西绿灯15秒，黄灯3秒南北红灯18秒。

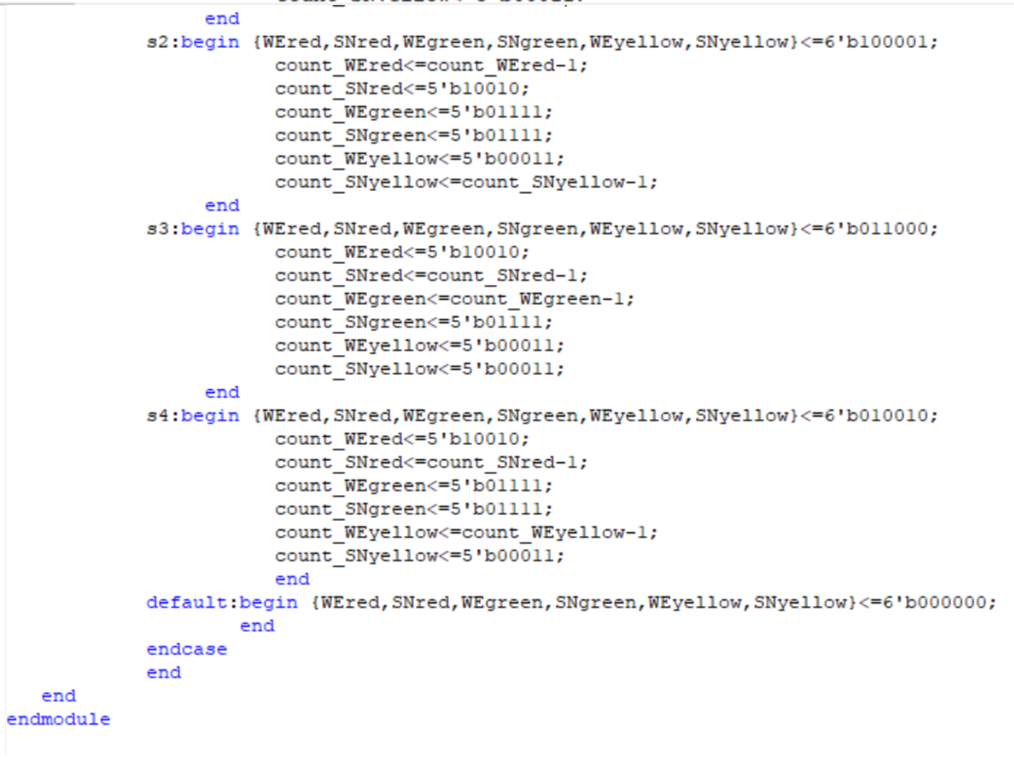
（1）设计代码



```
module sixlight(clk,rst,cnt_out,count_WEred,count_SNred,count_WEgreen,count_SNgreen,
count_WEyellow,count_SNyellow,WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow
    );
input clk,rst;
output reg [5:0] cnt_out;
output reg WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow;
output reg [4:0] count_WEred,count_SNred,count_WEgreen,count_SNgreen,
count_WEyellow,count_SNyellow;
reg [2:0] current_state,next_state;
parameter IDLE=3'b000,s1=3'b001,s2=3'b010,s3=3'b011,s4=3'b100;
always@(posedge clk or negedge rst)
   begin
     if(!rst)
       cnt_out<=0;
     else if(cnt_out==6'b100100)
             cnt_out<=1;
          else
             cnt_out<=cnt_out+1;
   end

always@(posedge clk or negedge rst)
   begin
     if(!rst)
     begin
    current_state<=IDLE;
     end
     else
     current_state<=next_state;
   end

always@(rst or current_state or cnt_out)
   begin
     case(current_state)
        IDLE:begin next_state<=s1;end
        s1:begin if(cnt_out==6'b001111)
                  next_state<=s2;
                  else
                  next_state<=s1;
                  end
        s2:begin if(cnt_out==6'b010010)
```

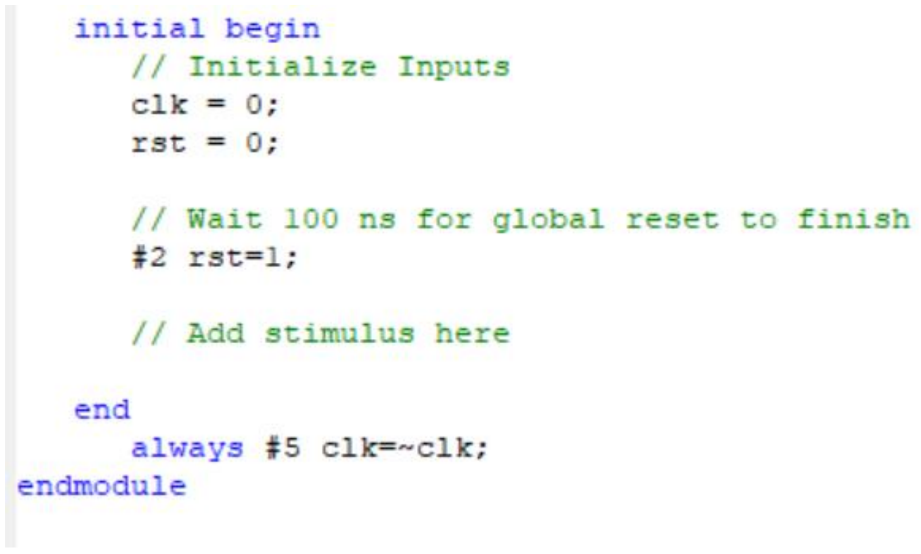


```
                  next_state<=s3;
                  else
                  next_state<=s2;
                  end
        s3:begin if(cnt_out==6'b100001)
                  next_state<=s4;
                  else
                  next_state<=s3;
                  end
        s4:begin if(cnt_out==6'b100100)
                  next_state<=s1;
                  else
                  next_state<=s4;
                  end
        default: begin next_state<=IDLE; end
     endcase
   end
always@(posedge clk or negedge rst)
   begin
        if(!rst)
           {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b000000;
        else
           begin
            {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b000000;
            case(next_state)
            IDLE:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b000000;
                       count_WEred<=5'b10010;
                       count_SNred<=5'b10010;
                       count_WEgreen<=5'b01111;
                       count_SNgreen<=5'b01111;
                       count_WEyellow<=5'b00011;
                       count_SNyellow<=5'b00011;
                 end
            s1:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b100100;
                       count_WEred<=count_WEred-1;
                       count_SNred<=5'b10010;
                       count_WEgreen<=5'b01111;
                       count_SNgreen<=count_SNgreen-1;
                       count_WEyellow<=5'b00011;
                       count_SNyellow<=5'b00011;
```



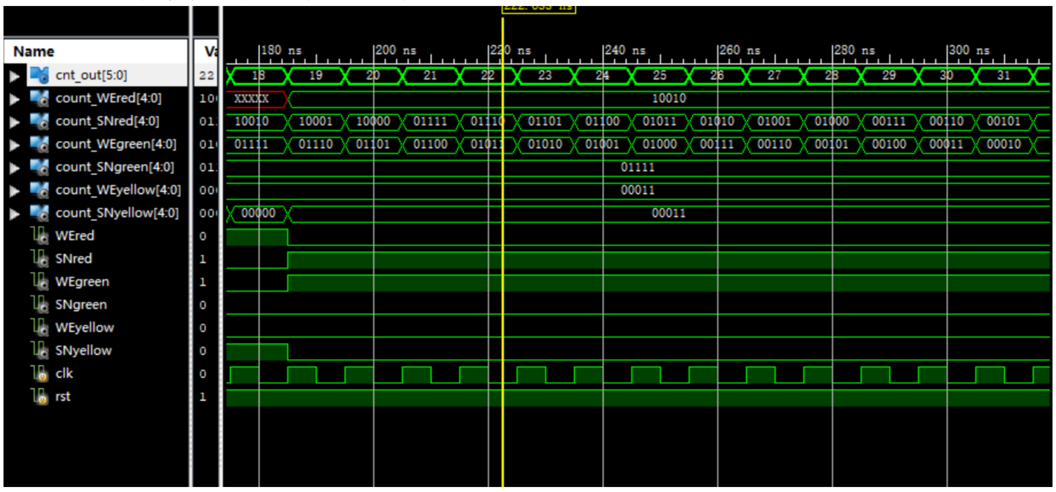
```
                 end
            s2:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b100001;
                       count_WEred<=count_WEred-1;
                       count_SNred<=5'b10010;
                       count_WEgreen<=5'b01111;
                       count_SNgreen<=5'b01111;
                       count_WEyellow<=5'b00011;
                       count_SNyellow<=count_SNyellow-1;
                 end
            s3:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b011000;
                       count_WEred<=5'b10010;
                       count_SNred<=count_SNred-1;
                       count_WEgreen<=count_WEgreen-1;
                       count_SNgreen<=5'b01111;
                       count_WEyellow<=5'b00011;
                       count_SNyellow<=5'b00011;
                 end
            s4:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b010010;
                       count_WEred<=5'b10010;
                       count_SNred<=count_SNred-1;
                       count_WEgreen<=5'b01111;
                       count_SNgreen<=5'b01111;
                       count_WEyellow<=count_WEyellow-1;
                       count_SNyellow<=5'b00011;
                       end
            default:begin {WEred,SNred,WEgreen,SNgreen,WEyellow,SNyellow}<=6'b000000;
                    end
            endcase
            end
   end
endmodule
```

（2）测试代码

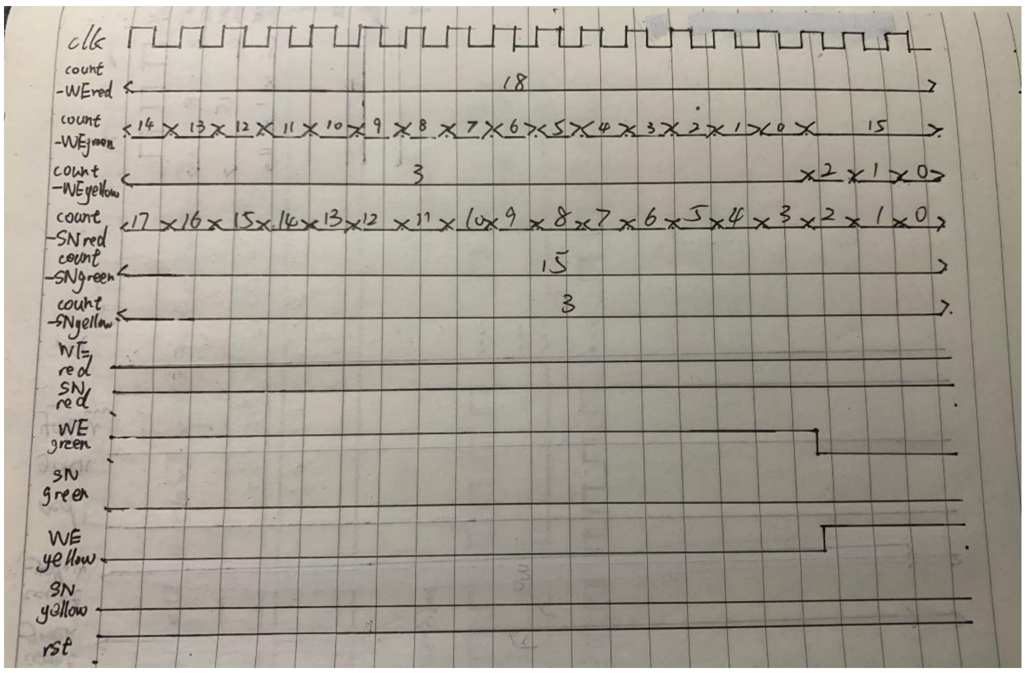


```
    initial begin
        // Initialize Inputs
        clk = 0;
        rst = 0;

        // Wait 100 ns for global reset to finish
        #2 rst=1;

        // Add stimulus here

    end
        always #5 clk=~clk;
endmodule
```

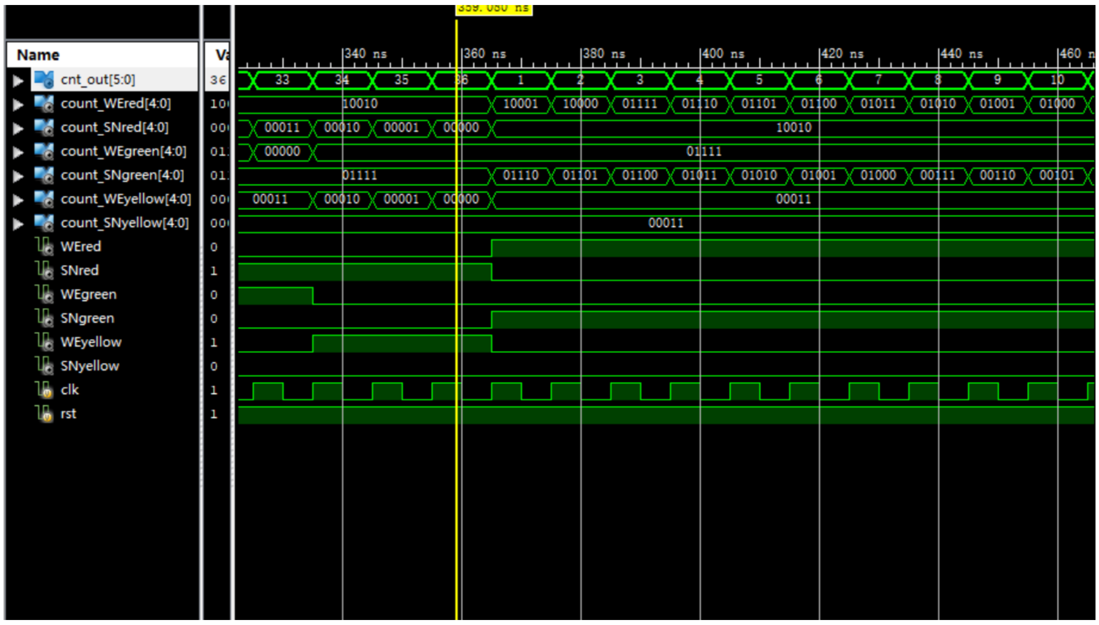
（3）仿真结果（截图及手绘）



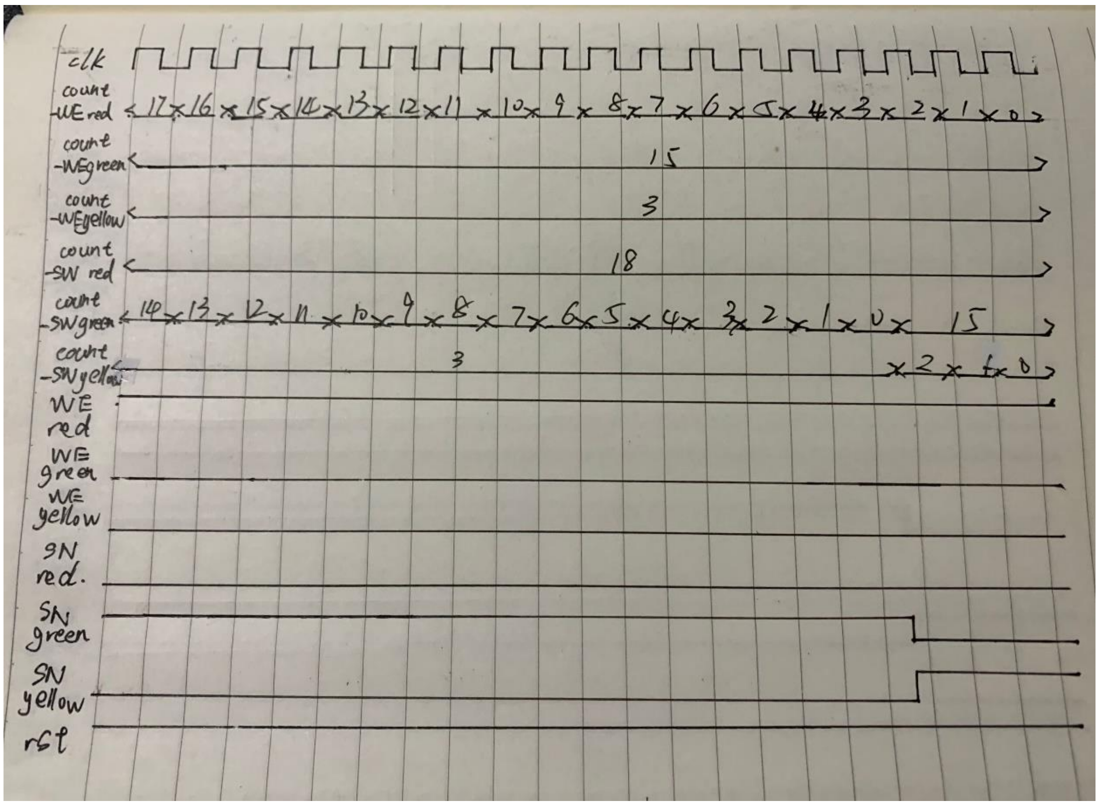
东西方向绿灯黄灯及南北方向红灯倒计时仿真截图



东西方向绿灯黄灯及南北方向红灯倒计时手绘波形图



东西方向红灯及南北方向绿灯黄灯倒计时仿真截图



东西方向红灯及南北方向绿灯黄灯倒计时手绘波形图

六、实验小结

对三段式有限状态机的设计有了具体的了解，掌握三段式有限状态机中各个always模块分别起的作用，了解了如何运用多个always模块分块对任务进行设计，对Verilog语言的运用有了更深的理解。掌握了Testbench中时序逻辑测试文件的写法。