综合实验报告

2021 年 1 月 6 日　　成绩：_____

姓名	刘合	学号		班级	
专业	计算机科学与技术		课程名称	数字电路课程设计	
任课老师		指导老师		机位号	
实验序号	综合实验	实验名称	交通灯设计		
实验时间	2021.1.6	实验地点		实验设备号	

一、实验目的与要求

1. 实验目的：、

（1）学习数码管动态扫描方法，熟悉模块调用的方法，学习利用状态机来实现各个状态之间转换的编程方法，锻炼编程设计数字系统的能力；

（2）掌握灵活运用 Verilog HDL 语言进行各种描述与建模的技巧和方法。

2. 实验要求:

(1)使用合适的或你喜欢的描述方式，编程实现规定功能的交通灯系统。先编写底层交通灯控制模块，修改实验 18 的二进制转 BCD 码模块和基本数码管显示模块，再编写顶层的控制模块。

(2)课前任务:在 Xilinx Vivado 上完成创建工程、编辑程序源代码、编译、综合、仿真、验证,确保逻辑正确性。

(3)实验室任务：根据源程序设计和 HCS-A01 实验板的结构，配置管脚。CLK 连接到 E3（100MHz 时钟输入）。数码管显示 A[3:0]和 seg[7:0]连接到表 6.2 所示的管脚上。启动键 start,系统复位键 CLR，暂停键连接到 HCS-A01 实验板的按钮，主干道阻塞不通行 stopa 和次干道阻塞不通行 stopb，连接到 HCS-A01 实验板的拨码开关。

(4)生成*.bit 文件,下载到 HCS-A01 实验板上的 FPGA 中

(5)按动按键，完成板级验证。

(6)撰写实验报告:含实验程序代码、激励代码及其仿真波形、综合得到的电路图、

引脚配置、实验结果分析,以及你对本实验的“思考与探索”部分所作的思考与探索。

3. 实验内容与原理：

设计一个十字路口的交通灯系统。

系统输入：时钟 CLK、系统复位 CLR、交通灯正常启动 start、主干道阻塞不通行 stopa，次干道阻塞不通行 stopb。系统暂停 pause 按钮用于暂停系统，便于观察实验结果。

系统输出：在数码管（高 2 片）上显示主干道读秒倒计时，在数码管（低 2 片）上显示次干道读秒倒计时，6 个 LED 灯分别显示主干道和次干道的红绿黄灯。

系统的运行规则是：

（1）按下复位按钮，主干道和次干道的红灯皆亮，且主干道和次干道都不显示读秒，4 个数码管显示“- - - -”;

（2）当主干道阻塞时，主干道和次干道都不显示读秒，数码管无显示，主干道的红灯亮，次干道绿灯长亮；

（3）当次干道阻塞时，主干道和次干道都不显示读秒，数码管无显示，次干道的红灯亮，主干道绿灯长亮；

（4）当按下启动按钮，系统启动；

（5）交通灯正常工作时，主干道和次干道都显示读秒，系统在四个状态之间切换，这 4 个状态分别是：

state1：主干道通行次干道禁行，主干道绿灯亮，次干道红灯亮，时长 35 秒；

state2：主干道黄灯亮，次干道红灯亮，时长 5 秒；

state3：次干道通行主干道禁行。干道红灯亮，次干道绿灯亮，时长 25 秒；

state4：主干道红灯亮，次干道黄灯亮，时长 5 秒。

状态	主干道	次干道	时间（秒）
state1	绿灯亮	红灯亮	35
state2	黄灯亮	红灯亮	5
state3	红灯亮	绿灯亮	25

state4	红灯亮	黄灯亮	5

（6）当暂停按钮按下时，主干道和次干道计数暂停，红绿黄灯保持不变，观察结果。松开暂停按钮，系统继续运行。

二、实验设计与程序代码

1、 模块设计说明

程序包含了 4 个模块：顶层模块，交通灯逻辑控制模块，分频模块和数码管扫描显示模块。输入端口为 clk_1M,CLR,start,stopa,stopb,pause；

输出端口为 AN,seg,light。

其中顶层模块是有序调用其他模块的程序；

交通灯逻辑控制模块其中包含了交通灯所有的状态变化与控制的逻辑程序；

分频模块是为了给每个底层模块分配不同的时钟频率；

数码管扫描显示模块是为了把 BCD 码数据以十进制数字显示出来

2、 实验程序源代码及注释等

```
module jiaotongdeng(clk_1M,CLR,start,stopa,stopb,pause,AN,seg,light); //顶层模块
    input clk_1M;
    input CLR;
    input start;
    input stopa;
    input stopb;
    input pause;
    output   [3:0]AN;
    output   [7:0]seg;
    output   [5:0]light;
    wire CLK;
    wire clk;
    wire   [3:0]secah;
    wire   [3:0]secal;
    wire   [3:0]secbh;
    wire   [3:0]secbl;
```

```
    wire    [1:0]state;
    fpq2 aa(clk_1M,CLK);
    jtd cc(CLK,CLR,start,stopa,stopb,pause,light,secah,secal,secbh,secbl,state);
    fpq1 dd(clk_1M,clk);
    smgsm bb(clk,secah,secal,secbh,secbl,AN,seg);
endmodule

module jtd(
    input CLK,       //时钟
    input CLR,     //系统复位
    input start, //交通灯正常启动
    input stopa, //主干道阻塞不通行
    input stopb, //次干道阻塞不通行
    input pause, //系统暂停
    output reg [5:0]light, //前三位主干道的三盏 红 黄 绿 LED 灯  后三位次干道的三盏 红 黄 绿 LED 灯  从左到右
    output reg [3:0]secah,   //主干道读秒倒计时十位
    output reg [3:0]secal,   //主干道读秒倒计时个位
    output reg [3:0]secbh,   //次干道读秒倒计时十位
    output reg [3:0]secbl,   //次干道读秒倒计时个位
    output reg [1:0]state //状态机
     );
     parameter s1=2'b00,s2=2'b01,s3=2'b10,s4=2'b11;//  四个循环状态
always @(posedge CLK or posedge CLR)
begin
                //state=s1;
               if(CLR) //  复位情况控制
                       begin
                               secah<=4'b0000;              //主干道读秒倒计时十位
```

```
                secal<=4'b0000;              //主干道读秒倒计时个位
                secbh<=4'b0000;              //次干道读秒倒计时十位
                secbl<=4'b0000;              //次干道读秒倒计时个位
                state<=s1;
            end
        else if(stopa)
            begin
                state<=s3;
                 secah<=4'bzzzz;             //主干道读秒倒计时十位
                 secal<=4'bzzzz;             //主干道读秒倒计时个位
                 secbh<=4'bzzzz;             //次干道读秒倒计时十位
                 secbl<=4'bzzzz;             //次干道读秒倒计时个位
            end
        else if(stopb)
            begin
                state<=s1;
                 secah<=4'bzzzz;             //主干道读秒倒计时十位
                 secal<=4'bzzzz;             //主干道读秒倒计时个位
                 secbh<=4'bzzzz;             //次干道读秒倒计时十位
                 secbl<=4'bzzzz;             //次干道读秒倒计时个位
            end
        else if(pause==0&start)
            begin
                case(state) //状态机
                    s1:begin
                                secah<=4'b0011;             //主干道读秒倒计时十位 3
                                secal<=4'b0101;             //主干道读秒倒计时个位 5
                                secbh<=4'b0011;             //次干道读秒倒计时十位 3
                                secbl<=4'b0101;             //次干道读秒倒计时个位 5
if(secah!=0&secbh!=0&secal==0&secbl==0)
```

```
                begin
                    secah<=secah-1'b1;
                    secal<=4'b1001;
                    secbh<=secbh-1'b1;
                    secbl<=4'b1001;
                end
            if(secal>0&secbl>0)
              begin
                  secah<=secah;
                  secal<=secal-1'b1;
                  secbh<=secbh;
                  secbl<=secbl-1'b1;
              end
            else
              begin
                  if(secah==0&secal==0&secbh==0&secbl==0)
                      state<=s2;
              end
    end
                          s2:begin
                                  secah<=4'b0000;          //主干道读秒倒计时十位 0
                                  secal<=4'b0101;          //主干道读秒倒计时个位 5
                                  secbh<=4'b0000;          //次干道读秒倒计时十位 0
                                  secbl<=4'b0101;          //次干道读秒倒计时个位 5

if(secah!=0&secbh!=0&secal==0&secbl==0)
        begin
            secah<=secah-1'b1;
            secal<=4'b1001;
            secbh<=secbh-1'b1;
```

```
            secbl<=4'b1001;
        end
  else if(secal>0&secbl>0)
         begin
              secah<=secah;
              secal<=secal-1'b1;
              secbh<=secbh;
              secbl<=secbl-1'b1;
           end
else if(secah!=0&secbh!=0&secal==0&secbl==0)
        begin
            secah<=secah-1'b1;
            secal<=4'b1001;
            secbh<=secbh-1'b1;
            secbl<=4'b1001;
        end
else
    begin
          if(secah==0&secal==0)
              state<=s3;
    end
end
                              s3:begin
                                        secah<=4'b0010;          //主干道读秒倒计时十位 2
                                        secal<=4'b0101;          //主干道读秒倒计时个位 5
                                        secbh<=4'b0010;          //次干道读秒倒计时十位 2
                                        secbl<=4'b0101;          //次干道读秒倒计时个位 5
if(secah!=0&secbh!=0&secal==0&secbl==0)
        begin
            secah<=secah-1'b1;
```

```
                secal<=4'b1001;
                secbh<=secbh-1'b1;
                secbl<=4'b1001;
        end
  else if(secal>0&secbl>0)
            begin
                 secah<=secah;
                 secal<=secal-1'b1;
                 secbh<=secbh;
                 secbl<=secbl-1'b1;
            end
  else
       if(secah==0&secal==0)
            state<=s4;
       end
                                s4:begin
                                        secah<=4'b0000;          //主干道读秒倒计时十位 0
                                        secal<=4'b0101;          //主干道读秒倒计时个位 5
                                        secbh<=4'b0000;          //次干道读秒倒计时十位 0
                                        secbl<=4'b0101;          //次干道读秒倒计时个位 5
if(secah!=0&secbh!=0&secal==0&secbl==0)
     begin
            secah<=secah-1'b1;
            secal<=4'b1001;
            secbh<=secbh-1'b1;
            secbl<=4'b1001;
     end
else if(secal>0&secbl>0)
              begin
                   secah<=secah;
```

```
                secal<=secal-1'b1;
                secbh<=secbh;
                secbl<=secbl-1'b1;
            end
else
    begin
        if(secah==0&secal==0)
            state<=s1;
    end
end
                        default:state<=s1;
                    endcase
                end
else
    begin
        if(pause)
            begin
                light<=light;
                secah<=secah;
                secal<=secal;
                secbh<=secbh;
                secbl<=secbl;
            end
    end
end
always@(*)
begin
    if(CLR)
      light<=6'b100100; //主从干道红灯亮
    else if(stopa)
```

```
        light<=6'b100001; //主红次绿
    else if(stopb)
        light<=6'b001100; //主绿次红
    else
    begin
        if(start)
            begin
                case(state)
                        s1:light<=6'b001100;   //主绿次红
                        s2:light<=6'b010100;   //主黄次红
                        s3:light<=6'b100001;   //主红次绿
                        s4:light<=6'b100010;   //主红次黄
                        default:light<=6'b000000;
                endcase
            end
    end
end
endmodule

module smgsm(
    input CLK, //扫描的间隔始终，控制数码管轮流点亮的频率
    input [3:0]secah,   //主干道读秒倒计时十位
    input [3:0]secal,   //次干道读秒倒计时个位
    input [3:0]secbh,   //主干道读秒倒计时十位
    input [3:0]secbl,   //次干道读秒倒计时个位
    output reg [3:0]AN,//位选
    output reg [7:0]seg //段选
    );
    reg [3:0] data;
```

```
    reg [1:0] bit_sel;   //数码管计数器指示，00~11  最左到最右
initial
            begin
                    bit_sel<=2'b00;
            end
    always@(posedge CLK)
        begin
             bit_sel<=bit_sel+1'b1;
             case(bit_sel)
                    2'b00:begin data<=secah;AN=4'b0111; end
                    2'b01:begin data<=secal;AN=4'b1011; end
                    2'b10:begin data<=secbh;AN=4'b1101;end
                    2'b11:begin data<=secbl;AN=4'b1110;end
                    default:begin data<=data;AN=4'b1111;end
             endcase

             case(data)
                      0:seg<=8'b00000011;
                    1:seg<=8'b10011111;
                    2:seg<=8'b00100101;
                    3:seg<=8'b00001101;
                    4:seg<=8'b10011001;
                    5:seg<=8'b01001001;
                    6:seg<=8'b01000001;
                    7:seg<=8'b00011111;
                    8:seg<=8'b00000001;
                    9:seg<=8'b00001001;
                    default:seg<=8'b11111111;
             endcase
        end
```

```
endmodule

module fpq1(              //分频  数码管扫描
    input clk_1M,
    output reg clk
        );
        reg [31:0]counter;
        initial
              begin
                  counter<=31'd0;
                  clk<=1'b0;
              end
        always@(posedge clk_1M)
        begin
           if(counter==31'd2000)
                    begin
                        clk<=~clk;
                        counter<=31'd0;
                    end
           else
                   counter<=counter+1'b1;
        end
endmodule

module fpq2(              //分频  交通灯倒计时
    input clk_1M,
    output reg CLK
        );
        reg [31:0]counter;
        initial
```

```
            begin
                counter<=31'd0;
                CLK<=1'b0;
            end
        always@(posedge clk_1M)
        begin
           if(counter==31'd50000000)
                    begin
                        CLK<=~CLK;
                        counter<=31'd0;
                    end
           else
                    counter<=counter+1'b1;
        end
endmodule
```

三、实验仿真

1.仿真代码

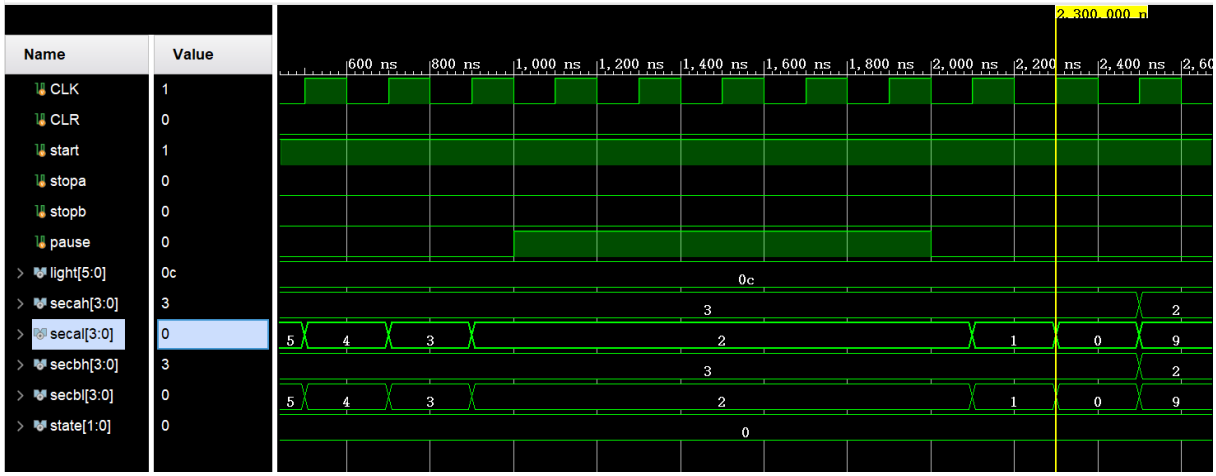
这里只是交通灯控制模块的仿真激励代码

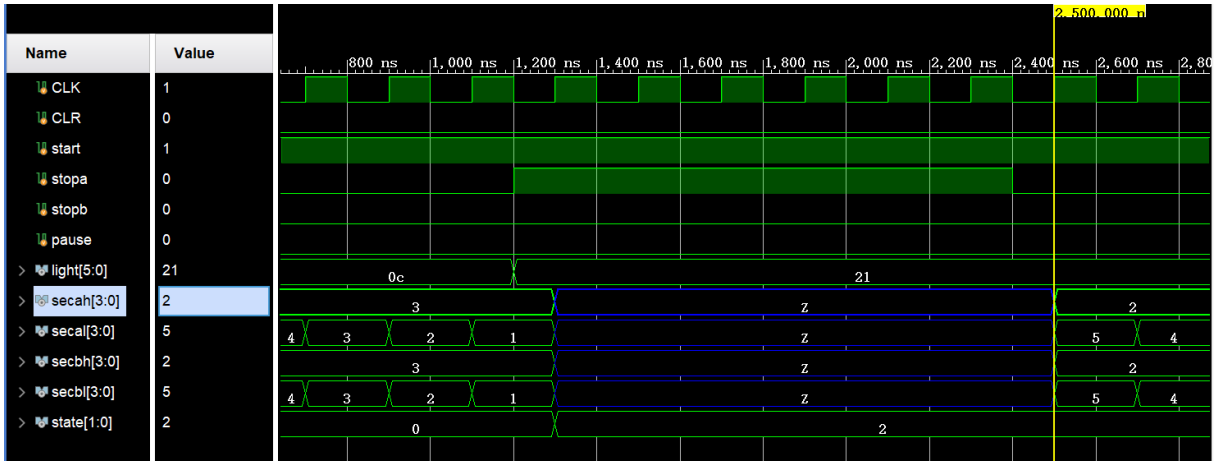
```
 module jtdfz;
//INPUT
reg CLK;       //时钟
reg CLR;     //系统复位
reg start; //交通灯正常启动
reg stopa; //主干道阻塞不通行
reg stopb; //次干道阻塞不通行
reg pause; //系统暂停
 //OUTPUT
wire [5:0]light; //前三位主干道的三盏 红 黄 绿 LED 灯  后三位次干道的三盏 红 黄 绿 LED 灯  从左到右
```

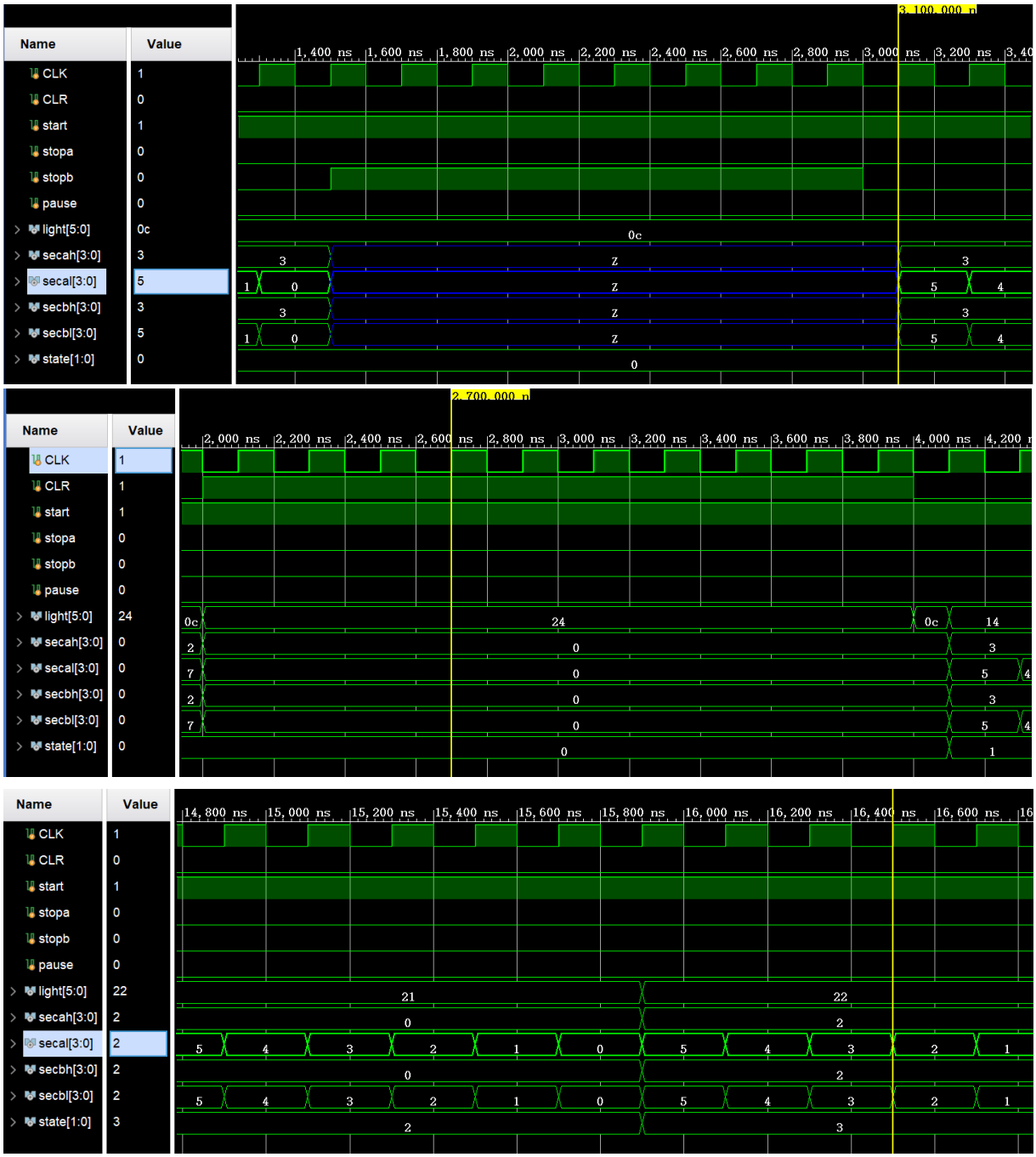
```
//wire [7:0]seca;   //主干道读秒倒计时
//wire [7:0]secb;   //次干道读秒倒计时
wire[3:0]secah;   //主干道读秒倒计时十位
wire[3:0]secal;   //次干道读秒倒计时个位
wire[3:0]secbh; //主干道读秒倒计时十位
wire[3:0]secbl; //次干道读秒倒计时个位
wire [1:0]state; //状态机
initial
        begin
            CLK=0;CLR=0;start=1;stopa=0;stopb=0;pause=0;
         end
always
        begin
            #10 CLK=~CLK;
        end
always
        begin
            #20000 CLR=~CLR;
        end
always
        begin
            #8000 start=~start;
        end
  always
        begin
            #12000 stopa=~stopa;
        end
  always
        begin
            #15000 stopb=~stopb;
```

```
        end
  always
        begin
            #10000 pause=~pause;
        end
  jtd ha(CLK,CLR,start,stopa,stopb,pause,light,secah,secal,secbh,secbl,state);
endmodule
```

2.仿真波形







3.仿真结果分析

对图中的波形进行分析，对与输入的 CLK,CLR,start,stopa,stopb,pause 的值，

当 CLR=1 时，light=100100 主次干道都是红灯,secah,secal,secbh,secbl 都为 0。

当 stopa=1 时，light=100001 主干道红灯次干道绿灯,

secah,secal,secbh,secbl 都为 z,高阻抗不输出。

当 stopb=1 时，light=001100 主干道红灯次干道绿灯,

secah,secal,secbh,secbl 都为 z,高阻抗不输出。

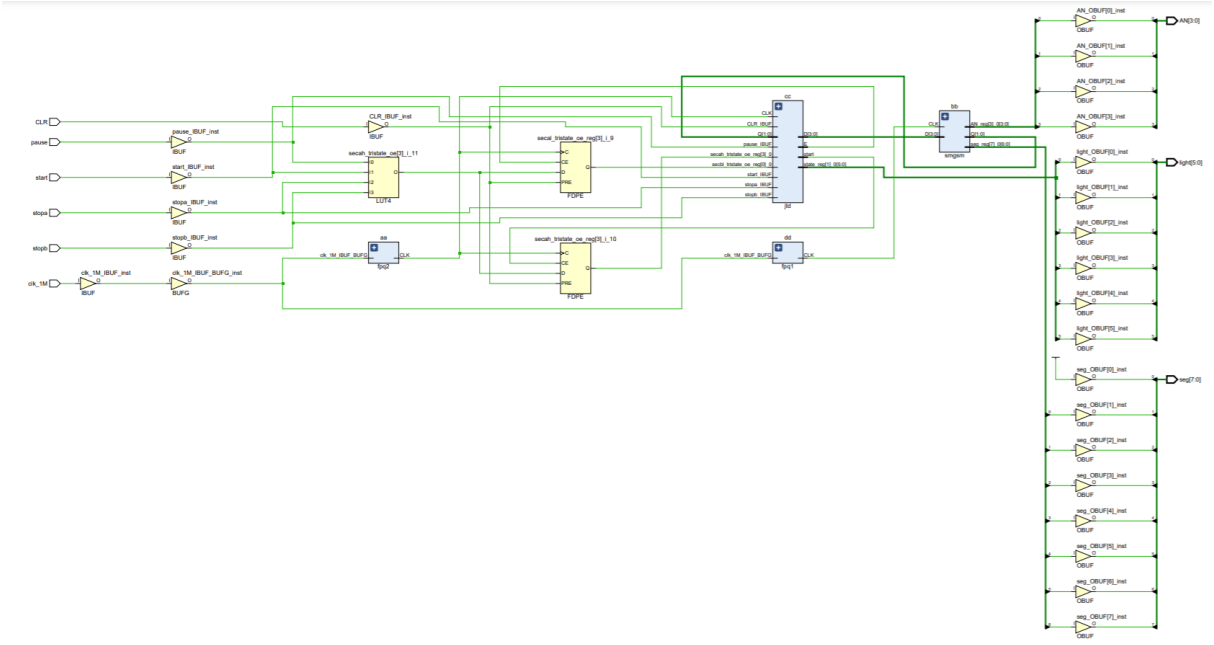
当 pause=1 时，light=light 主干道次干道灯保持不变,

secah,secal,secbh,secbl 也都保持不变。

当 start=1 时，状态 s1 到 s2 到 s3 到 s4 再到 s1；变化正确；但出现了抖动的情况，产生了奇怪的结果。但大体符合。

可见其符合交通灯控制的逻辑功能，但有抖动。

四、电路图



五、引脚配置

```
set_property IOSTANDARD LVCMOS18 [get_ports{AN[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {light[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {light[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {light[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {light[2]}]
```

```
set_property IOSTANDARD LVCMOS18 [get_ports {light[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {light[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports clk_1M]
set_property IOSTANDARD LVCMOS18 [get_ports CLR]
set_property IOSTANDARD LVCMOS18 [get_ports pause]
set_property IOSTANDARD LVCMOS18 [get_ports start]
set_property IOSTANDARD LVCMOS18 [get_ports stopa]
set_property IOSTANDARD LVCMOS18 [get_ports stopb]
set_property PULLDOWN true [get_ports clk_1M]
set_property PULLDOWN true [get_ports CLR]
set_property PULLDOWN true [get_ports pause]
set_property PULLDOWN true [get_ports start]
set_property PULLDOWN true [get_ports stopa]
set_property PULLDOWN true [get_ports stopb]
set_property PACKAGE_PIN V5 [get_ports CLR]
set_property PACKAGE_PIN U16 [get_ports pause]
set_property PACKAGE_PIN T4 [get_ports start]
set_property PACKAGE_PIN V6 [get_ports stopa]
set_property PACKAGE_PIN T5 [get_ports stopb]
set_property PACKAGE_PIN U6 [get_ports {light[5]}]
set_property PACKAGE_PIN R5 [get_ports {light[4]}]
set_property PACKAGE_PIN U7 [get_ports {light[3]}]
```

set_property PACKAGE_PIN R6 [get_ports {light[2]}]

set_property PACKAGE_PIN R7 [get_ports {light[1]}]

set_property PACKAGE_PIN U8 [get_ports {light[0]}]

set_property PACKAGE_PIN C9 [get_ports {AN[0]}]

set_property PACKAGE_PIN C10 [get_ports {AN[1]}]

set_property PACKAGE_PIN D10 [get_ports {AN[2]}]

set_property PACKAGE_PIN C11 [get_ports {AN[3]}]

set_property PACKAGE_PIN F14 [get_ports {seg[7]}]

set_property PACKAGE_PIN N14 [get_ports {seg[6]}]

set_property PACKAGE_PIN J13 [get_ports {seg[5]}]

set_property PACKAGE_PIN G13 [get_ports {seg[4]}]

set_property PACKAGE_PIN F13 [get_ports {seg[3]}]

set_property PACKAGE_PIN G14 [get_ports {seg[2]}]

set_property PACKAGE_PIN M13 [get_ports {seg[1]}]

set_property PACKAGE_PIN H14 [get_ports {seg[0]}]

set_property PACKAGE_PIN E3 [get_ports clk_1M]

六、思考与探索

1.实验结果记录：

CLR	start	stopa	stopb	pause	主干道倒计时	次干道倒计时	交通灯
1	1	0	0	0	0	0	主红次红
0	1	1	0	0	无	无	主红次绿
0	1	0	1	0	无	无	主绿次红
0	1	0	0	0	35	35	主绿次红
0	1	0	0	0	5	5	主黄次红
0	1	0	0	1	3	3	主黄次红
0	1	0	0	0	3	3	主黄次红
0	1	0	0	0	25	25	主红次绿
0	1	0	0	0	5	5	主红次黄

2.实验结论

刚开始的代码实验结果并不正确，经过老师的指导后，我把控制灯信号的代码提了出来，不再受其他因素的干扰，实验结果就配对正确了。基本上实现了交通灯的逻辑控制与状态转化。但还有个问题就是不怎么符合实际应用。

3.问题与解决方案

问题：实验中出现了状态倒计时显示正确，交通灯的显示却受抖动干扰，不能正确显示的问题以及 pause 代码重复使用，程序显得臃肿的问题，以及不符合实际的问题。

解决方案：其中交通灯的的显示问题，我根据老师的指导把其单独提出在一个 always 中，不受其他干扰，可以正确显示。

```
  always@(*)
begin
    if(CLR)
       light<=6'b100100; //主从干道红灯亮
    else if(stopa)
       light<=6'b100001; //主红次绿
    else if(stopb)
       light<=6'b001100; //主绿次红
    else
    begin
        if(start)
            begin
                case(state)
                        s1:light<=6'b001100;   //主绿次红
                        s2:light<=6'b010100;   //主黄次红
                        s3:light<=6'b100001;   //主红次绿
                        s4:light<=6'b100010;   //主红次黄
                        default:light<=6'b000000;
                 endcase
             end
```

```
end
```

对于 pause 重复判断使用相同代码的问题，我把判定条件改成了

```
else if(pause==0&start) ....
else
    begin
        if(pause)
            begin
                light<=light;
                secah<=secah;
                secal<=secal;
                secbh<=secbh;
                secbl<=secbl;
            end
    end
```

即解决了代码重复问题。

4.思考题

利用状态机编程应该注意什么？

答：（１）无论您决定采用哪一种方法来实现状态机，都需要使用 CASE 语句来评估下一状态的判定和任何输出。

（２）状态机的输出与时钟保持同步。

（３）注意状态编码