集成电路设计实验报告

himingway

2015 年 12 月 1 日

1 实验目的

通过交通灯的设计仿真和综合，体会复杂时序的实现方法，学会用框图表示程序的设计思路，掌握中小规模集成电路的设计方法及仿真技巧。

2 设计要求

设计一个十字路口交通信号灯的控制电路。要求红、绿灯按一定的规律亮和灭，并在亮灯期间进行倒计时，且将运行时间用数码管显示出来。绿灯亮时，为该车道允许通信信号，红灯亮时，为该车道禁止通信信号。要求主干道每次通行时间为 Tx 秒，支干道每次通行时间为 Ty 秒。每次变换运行车道前绿灯闪烁，持续时间为 5s，即车道要由 X 转换为 Y 时，X 在通行时间只剩下 5 秒钟时，绿灯闪烁显示，Y 仍为红灯。

3 项目链接

Github：`https://github.com/himingway/traffic_light`
网站：`http://www.turnright.xyz/archives/1241.html`

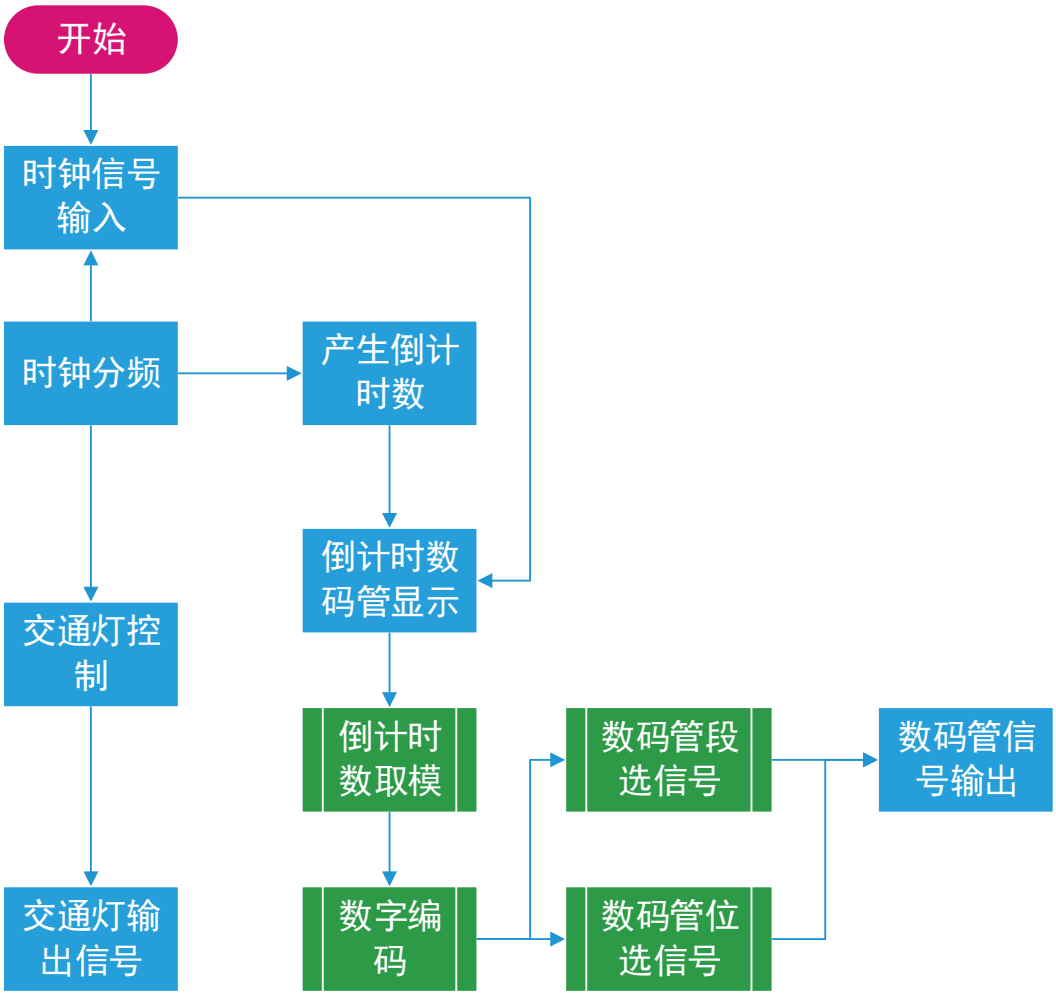
4 设计方案

4.1 设计思路

整个设计分为三个部分。第一个部分是顶层模块，将各个模块封装起来。第二个部分是控制模块，用来控制“交通灯”的亮灭。第三个部分是“倒计时产生”模块，用来产生倒计时数。第四个部分是“数码管显示”模块，包括数字取模、BCD 编码和数码管段选与位选这三个部分。

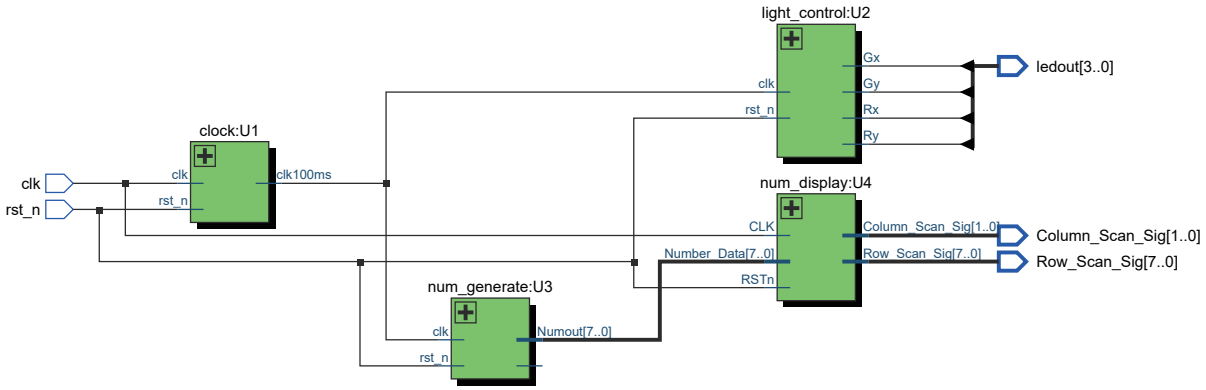
其中控制模块、和“倒计时产生”采用周期为 10Hz 的时钟作为输入时钟，“数码管显示”模块采用较高的时钟频率输入。实验设计思路如下流程图所示。

4.2 设计流程图



5 模块设计

5.1 顶层模块



5.1.1 顶层模块功能

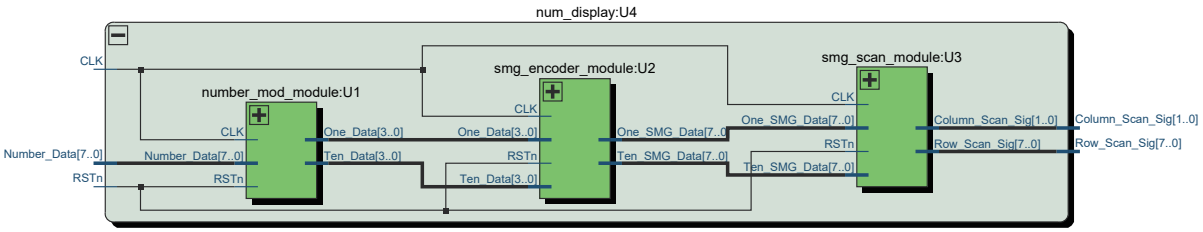
- clock 模块：产生周期为 100ms 的时钟。

- light_control 模块：产生控制“交通灯”亮灭的信号。
- num_generate 模块：产生“倒计时数字”信号。
- num_display 模块：控制数码管显示“倒计时数字”

5.1.2 顶层模块端口

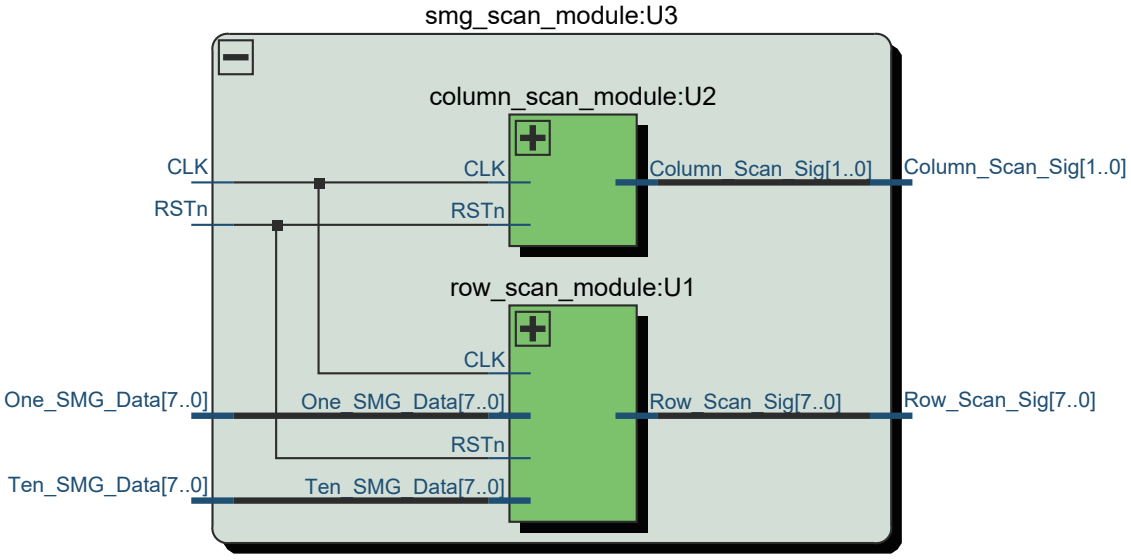
- clk：时钟信号输入。
- rst_n：异步复位信号输入。
- ledout：“交通灯”亮灭的信号输出。
- Column_Scan_Sig：数码管位选信号输出。
- Row_Scan_Sig：数码管段选信号输出。

5.2 num_display 模块



- number_mod_module：数字取模模块。
- smg_encoder_module：数码管编码模块。
- smg_scan_module：数码管扫描模块（用于位选和段选）

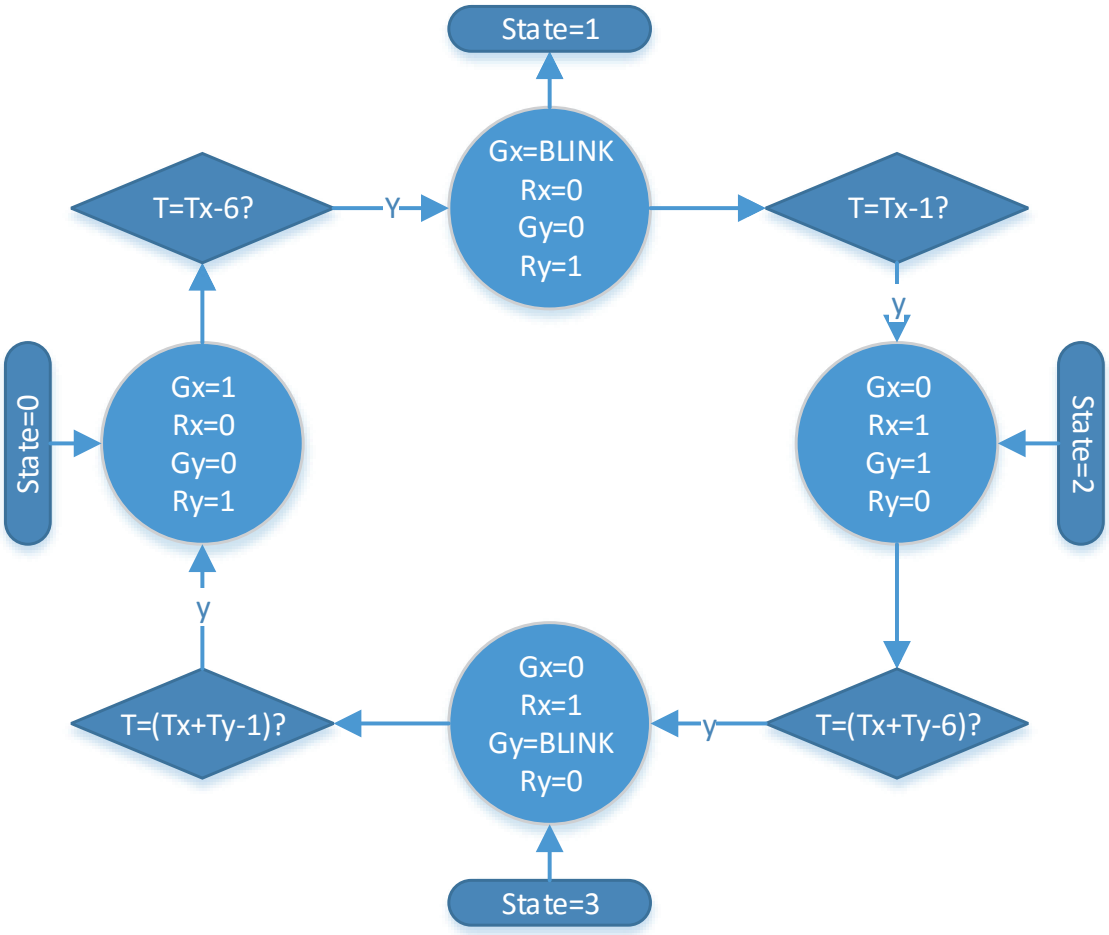
其中”*column_scan_module*”用于位选，”*raw_scan_module*”用于位选



6 时序逻辑设计

6.1 “交通信号灯”控制时序逻辑设计

6.1.1 module light_control 状态图



6.1.2 图例

Gx	X 车道绿灯
Rx	X 车道红灯
Gy	Y 车道绿灯
Ry	Y 车道红灯
BLINK	闪烁
Tx	X 车道通行时间
Ty	Y 车道通行时间

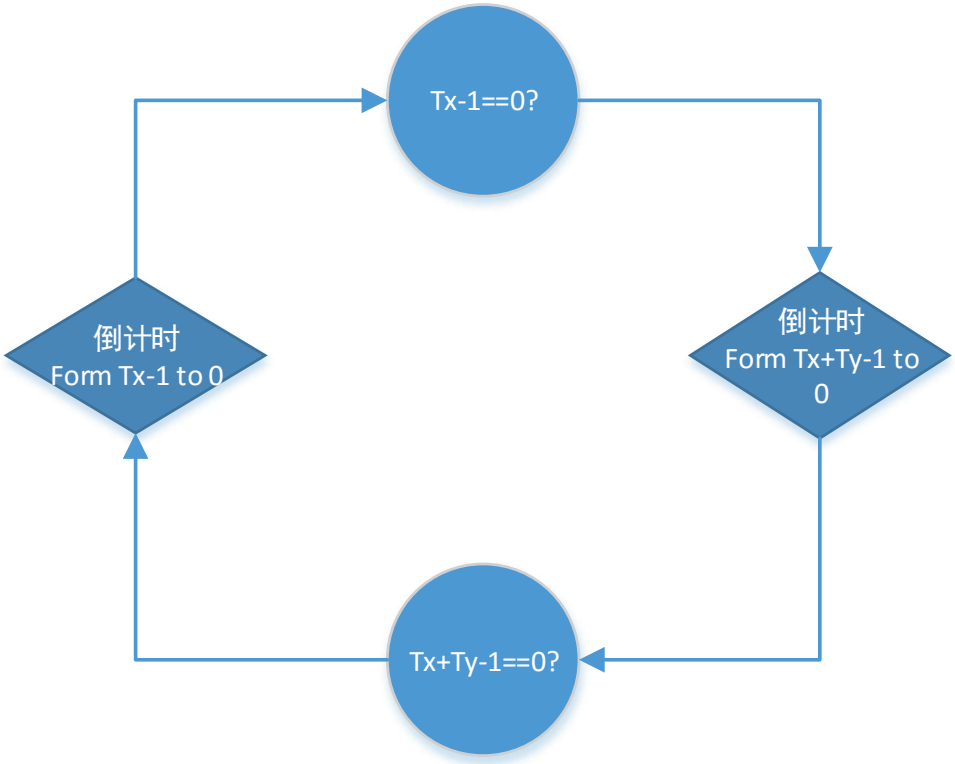
6.1.3 module light_control 状态图解释

开始时，X 车道绿灯亮红灯灭 (state=0)，Y 车道绿灯灭，红灯亮，计数器从 0 开始计数，当计数器计数了 (Tx-6) 个数时，进入 X 车道绿灯闪烁状态 (state=1)。这时，

计数器接着计数，当计数器计数了 (Tx-1) 个数时，X 车道绿灯停止闪烁，变为熄灭状态，红灯开始亮；Y 车道绿灯亮，红灯灭。即进入 (state=3) 状态。(state=3) 状态和 (state=4) 状态与 (state=0)、(state=1) 状态类似，经过一个周期 (T=Tx+Ty) 时长的循环，状态机重新回到初始的状态。

6.2 “倒计时数字”模块时序逻辑图

6.2.1 num_generate 状态图



6.2.2 num_generate 状态图解释

这是一个倒计时数产生模块，功能是产生“交通灯”所需要的倒计时数，以便于将倒计时数信号输入到数码管显示模块中。该模块时序功能很简单。当 X 车道通行时，计数器从 (Tx-1) 计数到 0；当 Y 车道通行时，计数器从 (Ty-1) 计数到 0。

7 其他模块的设计

7.1 分频器设计

7.1.1 分频器介绍

分频器是指使输出信号频率为输入信号频率整数分之一的电子电路。在许多电子设备中如电子钟、频率合成器等，需要各种不同频率的信号协同工作，常用的方法是以稳

定度高的晶体振荡器为主振源，通过变换得到所需要的各种频率成分，分频器是一种主要变换手段。

7.1.2 分频器设计实现

测试所用的硬件时钟输入频率是 50MHz，也就是晶振 1s 震荡 50M 次。本次设计要得到 10Hz 的时钟信号，需要用分频器对输入时钟进行分频。

下面对时钟频率进行计算：

1.FPGA 的时钟频率是 50MHz = 50_000_000Hz

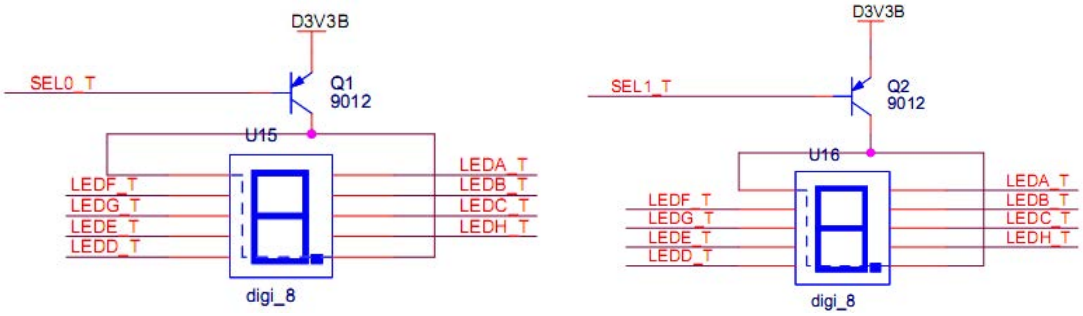
2. 要得到 10Hz 的时钟，计数器数到 50_000_000 / 10 = 5_000_000，输出时钟为一个周期。

3. 输出时钟的波形跳变 (0 变为 1 或者 1 变为 0) 时间为 5_000_000 / 2 = 2_500_000。

4. 由于计数器从 0 开始计数，相应的次数应该减去 1。即 5_000_000 -1 = 4_999_999，2_500_000 -1 = 2_499_999。

7.2 数码管显示模块

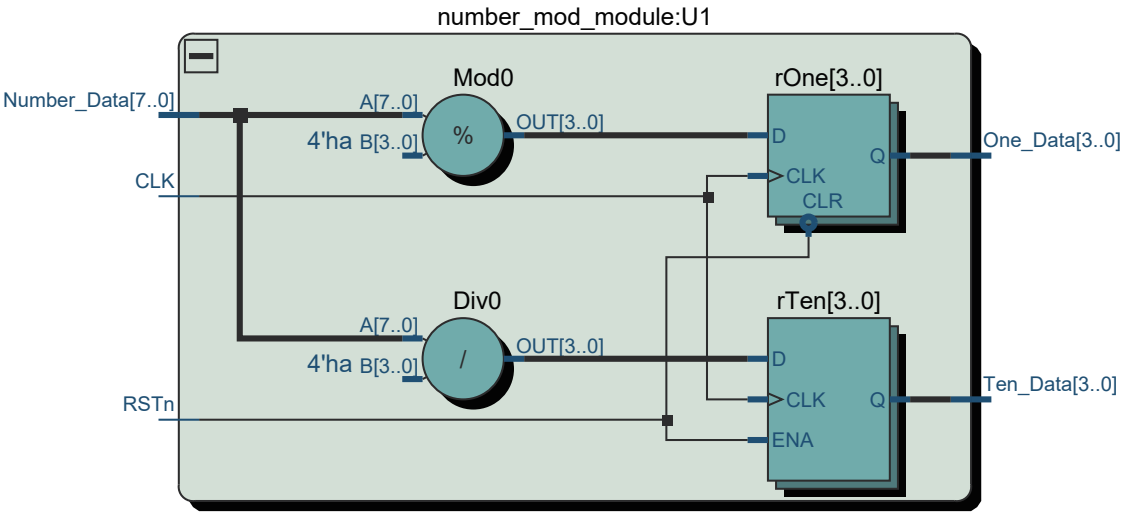
7.2.1 数码管驱动电路



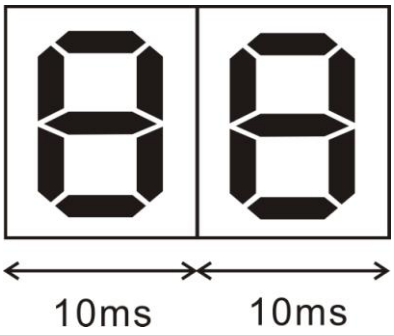
数码管是共阳，而是用 PNP 管来反响驱动并且控制列扫描（SEL0_T 和 SEL1_T）。而且所有的数码管的“段选信号”（LEDA .. LEDH）都共用同样的引脚。结论来说，数码管都信号都是“低电平有效”。

7.2.2 数字取模模块

“number_mod_module.v”的设计是利用数学运算符“%”和“/”分别取得十位和个位。因为是十位取位的关系，所以最大的输入数是 0099 而已。



7.2.3 数码管段选和位选模块



因为要求是两个数码管资源，假设各个数码管点亮时间是 10ms，两个数码管所占用的时间自然是 20ms。换句话说，就是完成一次扫描占用 20ms 的周期时间。column_scan_module.v 是负责“列扫描”，亦即每隔 10ms 就使能（点亮）不同的数码管。row_scan_module.v 主要是每隔 10ms，输出不同的数码管码。

8 verilog 代码

8.1 traffic_light.v

这是本设计的顶层模块，包含了所有模块的全部功能。

该模块设置了两个参数，Tx 和 Ty，当修改 X 车道和 Y 车道的通行时间时，仅仅需要改动参数大小即可，无需改变整个代码。这是本设计的优点。

```
1 module traffic_light (
2   input clk,     // Clock
3   input rst_n,   // Asynchronous reset active low
4   output [7:0]Row_Scan_Sig, //数码管段选
5   output[1:0]Column_Scan_Sig, //数码管位选
```

```
6     output [3:0]ledout //灯控制输出
7  );
8
9  parameter Tx=30; //X车道通行时间
10 parameter Ty=15; //Y车道通行时间
11
12 wire clk100ms;
13 wire [7:0] Numout;
14 clock U1 //时钟模块
15 (
16    .clk(clk),
17    .rst_n(rst_n),
18    .clk100ms(clk100ms)
19    );
20
21 light_control #(Tx,Ty) U2 //交通灯控制模块
22 (
23    .clk(clk100ms),
24    .rst_n(rst_n),
25    .Gx(ledout[0]),
26    .Rx(ledout[1]),
27    .Gy(ledout[2]),
28    .Ry(ledout[3])
29    );
30
31 num_generate #(Tx,Ty) U3 //倒计时数参数模块
32 (
33    .clk(clk100ms),
34    .rst_n(rst_n),
35    .Numout(Numout)
36    );
37
38 num_display U4 //倒计时数显示模块
39 (
40    .CLK(clk),
41    .RSTn(rst_n),
42    .Number_Data(Numout),
```

```
43     .Row_Scan_Sig(Row_Scan_Sig),
44     .Column_Scan_Sig(Column_Scan_Sig)
45     );
46 endmodule
```

8.2 clock.v

时钟分频模块，用来产生 100ms 的周期的时钟。

至于为啥要用 100ms 的时钟，这里有两个原因。第一个原因是绿灯闪烁周期设为 1s，从亮到灭或者从灭到亮分别要用 500ms，虽然产生 1s 周期的时钟更方便些，但是 1s 周期的时钟无法分成比 1s 还要小的时钟周期单位，也就是说分频容易倍频难。第二个原因是便于仿真，仿真时，只要输入几百个时钟周期就能完成一个周期的“红绿灯”闪烁功能。

```
1  /*This module is used to generate the 100ms clock*/
2  module clock (
3    input clk,     // Clock
4    input rst_n, // Asynchronous reset active low
5    output clk100ms //100ms周期的时钟输出
6  );
7  reg [23:0] cnt100ms;
8  reg rclk100ms;
9  //the clock of the FPGA is 50MHz=50_000_000HZ
10 //generate 100ms clock
11 always @(posedge clk or negedge rst_n) begin : proc_cnt100ms
12   if(~rst_n) begin
13     cnt100ms <= 0;
14     rclk100ms <=0;
15   end else begin
16     if(cnt100ms == 24'd2_499_999) begin
17       cnt100ms <= 24'd0;
18       rclk100ms <= ~rclk100ms;//一个周期计数器计数2_499_999次
19     end
20     else
21       cnt100ms <= cnt100ms + 1'd1;
22   end
23 end
24 assign clk100ms = rclk100ms;
```

```
25 endmodule
```

8.3 light_control.v

交通灯控制模块，用来产生控制交通灯亮灭的信号。状态图详见 **6.1.1**.

```
1  module light_control (
2    input clk,     // Clock 输入时钟周期为100ms
3    input rst_n,   // Asynchronous reset active low
4    output Gx,Rx,Gy,Ry
5    /*信号输出，其中Gx为X车道绿灯，Rx为x车道红灯，Gy为x车道红灯，
       Ry为x车道红灯*/
6  );
7
8  reg rGx;
9  reg rGy;
10 reg rRx;
11 reg rRy;
12 reg [1:0] state;
13 reg [23:0] cnt;
14
15 parameter Tx =30; //X车道通行时间
16 parameter Ty = 15; //Y车道通行时间
17
18 /*计数器，从(Tx+Ty)*10-1开始倒计数，计数到0然后重新开始计数，在
     输入时钟周期为100ms的条件下，每一个计数周期用时为(Tx+Ty)秒*/
19 always @(posedge clk or negedge rst_n) begin : proc_cnt
20   if(~rst_n) begin
21     cnt <= 0;
22   end else begin
23     if (cnt == (Tx+Ty)*10-1)
24       cnt <= 0;
25     else
26       cnt <= 1'd1+cnt;
27   end
28 end
29
30 \*交通灯控制模块状态机，原理解释详见6.1.3*\
```

```
31 always @(posedge clk or negedge rst_n) begin :
     proc_light_control
32   if(~rst_n) begin
33     state <= 2'd0;
34     rGx <= 1'd0;
35     rGy <= 1'd0;
36     rRx <=1'd0;
37     rRy <= 1'd0;
38   end else begin
39     case (state)
40       2'd0:begin \\X车道通行，开始倒计时
41         if(cnt == Tx * 10 - 51) begin
42           state <= 2'd1;
43           rGx <= 1'b0;
44         end
45         else
46           rGx <= 1'b1;
47           rRy <= 1'b1;
48           rRx <= 1'b0;
49           rGy <= 1'b0;
50       end
51       2'd1:begin \\X车道绿灯闪烁
52         if(cnt == Tx*10 -1) begin
53           state <= 2'd2;
54           rGx <= 1'b0;
55         end
56         else if(cnt == Tx*10 - 6) begin
57           rGx <= 1'b1;
58         end
59         else if(cnt == Tx*10 - 11) begin
60           rGx <= 1'b0;
61           end
62         else if(cnt == Tx*10 - 16) begin
63           rGx <= 1'b1;
64           end
65         else if(cnt == Tx*10 - 21) begin
66           rGx <= 1'b0;
```

```
67            end
68          else if(cnt == Tx*10 - 26) begin
69            rGx <= 1'b1;
70            end
71          else if(cnt == Tx*10 - 31) begin
72            rGx <= 1'b0;
73            end
74          else if(cnt == Tx*10 - 36) begin
75            rGx <= 1'b1;
76            end
77          else if(cnt == Tx*10 - 41) begin
78            rGx <= 1'b0;
79            end
80          else if(cnt == Tx*10- 46) begin
81            rGx <= 1'b1;
82            end
83        end
84        2'd2:begin \\Y车道通行，开始倒计时
85          if(cnt == (Tx+Ty)*10-51) begin
86            state <= 2'd3;
87            rGy <= 1'b0;
88          end
89          else begin
90            rRy <=1'b0;
91            rGy <= 1'b1;
92            rRx <= 1'b1;
93          end
94        end
95        2'd3:begin \\Y车道绿灯闪烁
96          if(cnt == (Tx+Ty)*10-1) begin
97            rGy <= 1'b0;
98            state <= 2'd0;
99          end
100         else if(cnt == (Tx+Ty)*10-6) begin
101           rGy <= 1'b1;
102         end
103         else if(cnt == (Tx+Ty)*10-11) begin
```

```
104             rGy<= 1'b0;
105             end
106           else if(cnt == (Tx+Ty)*10-16) begin
107             rGy <= 1'b1;
108             end
109           else if(cnt == (Tx+Ty)*10-21) begin
110             rGy <= 1'b0;
111             end
112           if(cnt == (Tx+Ty)*10-26) begin
113             rGy <= 1'b1;
114           end
115           else if(cnt == (Tx+Ty)*10-31) begin
116             rGy <= 1'b0;
117           end
118           else if(cnt == (Tx+Ty)*10-36) begin
119             rGy<= 1'b1;
120             end
121           else if(cnt == (Tx+Ty)*10-41) begin
122             rGy <= 1'b0;
123             end
124           else if(cnt == (Tx+Ty)*10-46) begin
125             rGy <= 1'b1;
126             end
127         end
128         default : state <= 2'd0;
129       endcase
130     end
131 end
132
133 assign Gx = rGx;
134 assign Gy = rGy;
135 assign Rx = rRx;
136 assign Ry = rRy;
137 endmodule
```

8.4 num_generate

数字产生模块，状态图如 6.2.1 所示。但是这里没有用状态机，用了类似计算机编程语言的顺序写法，效果和状态机完全一致，实现的功能电路也一致，这也本程序的一个尝试。

```
1  module num_generate (
2    input clk,     // Clock
3    input rst_n, // Asynchronous reset active low
4    output [7:0]Numout, //倒计时数字输出
5    );
6
7  parameter Tx =30;
8  parameter Ty = 15;
9
10 reg [7:0] rNumout = Tx-1;
11 reg [8:0]cnt1;
12 reg clk1;
13 reg flag;
14
15 always @(posedge clk or negedge rst_n) begin : proc_Numour
16   if(~rst_n) begin
17     rNumout <= Tx-1;
18     cnt1 <=0;
19   end else begin //X车道倒计时
20     if(flag ==0) begin
21       if(cnt1 == 9) begin
22         cnt1 <=0;
23         rNumout <=rNumout -1'b1;
24       end
25       else cnt1 <= cnt1 +1'b1;
26       if(rNumout == 8'b11111111) begin
27         rNumout <= Ty-1;
28         flag <=1; //当X车道倒计时完毕时
29       end
30     end
31     else if (flag==1) begin
32       if(cnt1 == 9) begin //Y车道倒计时
```

```
33            cnt1 <=0;
34            rNumout <=rNumout -1'b1;
35          end
36          else cnt1 <= cnt1 +1'b1;
37          if(rNumout == 8'b11111111) begin
38            rNumout <= Tx-1;
39            flag <=0;
40          end
41        end
42        end
43      end
44      assign Numout = rNumout;
45
46  endmodule
```

8.5 num_display

数码管显示模块的顶层模块，包含三个部分。一、数字取模模块，也就是把两位数数字拆成十位数数字和个位数数字。二、BCD 编码器编码模块，将一为数字转换位数码管编码。三、数码管扫描模块，包括段选扫描和位选扫描。

```
1  module num_display
2  (
3        CLK, RSTn,
4      Number_Data, //输入倒计时数字信号
5      Row_Scan_Sig, Column_Scan_Sig //输出段选和位选信号
6  );
7
8        input CLK;
9      input RSTn;
10     input [7:0]Number_Data;
11     output [7:0]Row_Scan_Sig;
12     output [1:0]Column_Scan_Sig;
13
14     wire [3:0]Ten_Data;
15     wire [3:0]One_Data;
16
17     number_mod_module U1 //数字取模模块
```

```
18      (
19          .CLK( CLK ),
20          .RSTn( RSTn ),
21          .Number_Data( Number_Data ),
22          .Ten_Data( Ten_Data ),
23          .One_Data( One_Data )
24      );
25
26
27
28      wire [7:0]Ten_SMG_Data;
29      wire [7:0]One_SMG_Data;
30
31      smg_encoder_module U2 //BCD编码器编码模块
32      (
33          .CLK( CLK ),
34          .RSTn( RSTn ),
35          .Ten_Data( Ten_Data ),
36          .One_Data( One_Data ),
37          .Ten_SMG_Data( Ten_SMG_Data ),
38          .One_SMG_Data( One_SMG_Data )
39      );
40
41      smg_scan_module U3 //数码管扫描模块
42      (
43          .CLK( CLK ),
44          .RSTn( RSTn ),
45          .Ten_SMG_Data( Ten_SMG_Data ),
46          .One_SMG_Data( One_SMG_Data ),
47          .Row_Scan_Sig( Row_Scan_Sig ),
48          .Column_Scan_Sig( Column_Scan_Sig )
49      );
50
51  endmodule
```

8.6 number_mod_module

数字取模模块。功能很简单，利用数学运算符“%”和“/”分别取得十位和个位。将输出的数字送入编码器。

```
module number_mod_module
(
  CLK,RSTn,Number_Data,Ten_Data,One_Data
);
  input CLK;
  input RSTn;
  input [7:0]Number_Data;
  output[3:0]Ten_Data; //输出十为数字
  output[3:0]One_Data; //输出个位数字

  /*****************************************/

   reg [31:0]rTen;
   reg[31:0]rOne;

   always@(posedge CLK or negedge RSTn)
    if(!RSTn)
      begin
        rOne<=32'd0;
      end
    else
      begin
        rTen<=Number_Data/10; //获取十位数字
        rOne<=Number_Data%10; //获取个位数字
      end

      assign Ten_Data=rTen[3:0];
      assign One_Data=rOne[3:0];
  endmodule
```

8.7 smg_encoder_module

```
module smg_encoder_module
```

```
2  (
3    CLK,RSTn,Ten_Data,One_Data,Ten_SMG_Data,One_SMG_Data
4  );
5
6    input CLK;
7    input RSTn;
8    input [3:0]Ten_Data; //输入十位数数字
9    input [3:0]One_Data; //输入个位数数字
10   output[7:0]Ten_SMG_Data; //输出十位数编码信号
11   output[7:0]One_SMG_Data; //输出个位数编码信号
12
13   parameter
14   _0=8'b1100_0000,
15   _1=8'b1111_1001,
16   _2=8'b1010_0100,
17   _3=8'b1011_0000,
18   _4=8'b1001_1001,
19   _5=8'b1001_0010,
20   _6=8'b1000_0010,
21   _7=8'b1111_1000,
22   _8=8'b1000_0000,
23   _9=8'b1001_0000;
24
25   reg[7:0] rTen_SMG_Data;
26
27   always@(posedge CLK or negedge RSTn)
28     if(!RSTn)
29       begin
30         rTen_SMG_Data<=8'b1111_1111;
31       end
32     else
33       case(Ten_Data) //十位数字编码
34         4'd0:rTen_SMG_Data<=_0;
35         4'd1:rTen_SMG_Data<=_1;
36         4'd2:rTen_SMG_Data<=_2;
37         4'd3:rTen_SMG_Data<=_3;
38         4'd4:rTen_SMG_Data<=_4;
```

```
39            4'd5:rTen_SMG_Data<=_5;
40            4'd6:rTen_SMG_Data<=_6;
41            4'd7:rTen_SMG_Data<=_7;
42            4'd8:rTen_SMG_Data<=_8;
43            4'd9:rTen_SMG_Data<=_9;
44          endcase
45
46      reg[7:0]rOne_SMG_Data;
47
48      always@(posedge CLK or negedge RSTn)
49          if(!RSTn)
50            begin
51              rOne_SMG_Data<=8'b1111_1111;
52            end
53          else
54            case(One_Data) //个位数字编码
55            4'd0:rOne_SMG_Data<=_0;
56            4'd1:rOne_SMG_Data<=_1;
57            4'd2:rOne_SMG_Data<=_2;
58            4'd3:rOne_SMG_Data<=_3;
59            4'd4:rOne_SMG_Data<=_4;
60            4'd5:rOne_SMG_Data<=_5;
61            4'd6:rOne_SMG_Data<=_6;
62            4'd7:rOne_SMG_Data<=_7;
63            4'd8:rOne_SMG_Data<=_8;
64            4'd9:rOne_SMG_Data<=_9;
65          endcase
66    assign Ten_SMG_Data=rTen_SMG_Data;
67    assign One_SMG_Data=rOne_SMG_Data;
68  endmodule
```

8.8 smg_scan_module

数码管扫描顶层模块，包括数码管段选和位选。

```
1   module smg_scan_module
2   (
3    CLK,RSTn,Ten_SMG_Data,One_SMG_Data,
```

```
4     Row_Scan_Sig,Column_Scan_Sig
5   );
6
7     input CLK;
8     input RSTn;
9     input[7:0]Ten_SMG_Data;
10    input[7:0]One_SMG_Data;
11    output[7:0]Row_Scan_Sig;
12    output[1:0]Column_Scan_Sig;
13
14    row_scan_module U1 //段选模块
15    (
16       .CLK(CLK),
17       .RSTn(RSTn),
18       .Ten_SMG_Data(Ten_SMG_Data),
19       .One_SMG_Data(One_SMG_Data),
20       .Row_Scan_Sig(Row_Scan_Sig)
21    );
22
23    column_scan_module U2 //位选模块
24    (
25       .CLK(CLK),
26       .RSTn(RSTn),
27       .Column_Scan_Sig(Column_Scan_Sig)
28    );
29  endmodule
```

8.9 row_scan_module

数码管段选模块。每隔 10ms，将个位或十位数字编码信息交替输出到数码管中。

```
1  module row_scan_module
2  (
3     CLK,RSTn,Ten_SMG_Data,One_SMG_Data,Row_Scan_Sig
4  );
5     input CLK;
6     input RSTn;
7     input [7:0]Ten_SMG_Data;
```

```
8   input  [7:0]One_SMG_Data;
9   output [7:0]Row_Scan_Sig;
10
11  parameter T10MS=19'd499_999;
12
13  reg[18:0]  Count1;
14
15  /*分频器，产生10ms周期的时钟*/
16  always@(posedge CLK or negedge RSTn)
17    if(!RSTn)
18      Count1<=19'd0;
19    else if(Count1==T10MS)
20      Count1<=19'd0;
21    else
22      Count1<=Count1+19'b1;
23
24  reg[1:0]t;
25
26  /*每10ms，状态t改变一次*/
27  always @(posedge CLK or negedge RSTn)
28    if(!RSTn)
29      t<=2'd0;
30    else if (t==2'd2)
31      t<=2'd0;
32    else if (Count1==T10MS)
33      t<=t+1'b1;
34
35  reg[7:0]rData;
36
37  always@(posedge CLK or negedge RSTn)
38    if(!RSTn)
39      rData<=8'd0;
40    else if (Count1==T10MS)
41      case(t)
42        2'd0:rData<=Ten_SMG_Data; //将十位数字输送到数码管中
43        2'd1:rData<=One_SMG_Data; //将个位数字输送到数码管中
44      endcase
```

```
45
46    assign Row_Scan_Sig=rData;
47 endmodule
```

8.10 column_scan_module

数码管段选模块。每隔 10ms，交替使能个位和十位两个数码管。

```
1  module column_scan_module
2  (
3    CLK,RSTn,Column_Scan_Sig
4  );
5    input CLK;
6    input RSTn;
7    output [1:0] Column_Scan_Sig;
8
9    parameter T10MS=19'd499_999;
10
11   reg[18:0] Count1;
12
13   /*分频器，产生10ms周期的时钟*/
14   always @ (posedge CLK or negedge RSTn)
15     if(!RSTn)
16       Count1<=19'd0;
17     else if(Count1==T10MS)
18       Count1<=19'd0;
19     else
20       Count1<=Count1+19'b1;
21
22   reg[1:0]t;
23
24   /*每10ms，状态t改变一次*/
25   always@(posedge CLK or negedge RSTn)
26     if(!RSTn)
27       t<=2'd0;
28     else if (t==2'd2)
29       t<=2'd0;
30     else if(Count1==T10MS)
```

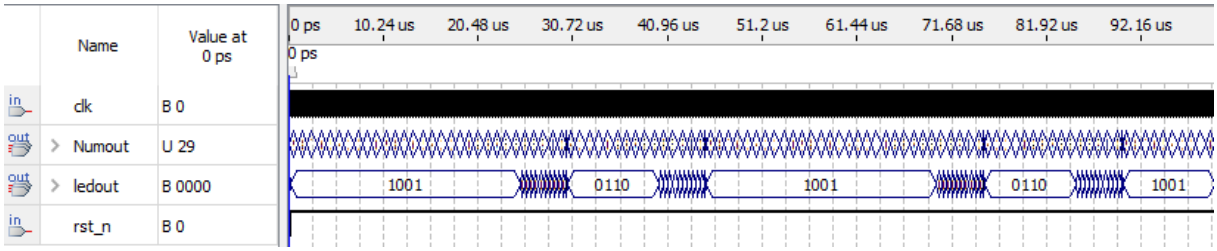
```
31          t<=t+1'b1;
32
33      reg[1:0]rColumn_Scan;
34
35      always@(posedge CLK or negedge RSTn)
36        if(!RSTn)
37          rColumn_Scan<=2'b10;
38        else if(Count1==T10MS)
39          case(t)
40            2'd0:rColumn_Scan<=2'b10; //十位位数码管使能
41            2'd1:rColumn_Scan<=2'b01; //个位位数码管使能
42          endcase
43
44      assign Column_Scan_Sig=rColumn_Scan;
45
46  endmodule
```

9 编译结果

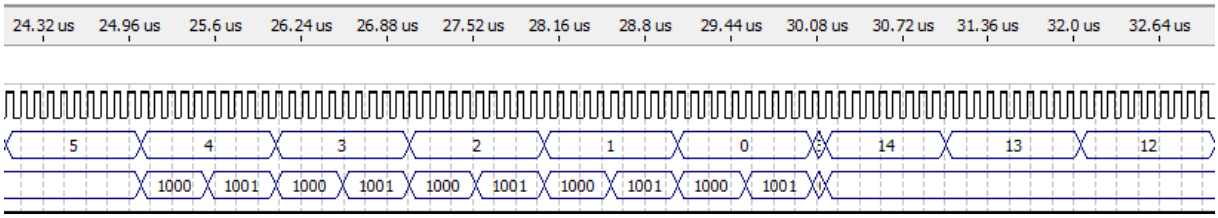
本次设计使用逻辑单元 339 个，寄存器 125 个，管脚 16 个（4 个灯信号输出，7 个数码管段选输出，2 个数码管位选输出，一个时钟输入和一个复位信号输入）。

Flow Summary	
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	traffic_light
Top-level Entity Name	traffic_light
Family	Cyclone IV E
Device	EP4CE6F17C7
Timing Models	Final
Total logic elements	339 / 6,272 (5 %)
Total combinational functions	338 / 6,272 (5 %)
Dedicated logic registers	125 / 6,272 (2 %)
Total registers	125
Total pins	16 / 180 (9 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

10　仿真结果



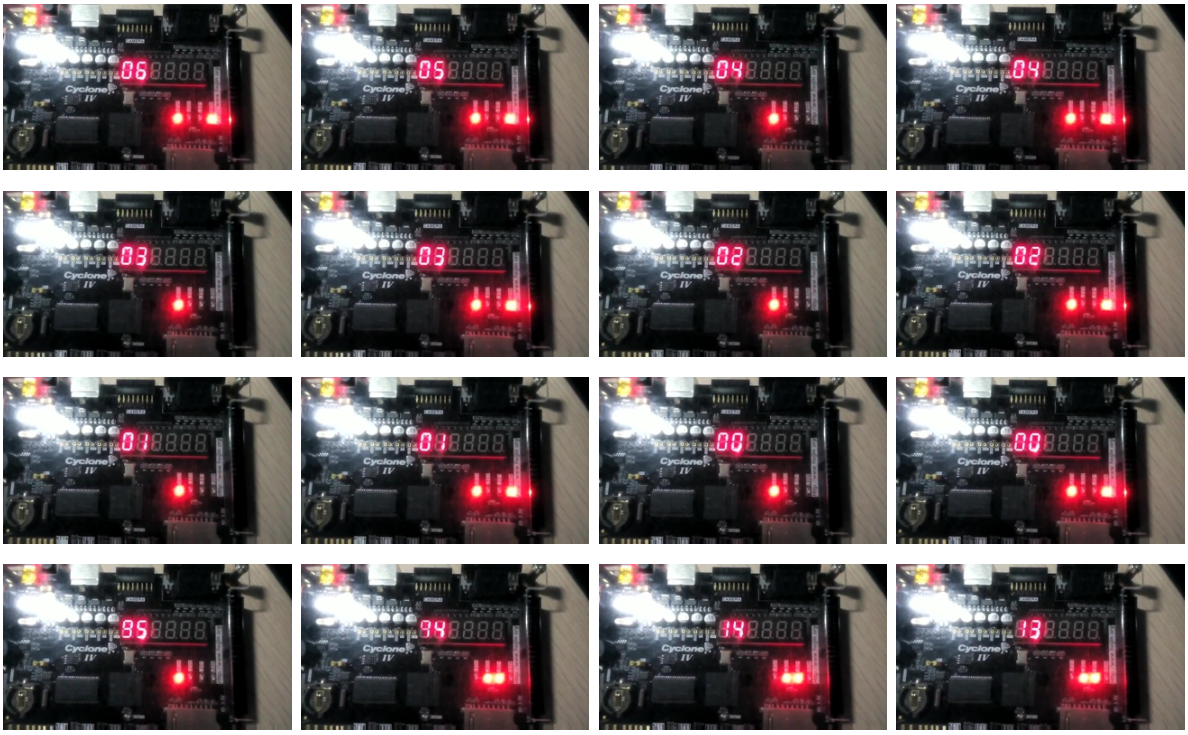
局部图：



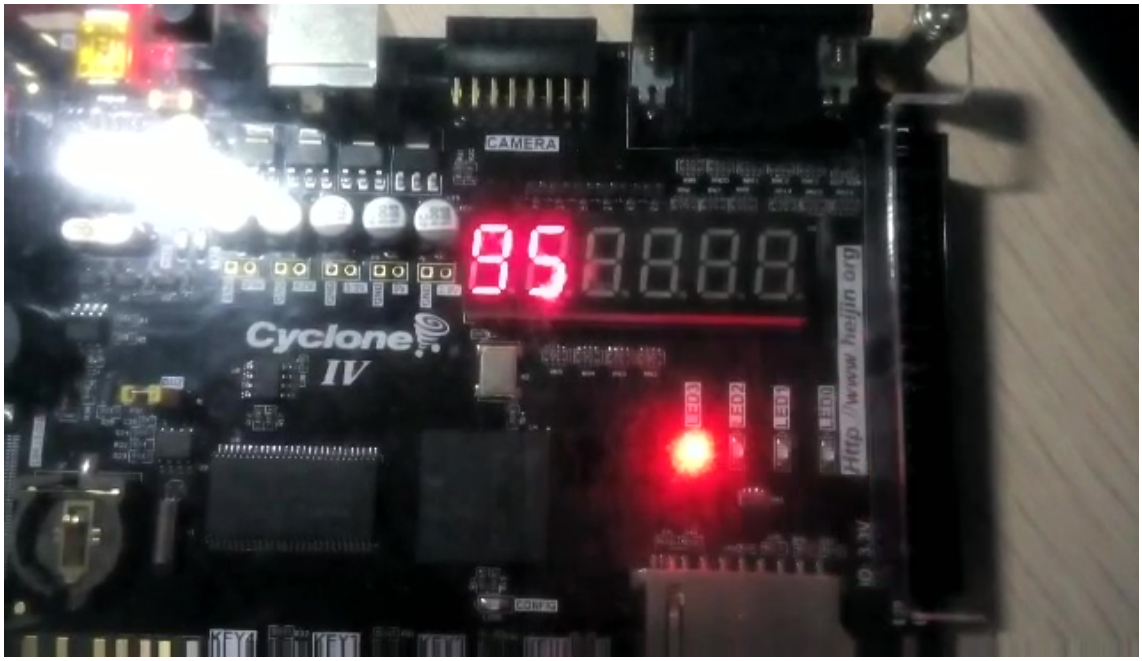
从波形图中，我们很容易看到随着时钟的输入，倒计时正常工作。当倒计时数到最后 5 秒时，灯输出信号不断的改变，也就是说绿灯在不断地闪烁，闪烁过程结束后，该车道的红灯亮绿灯灭，另一车道的红灯灭绿灯亮……周而复始，往复循环。

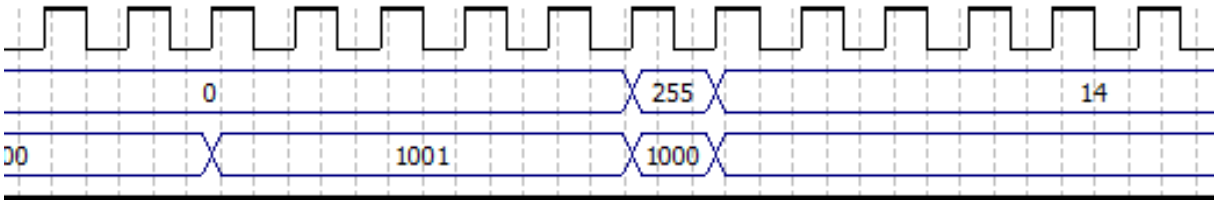
由于数码管显示模块无法仿真，在实际测试中给予展示。

11　测试结果



12　遇到的问题





在倒计时时候，X 车道从 Tx-1 秒倒计到 0 秒、切换到 Ty-1 秒是，会有短暂的“毛刺”毛刺，如仿真波形所示。

经过初步的分析，这是由于时序状态机在进行状态判断过程中会消耗一个时钟周期所造成的。在这个时钟周期内，状态进行判断，电路来不及变化，所以产生毛刺现象。不过这个“不正常”的现象并不影响精确的倒计时的时间，因为倒计时的时间总是由输入时钟来决定，跟输出电路无关。好在这个时间持续的很短，肉眼无法明显察觉，并且不影响设计功能，所以并无大碍。

希望随着以后更深入的学习，找到这个问题的关键，并解决它。