双面打印

组号

数字逻辑实验报告

实验 7

学　期	2022-2023 学年第 1 学期		实验日期	2023/03/01	
学　院	信息学部		专　业	计算机科学与技术（实验班）	
班　级	210710	学　号	21071003	姓　名	高立扬
组　号	43	学　号	21071004	姓　名	石昊阳

评　阅　内　容				
总体设计	详细设计	下　载	总　结	成　绩

题　目：	实验 7：实用电路设计（交通灯控制器）

一、功能描述

交通灯控制器实验电路的预期功能：

1.十字路口，东西方向、南北方向红黄绿交通灯各三盏，绿灯通行，红灯停止，黄灯过渡准备。红绿灯变化规律为：东西绿灯，南北红灯→东西黄灯，南北红灯→东西红灯，南北绿灯→东西红灯，南北黄灯→东西绿灯，南北红灯。

2.假设东西方向、南北方向两交通要道的通行时间控制基本相等。两组数码管作为东西、南北方向的倒计时显示，时间可以预置，如时间为红灯 59 秒、绿灯 56s，黄灯 3 秒。

3.具有复位功能，计数器恢复初始状态。

4.加入人互干预控制，使红绿灯在人工干预下可以停止计数并保持原来状态，东西，南北均为红灯状态，待特殊情况结束后能继续计数。

5.加入左转弯信号灯指示。

6.　自定义其它功能。

我们组的扩展功能：红绿灯倒计时快结束的时候，蜂鸣器开始响起间断性的蜂鸣，红灯、黄灯蜂鸣频率不同。红灯 10s 开始以 1hz 频率蜂鸣，黄灯 4s 开始以 2hz 频率蜂鸣。

二、总体设计

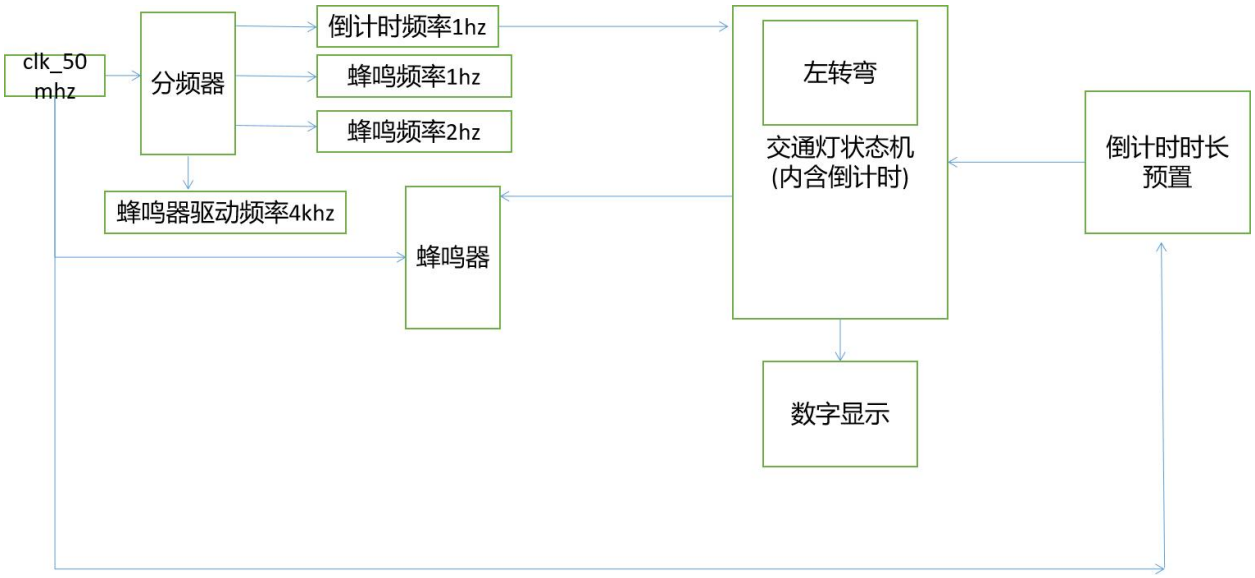


图 1.总体设计

三、详细设计

1、 交通灯状态机模块设计

交通灯内置了时间倒计时和左转弯状态，具体设计如下：

交通灯状态							
状态名	东西绿灯 南北红灯	东西黄灯 南北红灯	东西红灯 南北绿灯	东西红灯 南北黄灯	都是红灯	左转（1）	左转（2）
状态机编号	**001**	**010**	**100**	**101**	**111**	**011**	**110**

如图 3.每当红绿灯倒计时结束的时候，状态机就向下一状态推进，否则一直保持当前状态；需要 1 个按钮来负责复位功能，一个开关负责开启状态机，一个开关负责人工干预功能。

```
module main(reset, start, stop, clk, r, y, LSNG, LSNY, c_state, n_state, LSNR, LEWG,
    // start对应w_r的w模式
    input start, stop, clk, reset;
    input [5:0] r, y;
    output reg [2:0] c_state = 3'b111, n_state;
    // 模拟四种左转规则
    input [1:0] rules;
    output reg [1:0] turn_out;
    output reg LSNG, LSNY, LSNR, LEWG, LEWY, LEWR, left;

    output reg [5:0] r_count, y_count, l_count;
    // 判断状态转移专用
    output reg [31:0] condition;
    // 判断蜂鸣器频率专用——红灯快结束慢响，黄灯快结束快响
    output reg [1:0] ring;

    parameter l = 3'b111;
    parameter A=3'B001, B=3'B010, F=3'B011, C=3'B100, E=3'B111, D=3'B101, G=3'B110;
```

图 2.本模块所有的变量

```
// 状态转移
always @(c_state,start,n_state)
if(reset)
   if(start&  (!stop))
   begin
      case(c_state)
         A:if(r_count == 0 && y_count == y && l_count == l) n_state=B; else n_state=A;
         B:if(r_count == 0 && y_count == 0 && l_count == l) n_state=F; else n_state=B;
         F:if(r_count == 0 && y_count == 0 && l_count == 0) n_state=C; else n_state=F;
         C:if(r_count == 0 && y_count == y && l_count == l) n_state=D; else n_state=C;
         D:if(r_count == 0 && y_count == 0 && l_count == l) n_state=G; else n_state=D;
         G:if(r_count == 0 && y_count == 0 && l_count == 0) n_state=A; else n_state=G;
         E:n_state = A;
         default: n_state=n_state;
      endcase
   end
   else
      n_state = E;
else
   n_state = E;

// 状态更新
always @(posedge clk)
      c_state<=n_state;
```

图 3.状态机

如图 4.倒计时部分负责红绿灯的倒计时，一共有三个倒计时寄存器，分别负责红灯倒计时，黄灯倒计时和左转弯倒计时，当 stop 开关开启或者 reset 按钮按下，倒计时会锁定或重置。同时，通过倒计时的时间判断，该部分代码还会输出一个叫 ring 的一位二进制数字，来控制蜂鸣器的鸣叫。

```
if(reset)begin
   if(start&!stop)
   begin
   // ring
      if(r_count < 10 & y_count > 0 & r_count > 0) ring <= 2'b01;
      else if(r_count  == 0 & y_count < 5 & y_count > 0)    ring <= 2'b10;
      else  ring <= 2'b0;
   // count
      if(r_count > 0 & y_count > 0 & l_count > 0)
         r_count <= r_count - 1;
      else if(r_count == 0 & y_count > 0 & l_count > 0)
         y_count <= y_count - 1;
      else if(r_count == 0 & y_count == 0 & l_count > 0)
         l_count <= l_count - 1;
      else //if (r_count == 0 & y_count == 0 & l_count == 0)
      begin
         r_count <= r;
         y_count <= y;
         l_count <= l;
      end
   end
   else
      begin
         r_count <= r_count;
         y_count <= y_count;
         l_count <= l_count;
      end
end
else
begin
   r_count <= r;
   y_count <= y;
   l_count <= l;
   ring <= 2'b0;
end
```

图 4.倒计时模块

```
// 红绿灯
always@(n_state, start, stop)
if(start==1'b1 && stop == 0)
   begin
      case(n_state)
      A:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b1; LEWY = 1'b0; LEWR = 1'b0; left = 1'b1; end
      B:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b0; LEWY = 1'b1; LEWR = 1'b0; left = 1'b1; end
      F:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b1; end
      C:begin LSNG = 1'b1; LSNY = 1'b0; LSNR = 1'b0; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b1; end
      D:begin LSNG = 1'b0; LSNY = 1'b1; LSNR = 1'b0; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b1; end
      E:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b1; end
      G:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b0; end
      //F:begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b1; LEWY = 1'b0; LEWR = 1'b0; end
      default: begin LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b0; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b0; left = 1'b1; end
      endcase
   end
else
   begin
      LSNG = 1'b0; LSNY = 1'b0; LSNR = 1'b1; LEWG = 1'b0; LEWY = 1'b0; LEWR = 1'b1; left = 1'b1;
   end
```

图 5.红绿灯根据状态而亮起

如图 5.红绿灯如何亮起，取决于状态机的状态如何，为了消除 c_state 的滞后导致红绿灯转换滞后，这里选择通过 n_state 来判断红绿灯的转换。

2、 红绿灯时长预置模块设计

为了方便实验的演示，预置时长最多为 20 秒，左转弯倒计时固定位 7 秒，红黄绿灯的预置简化为红黄灯的预置（因为预置东西红灯相当于预置南北绿灯），用到 1 个按钮负责时长+1，1 个开关负责切换红黄两个灯的倒计时选择预置哪个，预置模式需要 1 个开关来激活。

```
input clk,
input cs,
input w_r,
input addr,
input button,
output reg [6 - 1:0] w_data,
output reg [6 - 1:0] array_reg1,
output reg [6 - 1:0] array_reg2
);


always@(posedge clk)
begin
   if(cs&&!w_r)
      begin
      case(addr)
         0:array_reg1 <= w_data;
         1:array_reg2 <= w_data;
         default:begin array_reg1 <= array_reg1; array_reg2 <= array_reg2; end
      endcase
      end
   else
      begin
      array_reg1 <= array_reg1;
      array_reg2 <= array_reg2;
      end
end

always@(negedge button)
   if(w_data < 20)
      w_data <= w_data + 1;
   else
      w_data <= 0;
```

图 6.时长预置模块

```
module buzz(clk, beep_r, clk_slow, clk_fast, ring, beep);
   // beep信号允许蜂鸣器进行取反操作
   // clk信号是工作频率信号
   input clk, clk_slow, clk_fast;
   input [1:0] ring;
   output reg beep;
   output reg beep_r;

   // 蜂鸣器beep时长
   always@(ring, clk_slow, clk_fast)
   begin
      if(ring == 2'b01)
         beep = clk_slow;
      else if(ring == 2'b10)
         beep = clk_fast;
      else
         beep = 0;
   end

   always@(posedge clk)
      if(beep)
         beep_r <= !beep_r;
      else
         beep_r <= beep_r;
endmodule
```

图 7.蜂鸣器模块

3、 分频器模块设计

同实验三的分频器

4、 蜂鸣器模块设计

根据倒计时来决定响的频率，蜂鸣器驱动需要 2k-4k hz 之间的频率，发出声音的原理是：在 2k-4k 的频率下，蜂鸣器不断接收到此频率的 01010...信号。落实在编程，可以通过一个 1’b ring 信号控制 1’b beep 信号接收 1hz 频率或 2hz 频率；随着 50mhz 的时钟，beep_r 信号根据 beep 不断取反并传输给蜂鸣器实现持久蜂鸣；其他情况 beep_r 就会停止取反，蜂鸣器接收不到持续取反的 beep，因此静音。beep_r 按照一定频率变化，可以实现持续蜂鸣变为间断的蜂鸣

5、 红绿灯倒计时数字显示模块设计

同七段数码管译码器。改进如下：

二位数显示：配合频扫，使传入数码管译码器的数据不断被赋值为十位和个位

在红绿灯扩展版上的数字显示：由于倒计时的逻辑是，当红黄灯和左转弯灯的倒计时都为 0 的时候，状态机才会转回 A 状态，三个倒计时才会重置。因此扩展板的倒计时数字显示可以通过判断倒计时是否为 0 来决定显示哪个倒计时，比如红灯倒计时大于零的时候，说明此时一定是红灯状态，如果红灯倒计时为 0，但是黄灯倒计时不为 0，则此时一定是黄灯状态。

```
module test02A (sel,out, x, a, b);
   input [1:0] sel;
   output [3:0] out;
   input [5:0] x;
   reg [3:0] out;
   output [3:0] a;
   output [3:0] b;

   assign a=x/10;
   assign b=x%10;

always @ (sel)
   begin
      case(sel)
         2'b00 : out = b;
         2'b01 : out = a;
      endcase
   end
endmodule
```

图 8.二位数显示

```
always@(x)
if(x > 0)
begin
    a=x/10;
    b=x%10;
end
else if(y>0)
begin
 a=y/10;
 b=y%10;
end
else
begin
 a=0;
 b=z;
end
```

图 9.扩展版数字显示

6、 顶层设计

我们采用文字编程来编写顶层文件，下面展示除变量声明之外的代码。通过所有模块的先后和逻辑顺序，来编写顶层文件。

```
48    frequency_divider(.clk_50mhz(clk_50mhz), .clk_1hz(clk_1hz), .clk_4khz(clk_4khz), .clk_2hz(clk_2hz), .clk_100hz(clk_100hz));
49    pretime(.clk(clk_50mhz), .cs(cs), .w_r(w_r), .addr(addr), .button(key), .array_reg1(array_reg1), .array_reg2(array_reg2), .w_data(w_data));
50    main(.start(w_r), .stop(stop), .clk(clk_1hz), .r(array_reg1), .y(array_reg2), .LSNG(LSNG), .LSNY(LSNY), .c_state(c_state), .n_state(n_state
51                .LSNR(LSNR), .LEWG(LEWG), .LEWY(LEWY), .LEWR(LEWR), .rules(rules), .turn_out(turn_out), .left(left), .ring(ring),.r_count(r_cou
52
53    buzz(.clk(clk_4khz), .beep_r(beep_r), .clk_slow(clk_1hz), .clk_fast(clk_2hz), .ring(ring), .beep(beep));
54
55    scan(.clk(clk_100hz), .ds(ds), .sel(sel));
56    test02(.sel(sel), .en(w_r), .out(out), .x(r_count), .y(y_count), .z(l_count) ,.a(a), .b(b));
57    LED(.in(out), .en(w_r), .out(led));
58
59    scanA(.clk(clk_100hz), .ds(dsa), .sel(sela));
60    test02A(.sel(sela), .out(outa), .x(w_data), .a(c), .b(d));
61    LEDA(.in(outa), .out(leda));
62
```

图 10.顶层文件代码

四、下载调试

1、引脚分配

表 1.引脚分配

端口名称	输入端								
	Clk_50mHZ	cs	W_r	addr	stop	Rules[0]	Rules[1]	key	reset
引脚编号	T1	M20	N18	AA15	V13	F8	E7	AA22	R18
平台端口	T1	SW2	SW1	SW3	SW4	SW8	SW7	F6	F10

表 1.续表

端	输出端

口名称	LEWG	LEWR	LEWY	LSNG	LSNR	LSNY	Beep_r	Ds[1]	Ds[0]
引脚编号	E13	A7	F13	F10	B16	D13	A8	B3	B4
平台端口	LEWG	LEWR	LEWY	LSNG	LSNR	LSNY	BUZZ	SEG_H1	SEG_H2

表 1.续表

端口名称	输出端								
	Dsa[1]	Dsa[0]	Led[6]	Led[5]	Led[4]	Led[3]	Led[2]	Led[1]	Led[0]
引脚编号	V16	AA17	N19	P20	W14	AB16	Y13	AB14	R19
平台端口	DS8	DS7	SEC_A	SEC_B	SEC_C	SEC_D	SEC_E	SEC_F	SEC_G

表 1.续表

端口名称	输出端							
	left	Leda[6]	Leda[5]	Leda[4]	Leda[3]	Leda[2]	Leda[1]	Leda[0]
引脚编号	U12	AA20	W20	R21	P21	N21	N20	M21
平台端口	LED1	LA	LB	LC	LD	LE	LF	LG

Node Name	Direction	Location
LEWG	Output	PIN_E13
LEWR	Output	PIN_A7
LEWY	Output	PIN_F13
LSNG	Output	PIN_F10
LSNR	Output	PIN_B16
LSNY	Output	PIN_D13
a[3]	Output	
a[2]	Output	
a[1]	Output	
a[0]	Output	
addr	Input	PIN_AA15
array...g1[5]	Output	
array...g1[4]	Output	
array...g1[3]	Output	
array...g1[2]	Output	
array...g1[1]	Output	
array...g1[0]	Output	
array...g2[5]	Output	
array...g2[4]	Output	
array...g2[3]	Output	
array...g2[2]	Output	
array...g2[1]	Output	
array...g2[0]	Output	
b[3]	Output	
b[2]	Output	
b[1]	Output	
b[0]	Output	
beep	Output	
beep_r	Output	PIN_A8
clk_50mhz	Input	PIN_T1
cs	Input	PIN_M20
ds[1]	Output	PIN_B3
ds[0]	Output	PIN_B4
dsa[1]	Output	PIN_V16
dsa[0]	Output	PIN_AA17
key	Input	PIN_AA22
l_count[5]	Output	
l_count[4]	Output	
l_count[3]	Output	
l_count[2]	Output	
l_count[1]	Output	
l_count[0]	Output	
led[6]	Output	PIN_N19
led[5]	Output	PIN_P20
led[4]	Output	PIN_W14
led[3]	Output	PIN_AB16
led[2]	Output	PIN_Y13
led[1]	Output	PIN_AB14
led[0]	Output	PIN_R19
leda[6]	Output	PIN_AA20
leda[5]	Output	PIN_W20
leda[4]	Output	PIN_R21
leda[3]	Output	PIN_P21
leda[2]	Output	PIN_N21
leda[1]	Output	PIN_N20
leda[0]	Output	PIN_M21
left	Output	PIN_U12
out[3]	Output	
out[2]	Output	
out[1]	Output	
out[0]	Output	
r_count[5]	Output	
r_count[4]	Output	
r_count[3]	Output	
r_count[2]	Output	
r_count[1]	Output	
r_count[0]	Output	
reset	Input	PIN_R18
ring[1]	Output	
ring[0]	Output	
rules[1]	Input	PIN_E7
rules[0]	Input	PIN_F8
stop	Input	PIN_V13
turn_out[1]	Output	
turn_out[0]	Output	
w_data[5]	Output	
w_data[4]	Output	
w_data[3]	Output	
w_data[2]	Output	
w_data[1]	Output	
w_data[0]	Output	
w_r	Input	PIN_N18

图 11.图 12.pin planner 分配图（没分配引脚的变量是验证变量功能的时候残留下来的）

2、实验现象

实验开始后将 CS 开关开启，按下 key 按钮预制红灯，开启 adrr 开关摁下 key 预制黄灯。预制结束后将 CS 开关关闭，W_r 开关开启，红绿灯开始工作。红灯倒计时结束后进入黄灯状态，黄灯倒计时结束后进入左转状态，左转倒计时结束后进入红灯状态，并依次循环。在黄灯倒计时三秒内，及红灯倒计时十秒内，蜂鸣器依次以较快的频率与较慢的频率蜂鸣。开启 stop 开关，将会亮起所有红灯并暂停倒计时。关闭 stop 开关，红绿灯按原状态继续运行。按下 reset 摁钮，红绿灯会回到初始状态。

五、问题总结

本次实验是数字逻辑的最终大实验，我们在进行实验的过程中遇到了很多困难，但是都一一解决，最后在规定时间内顺利完成了任务。

①顶层文件编写的时候，我们出现了传错变量的情况，通过老师的指导，我们发现了错误赋值的变量并及时改正；

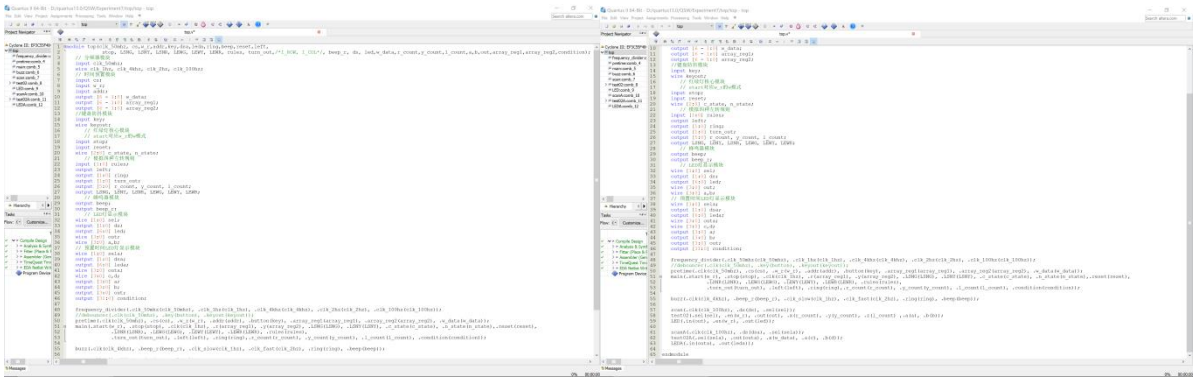
②由于实验涉及了四种不同的频率，因此跑波形图的时候会出现没波形的情况，如果通过波形图来验证实验的话，需要把所有的频率统一为 50mhz。

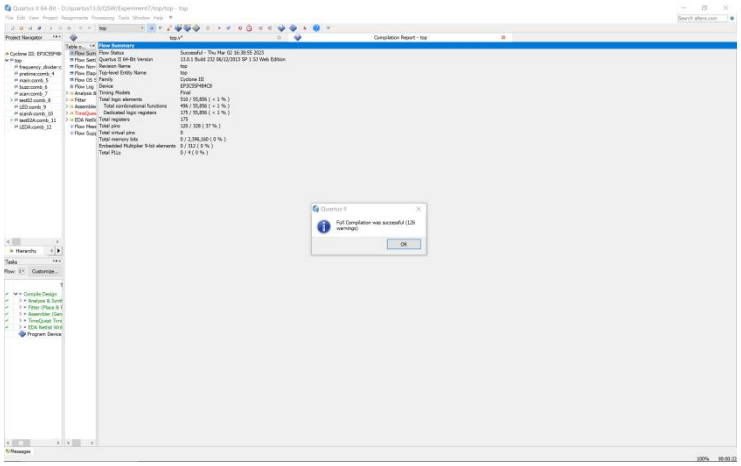
③预置时间的时候，我们的代码没有有效利用 if 语句，导致变量繁多复杂，无法赋值。通过不断演算，我们简化了代码，实现了预期的功能。

④状态机模块出错最多。一开始是状态转换逻辑出错，后来检查发现是倒计时模块有错，通过演算和仿真，我们去除了冗余的逻辑和变量，有效利用倒计时逻辑和 if 语句实现了预期功能。

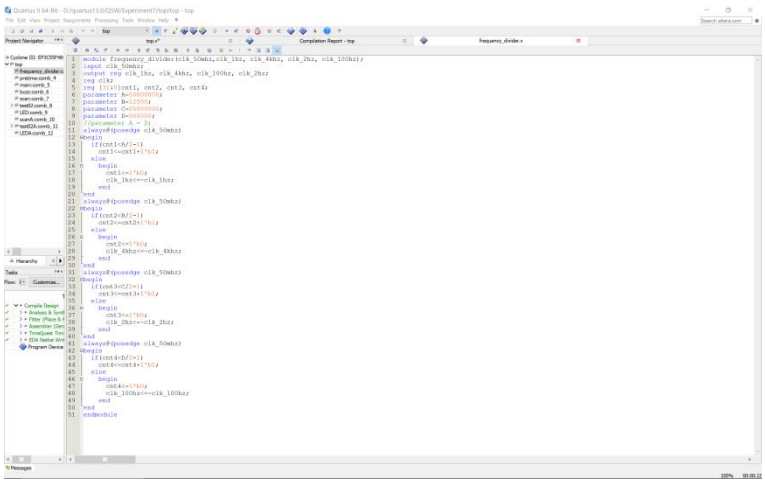
六、任务分配

模块	完成人
红绿灯状态机	高立扬
时间预置	高立扬
分频器	石昊阳
数字显示	石昊阳
蜂鸣器	石昊阳
顶层设计	高立扬

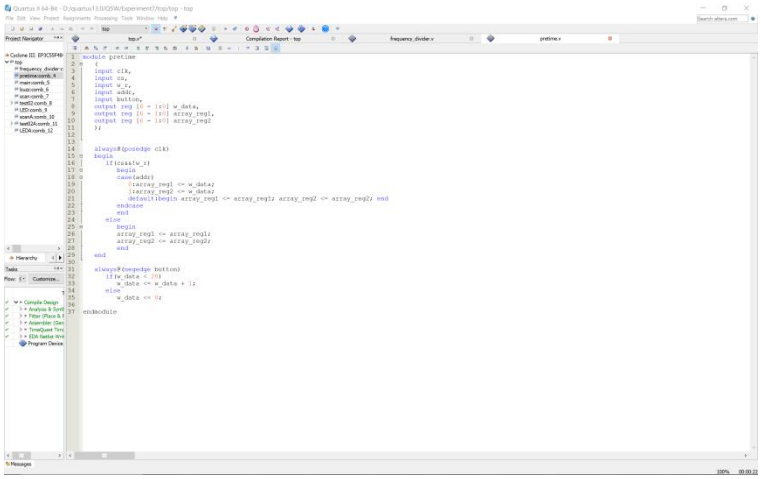




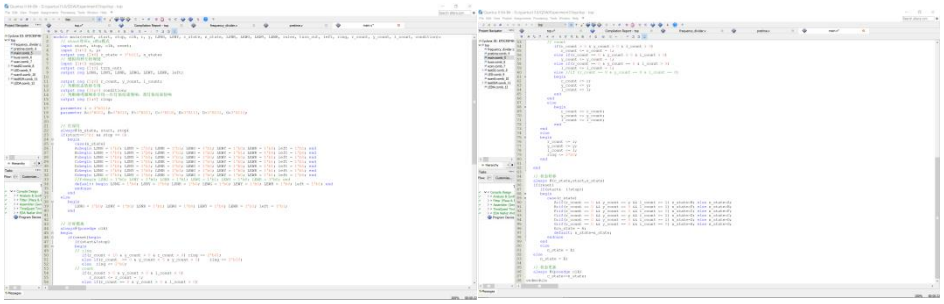
附图 1：顶层电路及编译成功信息



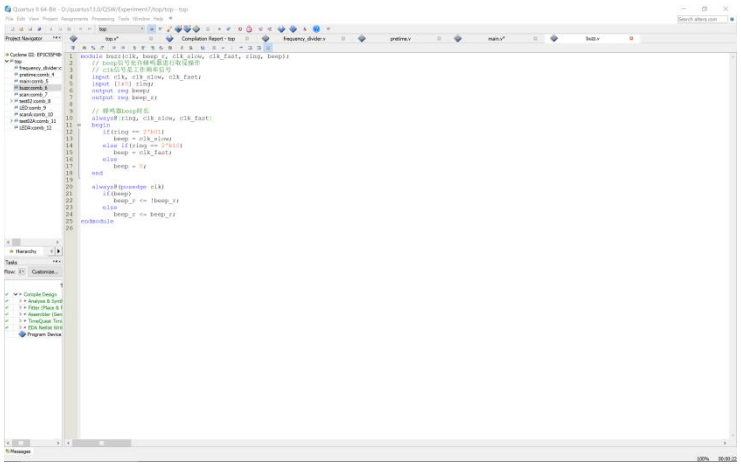
附图 2：分频器模块代码



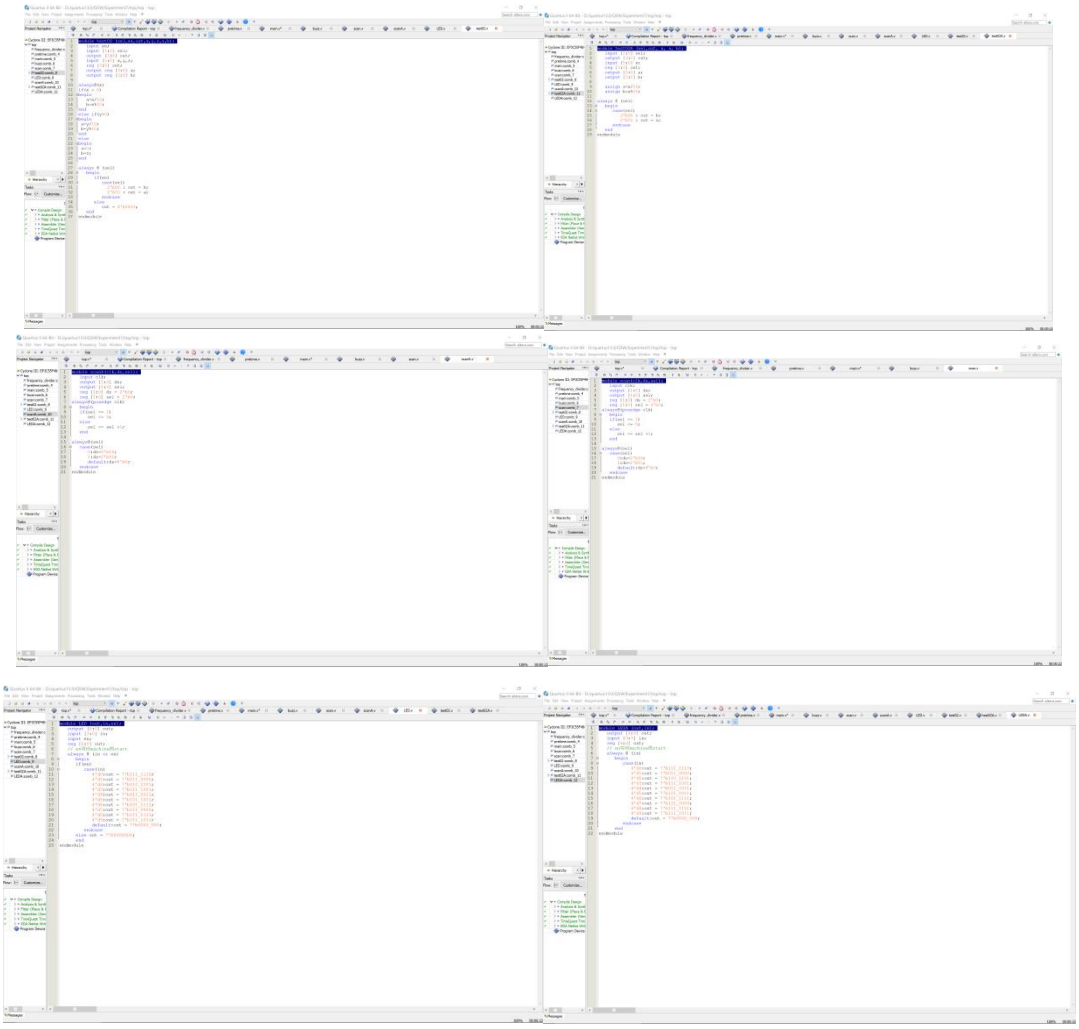
附图 3：预置时间模块代码



附图 4：状态机模块代码



附图 5：蜂鸣器模块代码



附图 6：LED 模块代码