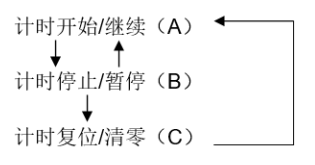
**实验四 综合电路设计实验报告**

1. **实验题目**

* 短跑计时器设计与实现（难度系数：0.9）

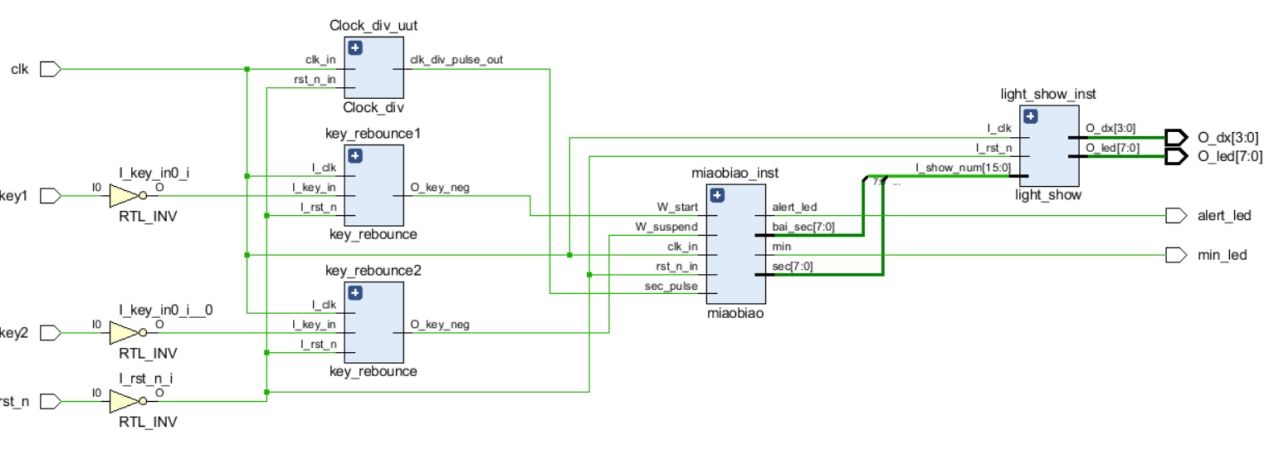
短跑计时器描述如下：

* + - 短跑计时器显示分、秒、毫秒；
    - “毫秒”用两位数码管显示：百位、十位；
    - “秒”用两位数码管显示：十位、个位；
    - “分”用一位 LED 灯显示，LED 灯“亮”为 1 分；
    - 最大计时为 1 分 59 秒 99，超限值时应可视或可闻报警；
    - 三个按键开关：计时开始/继续（A）、计时停止/暂停（B）、复位/清零（C），
    - 键控流程如下：



1. **电路设计**

如下图所示，该电路设计主要分为时钟分频模块（Clock\_div）、按键消抖模块(key\_rebounce)、秒表控制模块(miaobiao)、数码管显示模块(light\_show)。



1. **电路实现**

**3.1 时钟分频模块（Clock\_div）**

该模块主要是产生10ms单位的脉冲，实现功能为，当计数器cnt记到10ms时，clk\_div\_pulse\_out输出一个时钟周期的高电平。

主要程序如下：

reg [27:0] cnt=0;

always@(posedge clk\_in or negedge rst\_n\_in)

begin

if(!rst\_n\_in) begin

cnt<=0;

end else if(cnt>=(CLK\_DIV\_PERIOD-1)) begin

cnt<=0;

end else begin

cnt<=cnt+1'b1;

end

end

assign clk\_div\_pulse\_out = (cnt==(CLK\_DIV\_PERIOD-1))? 1'b1:1'b0;//输出高电平

**3.2 按键去抖模块(key\_rebounce)**

由于按键按下与松开时，会有抖动，需要滤除抖动，只产生一个按键信号，所以采用的原理是，当检测到按键值有变化时，会产生W\_change信号为1，此时，将延时计数器清零，当稳定后，延时计数器稳定计数，记到设定的延时阈值后，对按键值采样，从而滤除了按键抖动。

主要实现程序如下：

parameter C\_COUNTER\_NUM = 180000;

always@(posedge I\_clk or negedge I\_rst\_n)

if(!I\_rst\_n)//复位处理

R\_key\_in0 <= 1;

else //记录按键输入

R\_key\_in0 <= I\_key\_in;

//如果前后两个时钟按键输入数据不同，将此信号置为1

assign W\_change=(I\_key\_in & !R\_key\_in0)|(!I\_key\_in & R\_key\_in0);

always@(posedge I\_clk or negedge I\_rst\_n)

if(!I\_rst\_n) //复位处理

R\_count <= 0;

else if(W\_change)//按键输入发生改变，重新开始计数

R\_count <= 0;

else

R\_count <= R\_count + 1;

always@(posedge I\_clk or negedge I\_rst\_n)

if(!I\_rst\_n) //复位处理

O\_key\_out <= 1;

else if(R\_count >= C\_COUNTER\_NUM - 1)//更改输出信号

O\_key\_out <= I\_key\_in;

reg [1:0] O\_key\_out\_r;

always@(posedge I\_clk or negedge I\_rst\_n)

if(!I\_rst\_n) //复位处理

begin

O\_key\_out\_r <= 2'b11;

end

else

O\_key\_out\_r <= {O\_key\_out\_r[0],O\_key\_out};

assign O\_key\_neg = (~O\_key\_out\_r[0] ) && O\_key\_out\_r[1];

**3.3 秒表控制模块(miaobiao)**

主要实现对秒表的开始、暂停控制，还有秒表计时。控制原理为当W\_start为高，则stop信号拉高，当W\_suspend为高时，stop拉低，否则stop信号保持，从而产生stop为高时，下面的秒表计数有效，否则无效，从而控制启停。

定义sec[7:0]与bai\_sec[7:0]，高4位与低4位，分别计数进位。程序如下所示，当计数到1：59：99时，停止计数，亮灯报警。

主要程序如下：

always@(posedge clk\_in or negedge rst\_n\_in)

begin

if(!rst\_n\_in)

stop <=0;

else

if(W\_start)

stop <= 1;//控制启动与暂停

else if(W\_suspend)

stop <= 0;

end

always@(posedge clk\_in or negedge rst\_n\_in) begin

if(!rst\_n\_in)

begin

min<=1'b0;

sec<=8'h0;

bai\_sec<=8'h00;

end

else begin

if(sec\_pulse & stop & (~alert\_led)) begin //当有效的情况下，开始计数，否则停止

if(bai\_sec==8'h99) begin //记到99，则清零

bai\_sec<=8'h0;

if(sec==8'h59) begin //记到59，则清零

sec<=8'h0;

min <= 1'b1;

end else begin //秒计数

if(sec[3:0]==4'd9) begin

sec[3:0]<=4'd0;

sec[7:4]<=sec[7:4]+4'd1;

end else sec[3:0]<=sec[3:0]+4'd1;

end

end else begin //毫秒计数

if(bai\_sec[3:0]==4'd9) begin

bai\_sec[3:0]<=4'd0;

bai\_sec[7:4]<=bai\_sec[7:4]+4'd1;

end else bai\_sec[3:0]<=bai\_sec[3:0]+4'd1;

end

end

end

end

always@(posedge clk\_in or negedge rst\_n\_in)

begin

if(!rst\_n\_in)

alert\_led <= 1'b0;

else if((min == 1) && (sec==8'h59) && (bai\_sec==8'h99)) //比较到达报警值

alert\_led <= 1'b1;

end

**3.4数码管显示模块(light\_show)**

主要实现数码管的动态显示，首先定义时间延时参数C\_COUNTER\_NUM，当计数器R\_counter记到延时参数后，依次循环点亮4个数码管，同时循环切换显示不同的值，实现了数码管的动态显示。

主要程序代码如下：

parameter C\_COUNTER\_NUM = 100\_000;

reg [3:0] R\_temp; //当前显示的4-bit数据寄存器

reg [3:0] R\_dx\_temp;

reg [31:0] R\_counter;

always@(posedge I\_clk or negedge I\_rst\_n)

begin

if(!I\_rst\_n)

begin //复位处理

R\_dx\_temp <= 4'b0001;

R\_temp <= I\_show\_num[3:0];

R\_counter <=0;

end //依次循环显示4个数码管

else if(R\_dx\_temp == 4'b0001 && R\_counter >= C\_COUNTER\_NUM )

begin

R\_dx\_temp <= 4'b0010;

R\_temp <= I\_show\_num[7:4];

R\_counter <=0;

end

else if(R\_dx\_temp == 4'b0010 && R\_counter >= C\_COUNTER\_NUM )

begin

R\_dx\_temp <= 4'b0100;

R\_temp <= I\_show\_num[11:8];

R\_counter <=0;

end

else if(R\_dx\_temp == 4'b0100 && R\_counter >= C\_COUNTER\_NUM )

begin

R\_dx\_temp <= 4'b1000;

R\_temp <= I\_show\_num[15:12];

R\_counter <=0;

end

else if(R\_dx\_temp == 4'b1000 && R\_counter >= C\_COUNTER\_NUM )

begin

R\_dx\_temp <= 4'b0001;

R\_temp <= I\_show\_num[3:0];

R\_counter <=0;

end

else

R\_counter <=R\_counter + 1;

end

1. **电路验证**
   1. **TestBench**

仿真测试程序如下所示：

首先产生时钟信号clk，复位信号rst\_n，在initial 块中对按键key1模拟按下，则启动秒表，延时一段时间后，key2按下，则暂停秒表，过一段时间后，再次按下key1，启动秒表。

module top\_tb();

reg clk=0;

reg rst\_n=1; //key3

reg key1=0;

reg key2=0;

wire [6:0] O\_led;

wire [3:0] O\_dx ;

wire min\_led;

wire alert\_led;

always #5 clk = ~clk;

initial

begin

#1000;

rst\_n = 0;

#1000;

key1 = 1;

#1000;

key1 = 0;//启动

#100\_000\_00;

key2 = 1;

#1000;

key2 = 0;//暂停

#100\_000\_00;

key1 = 1;//启动

#1000;

key1 = 0;

#100\_000\_00;

$stop;

end

defparam i1.Clock\_div\_uut.CLK\_DIV\_PERIOD =100; //参数仿真定义

defparam i1.key\_rebounce1.C\_COUNTER\_NUM = 5;

defparam i1.key\_rebounce2.C\_COUNTER\_NUM = 5;

defparam i1.light\_show\_inst.C\_COUNTER\_NUM=10;

top i1(

.clk (clk ),

.rst\_n (rst\_n ), //key3

.key1 (key1 ),

.key2 (key2 ),

.O\_led (O\_led ),

.O\_dx (O\_dx ),

.min\_led (min\_led ),

.alert\_led (alert\_led )

);

endmodule

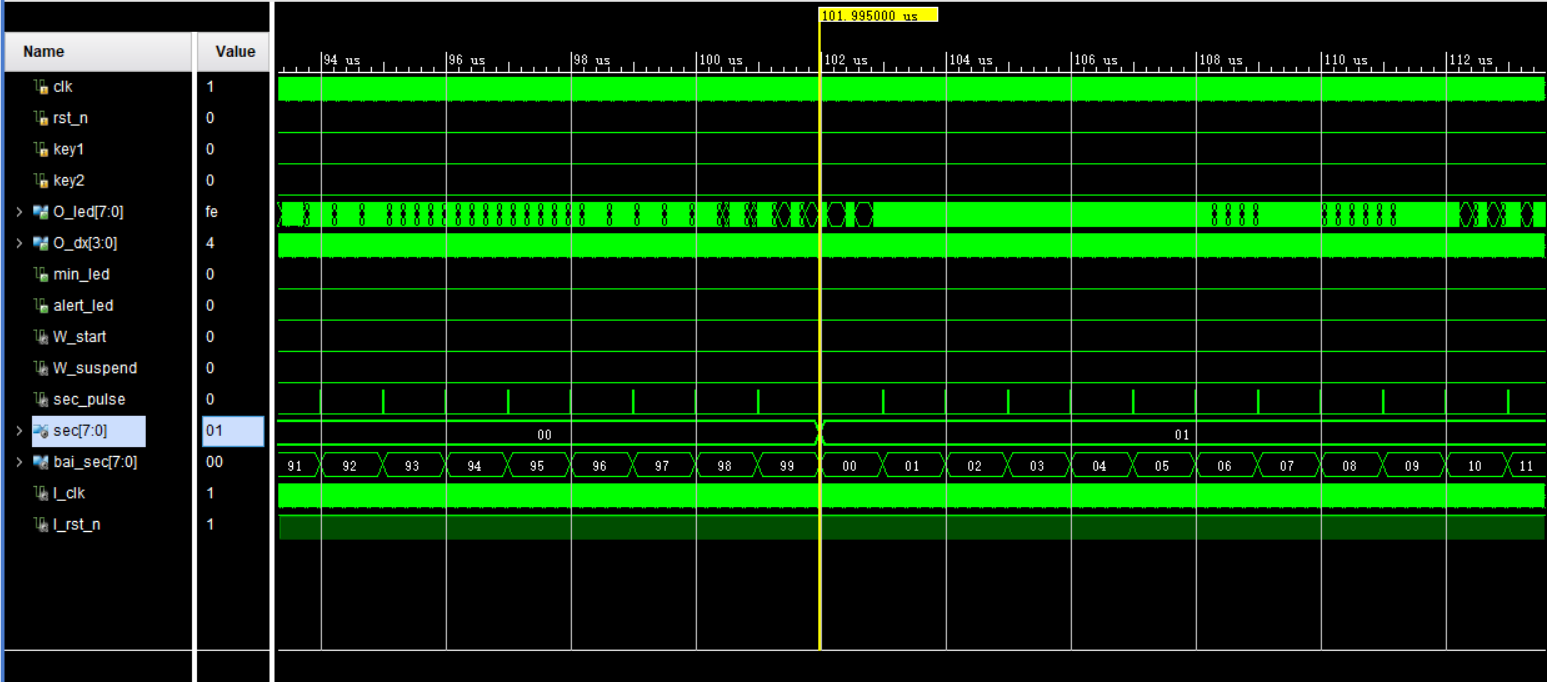
* 1. **仿真结果**

从下仿真图1可以看到，当按下按键key1后，秒表开始计数，随着sec\_pulse为高，bai\_sec依次加1。



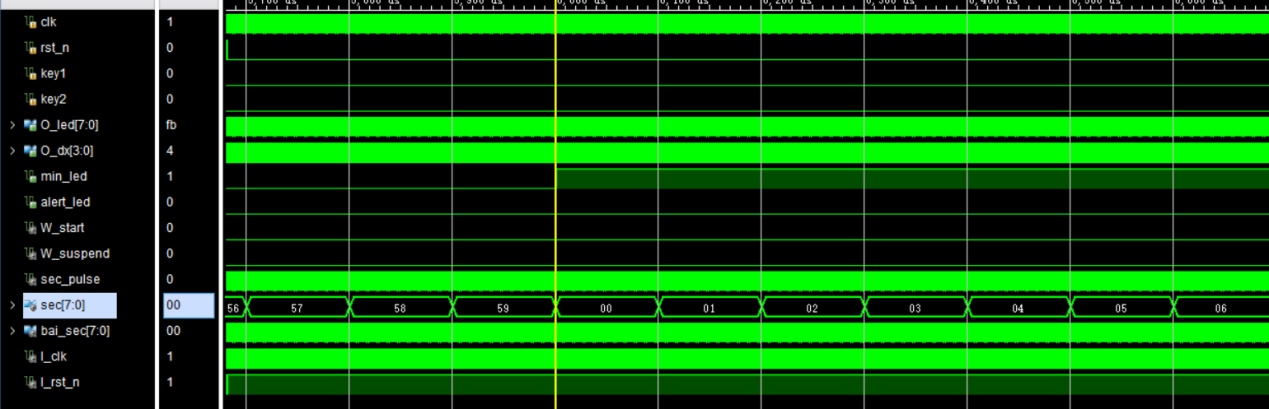
仿真图1

如下仿真图2所示，当bai\_sec= 99时，sec加1 。



仿真图2

如下仿真图3所示，当sec= 59时，min\_led为1 ，代表1分时间到。



仿真图3

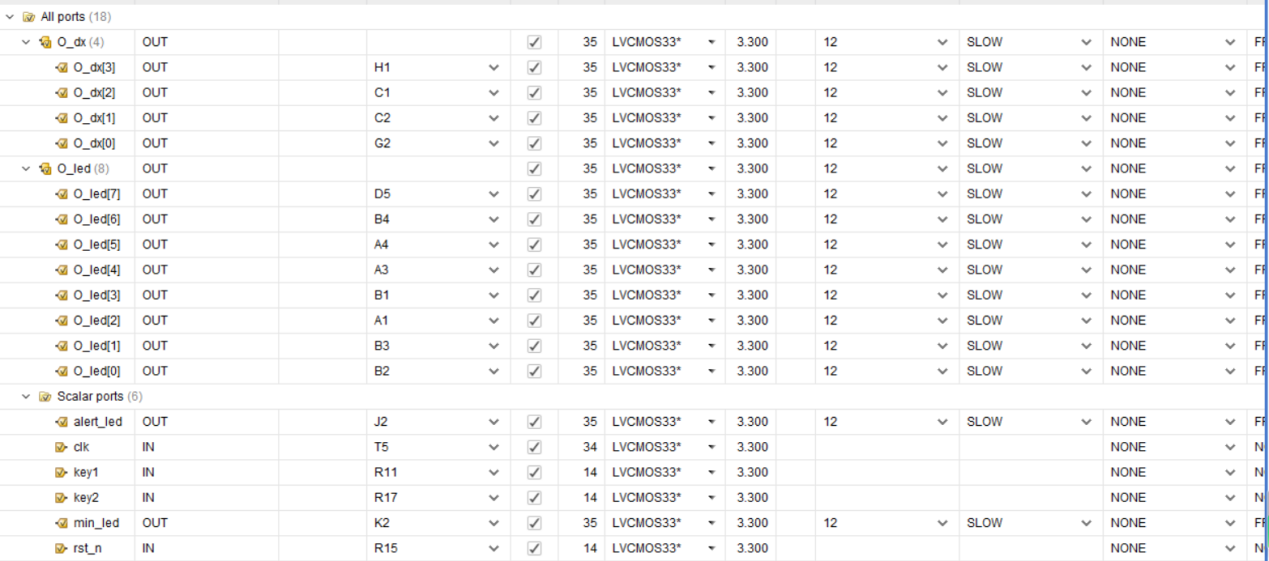
如下仿真图4所示，当计到1:59:99时，alert\_led为1 ，启动报警，暂停计时。同时可以看到，数码管在循环扫描。



仿真图4

1. **电路上板**

管脚配置：



上板情况：

上板后初始状态为秒和毫秒的四位数码管亮起，显示00’00。

按下开始键后，计时器开始计时，记录秒和毫秒的数码管显示对应数字。计时达到一分钟后，分钟灯亮起，计时达到1’59’99后，警报灯亮起，计时器停止计时。

计时过程中，随时按下暂停键，将会暂停计时，秒和毫秒的数码管显示的数字停止变动，暂停状态按下开始键会继续计时。暂停状态下或开始状态下按下复位键，计时器将回到初始状态，若分钟灯/警报灯亮起则全部熄灭，秒和毫秒的四位数码管显示00’00。