中山大學
SUN YAT-SEN UNIVERSITY

实验报告

多功能数字钟的设计

学　　院：数据科学与计算机学院

专　　业：软件工程（移动信息工程）

学生姓名：张 燕 梅

指导教师：保 延 翔

完成时间：2016 年 12 月 29 日

多功能数字钟的设计实验报告

摘要

本文利用 Verilog HDL 语言的设计方法设计多功能数字钟，并通过 vivado 2016.3 完成综合实现。此程序通过下载到 FPGA 芯片后，可应用于实际的数字钟显示中，实现了基本的计时显示（时分到分秒的切换）和设置，调整时间，闹钟设置的功能。

关键词：FPGA、Verilog HDL、数字钟

一、实验要求

设计一个多功能数字时钟，要求如下：
1、计时功能：包括时、分、秒的计时（时分和分秒可通过按键切换）；
2、清零功能：高电平时实现清零功能，低电平时正常计数；
3、具备校时功能，可以设置当前时间；
4、具备定时启动闹钟功能，可以设置启动闹钟时间，并通过 LED 闪烁来闹铃；
5、具备秒表计时功能；

二、各模块功能说明

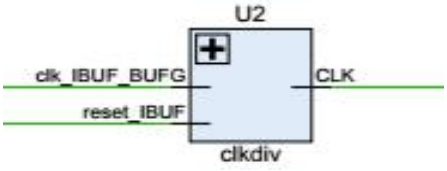
1、分频器模块

（1）产生 190HZ 信号用于数码管的扫描显示

由于 FPGA 内部提供的时钟信号频率大约为 100MHZ，在这个模块中将它转化为 190HZ 的信号用于数码管的扫描显示。

```
module clkdiv(
    input clk,
    input clr,
    output clk190
    );
    reg [40:0]q;// 41 位计数器
    always@(posedge clk or posedge clr)
    begin
        if(clr==1)
            q<=0;
        else
            q<=q+1;
     end
     assign clk190 = q[17];// 190 Hz 用于数码管的扫描显示
endmodule
```

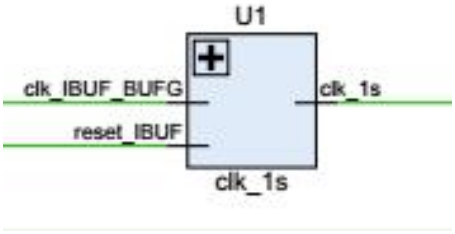
该模块生成的 RTL 图如下：



（2）产生 1 秒标准时钟信号

```
module clk_1s #(parameter N=99_999999)(
    input wire clk,reset,     //clk 接 w5
    output reg clk_1s
    );
    reg [31:0] count;

    always@(posedge clk, posedge reset)
    begin
        if(reset==1)
            count<=0;
        else
        begin
            clk_1s <= 0;
            if(count<N)
                count<=count+1;
            else
            begin
                clk_1s<=1;
                count<=0;
            end
        end
    end
endmodule
```

该模块生成的 RTL 图如下：



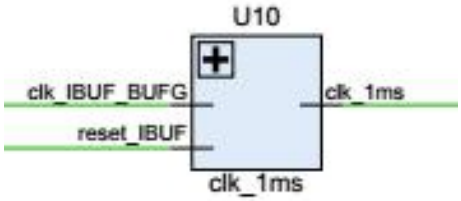
（3）产生 1 毫秒标准时钟信号

```
module clk_1ms #(parameter N=99_999)(
    input wire clk,reset,     //clk 接 w5
    output reg clk_1ms
    );
    reg [31:0] count;

    always@(posedge clk, posedge reset)
    begin
        if(reset==1)
            count<=0;
```

```
        else
        begin
            clk_1ms <= 0;
            if(count<N)
                count<=count+1;
            else
```

```
begin
                clk_1ms<=1;
                count<=0;
            end
        end
    end
endmodule
```

该模块生成的 RTL 图如下：



2、计时模块

计时模块是对 1 秒的信号源进行秒计时，计时满后向上进位的设计思想；当 reset 键为低电平时，通过采用十进制数计时，分别对时、分、秒三个数进行取余数、取除数，分别得到三位数的高位和低位。

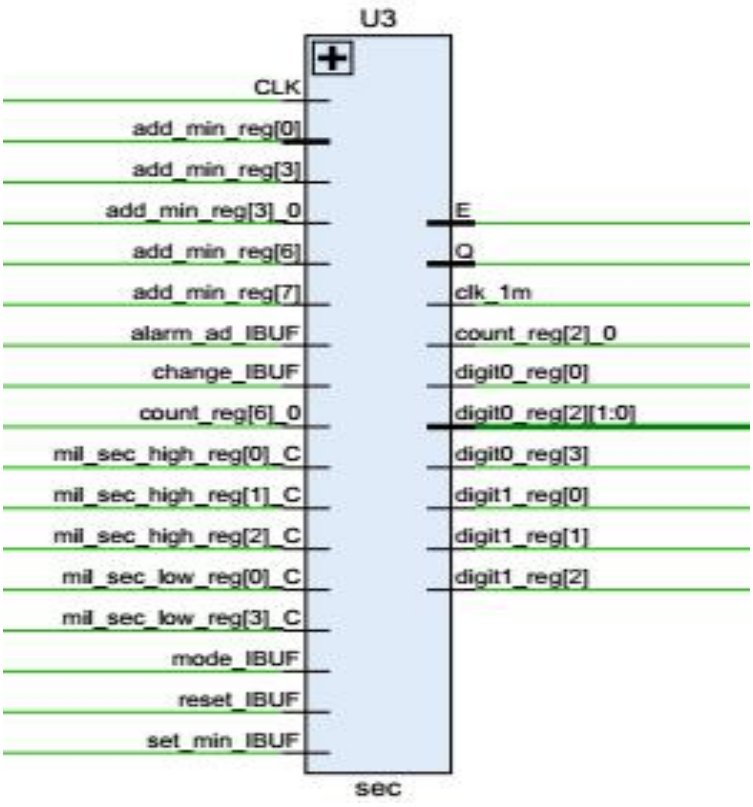
（1）秒计时模块

```
module sec(
    input wire clk,reset,
    output reg [3:0]sec_high,
    output reg [3:0]sec_low,
    output reg carry//进位
    );

    reg [7:0]count;//计数器

    always@(posedge clk, posedge reset)
    begin
        if(reset==1)//复位
        begin
            count<=0;
            sec_high<=4'd0;
            sec_low<=4'd0;
```

```
        end
        else
        begin
            carry <= 0;
            if(count==59)
            begin
                count<=0;
                sec_high<=4'd0;
                sec_low<=4'd0;
                carry<=1;
            end
            else
```

```
                carry<=0;
                count=count+1;
                sec_high <= count/10;
                sec_low <= count%10;//求余显示低位
            end
        end
    end
endmodule
```

该模块生成的 RTL 图如下：



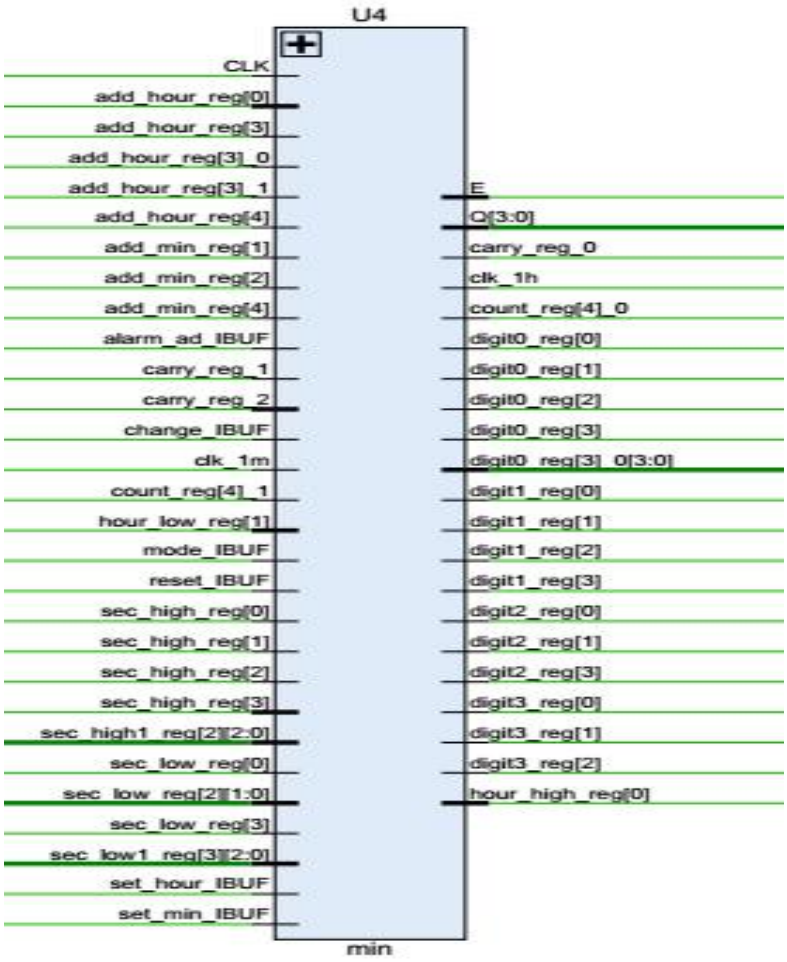
（2）分计时模块

当清零键为低电平时，如果校准分钟的按键为高电平并且未开启闹钟调节模式，则可以实现对电子钟分钟的时间校准；否则，为正常的分计时模块，原理与秒计时模块一致。

```
module min(
    input wire alarm_ad,
    input wire clk_1s,
    input wire set_min,
    input wire clk,reset,   //clk 输入秒计数模块的进位
    output reg [3:0]min_high,
    output reg [3:0]min_low,
    output reg carry
    );
    reg [7:0]count;
    always@(posedge clk_1s, posedge reset)
    begin
        if(reset==1)//复位
        begin
            count<=4'd0;
            min_high<=4'd0;
            min_low<=4'd0;
        end
        else if(set_min==1&&alarm_ad==0)
        begin
            if(count>=8'd59)
            begin
                min_low<=0;
                min_high<=0;
                count<=0;
                carry=1;
            end
            else
            begin
                carry=0;
                count=count+1;
```

```
                min_low<=count%10;
                min_high<=count/10;
            end
        end
        else if(clk==1)//clk 输入秒计数模块的进位
        begin
            if(count==8'd59)
            begin
                min_low<=0;
                min_high<=0;
                count<=0;
                carry=1;
            end
            else
            begin
                carry=0;
                count=count+1;
                min_low<=count%10;
                min_high<=count/10;
```

该模块生成的 RTL 图如下：



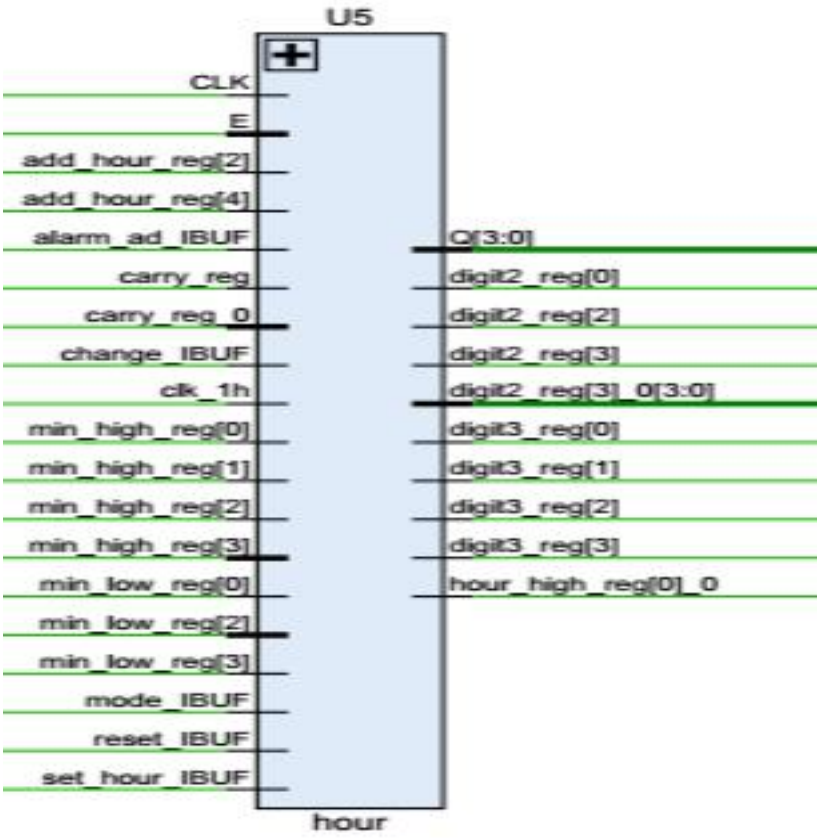
（3）时计时模块

时计时模块与分计时模块功能一致。

```
module hour(
    input wire alarm_ad,
    input wire clk_1s,
    input wire set_hour,
    input wire clk,reset,   //clk 输入分计数模块的进位
    output reg [3:0]hour_high,
    output reg [3:0]hour_low,
    output reg carry
    );
    reg [7:0]count;
    always@(posedge clk_1s, posedge reset)
    begin
        if(reset==1)
        begin
            count<=0;
            hour_high<=0;
            hour_low<=0;
        end
        else if(set_hour==1&&alarm_ad==0)
```

```
        begin
            if(count>=8'd23)
            begin
                count<=0;
                hour_low<=0;
                hour_high<=0;
                carry=1;
            end
            else
            begin
                carry=0;
                count=count+1;
                hour_low<=count%10;
                hour_high<=count/10;
            end
        end
        else if(clk==1)//clk 输入分计数模块的进位
        begin
            if(count==8'd23)
            begin
                count<=0;
                hour_low<=0;
                hour_high<=0;
                carry=1;
            end
            else
            begin
                carry=0;
                count=count+1;
                hour_low<=count%10;
                hour_high<=count/10;
            end
        end
    end

endmodule
```

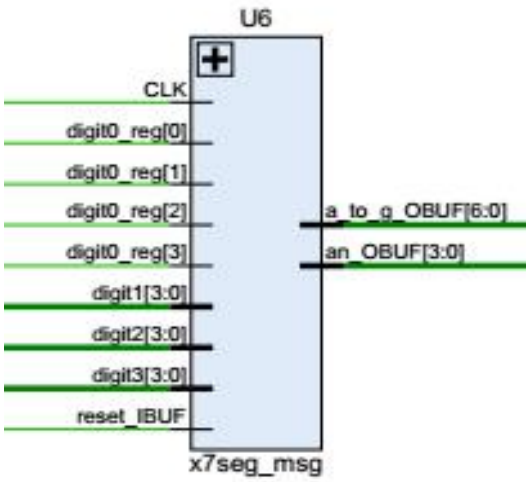
该模块生成的 RTL 图如下：



3.数码管显示模块

```
module   x7seg_msg(
    input clk,    //传入 clk190
    input clr,
    input [3:0]digit3,    //需要改成分钟和时钟
    input [3:0]digit2,
    input [3:0]digit1,
    input [3:0]digit0,
    output reg [6:0]a_to_g,   //段选段，七段管
    output reg [3:0]an   //位选端，四块显示屏分别显示的数字
    );
    reg [1:0]s;
    reg [3:0]digit;
    wire [3:0]aen;
    assign aen = 4'b1111;
    always@(posedge clk or posedge clr)
    begin
        if(clr==1)//清零
        begin
            s <= 0;
        end
        else //每次 190HZ 的频率闪过就选择亮哪一盏灯
            s <= s+1;
    end
    //确保每个数码管上都有显示数字
    always@(*)
    begin
        an = 4'b1111;
        if(aen[s]==1)
            an[s] = 0;    //显示第 s 位
        if(clr==1)
              an = 4'b1111;
    end
   //需要改变
    always@(*)
        case(s)
            0:digit = digit0;   //注意 n 取的最高位和最低位的顺序
            1:digit = digit1;
            2:digit = digit2;
            3:digit = digit3;
            default:digit = digit0[3:0];
        endcase
    always@(*)
        case(digit)
            0: a_to_g = 7'b0000001;
            1: a_to_g = 7'b1001111;
            2: a_to_g = 7'b0010010;
            3: a_to_g = 7'b0000110;
            4: a_to_g = 7'b1001100;
            5: a_to_g = 7'b0100100;
            6: a_to_g = 7'b0100000;
            7: a_to_g = 7'b0001111;
            8: a_to_g = 7'b0000000;
            9: a_to_g = 7'b0000100;
            default: a_to_g = 7'b1111111;
        endcase
endmodule
```

该模块生成的 RTL 图如下：



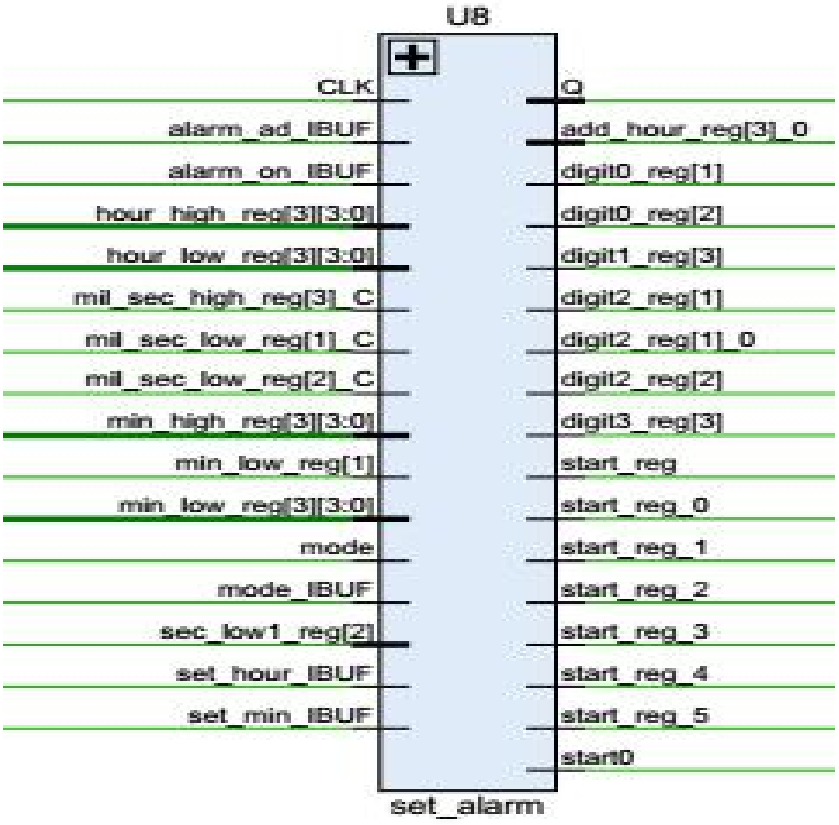
4.闹钟调时模块

当闹钟调时按键为高电平时，进入闹钟设置模式，此时通过按键分别对时钟和按键进行调整，设定闹钟。

```
module set_alarm(//闹钟调时设置模块
    input wire clk_1s,
    input wire alarm_ad,
    input wire set_alr_min, set_alr_hour,   //调整按键 和 set_min set_hour 共用
    output wire [3:0]add_hour_high,
    output wire [3:0]add_hour_low,
    output wire [3:0]add_min_high,
    output wire [3:0]add_min_low
    );

    reg [7:0] add_hour;
    reg [7:0] add_min;
        always@(posedge clk_1s)
    begin
        if(alarm_ad==1)
        begin
            if(set_alr_min==1)//调整分钟
            begin
                if(add_min>=8'd59)
                begin
                    add_min<=0;
                end
                else
                begin
                    add_min<=add_min+1;
                end
            end
            else if(set_alr_hour==1)//调整时钟
            begin
```

```
            begin
                start<=1;
            end
            else
            begin
               start<=0;
            end
        end

        always@(posedge clk, posedge reset)
        begin
            if(start==1&&reset==0)
            begin
                if(led_1==4'b0000)//四个 led 灯轮流闪烁
                    led_1<=4'b1000;
                else if(led_1 == 4'b1000)
                    led_1 <= 4'b0100;
                else if(led_1==4'b0100)
                     led_1<=4'b0010;
                else if(led_1==4'b0010)
                     led_1<=4'b0001;
                else if(led_1==4'b0001)
                     led_1<=4'b0000;
            end
            else if(start==0)
                led_1<=4'b0000;
        end
        assign led=led_1;
endmodule
```

该模块生成的 RTL 图如下：



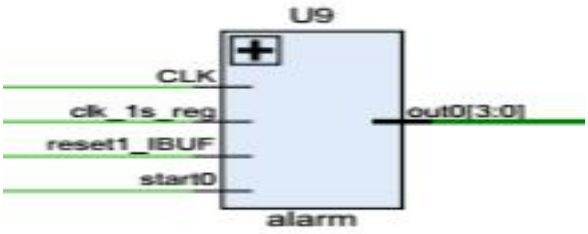
5.闹钟模块

当开启闹钟按键后，闹钟处于开启状态，此时时钟正常走，知道之前预定的时间，通过 LED 灯闪烁来实现闹铃，直到通过按键关闭闹钟。

```
module alarm(//闹钟模块
    input wire alarm_on,
    input wire [3:0]hour_high,
    input wire [3:0]hour_low,
    input wire [3:0]min_high,
    input wire [3:0]min_low,
    input wire [3:0]add_hour_high,
    input wire [3:0]add_hour_low,
    input wire [3:0]add_min_high,
    input wire [3:0]add_min_low,
    input wire clk,clk190,    //穿入 1s
    input wire reset,    //停止灯亮
    output [3:0] led//通过 led 灯来显示闹钟
    );
    reg [3:0]led_1;
    reg start;
    reg m;
    always@(posedge clk190)
    begin
        if((hour_high==add_hour_high)&&(hour_low==add_hour_low)&&
(min_high==add_min_high) && (min_low==add_min_low) && alarm_on==1)
```

```
            begin
                start<=1;
            end
            else
            begin
               start<=0;
            end
    end

    always@(posedge clk, posedge reset)
    begin
        if(start==1&&reset==0)
        begin
            if(led_1==4'b0000)//四个 led 灯轮流闪烁
                led_1<=4'b1000;
            else if(led_1 == 4'b1000)
                led_1 <= 4'b0100;
            else if(led_1==4'b0100)
                led_1<=4'b0010;
            else if(led_1==4'b0010)
                led_1<=4'b0001;
            else if(led_1==4'b0001)
                led_1<=4'b0000;
        end
        else if(start==0)
            led_1<=4'b0000;
    end
    assign led=led_1;
endmodule
```

该模块生成的 RTL 图如下：



6.秒表模块

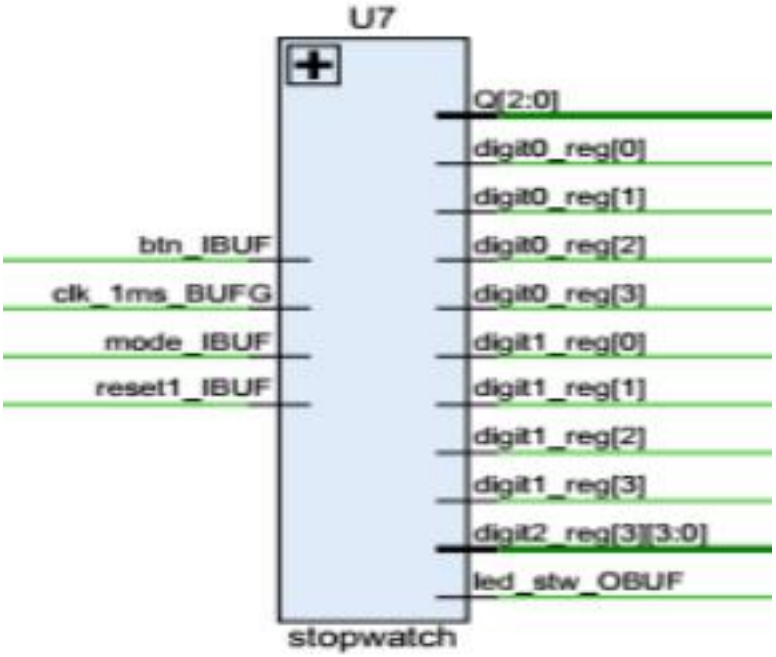
当按下 mode 键时，准备进入秒表模式，按下 btn 键后计时开始：同计时模块，秒表模块是对 1 毫秒的信号源进行计时，计时满后向上进位的设计思想；通过采用十进制数计时，对毫秒进行取余数、取除数，分别得到毫秒的个位，十位和百位，其中千位通过 LED 闪烁来表示，满 1000 毫秒则向高位进一位。

```
module stopwatch(//秒表
    input wire clk_1ms,reset1,mode,btn,
    output reg [3:0]mil_sec_high,
    output reg [3:0]mil_sec_low,
    output reg [3:0]sec_high1,
    output reg [3:0]sec_low1,
    output reg led);
```

```
    reg [12:0]mil_sec_cnt;
    reg [7:0]sec_cnt;
    reg carry;
    always@(posedge clk_1ms or posedge reset1)
    begin
        if(reset1==1)
        begin
            mil_sec_high=0;
            mil_sec_low=0;
            sec_high1=0;
            sec_low1=0;
            carry=0;
            mil_sec_cnt=0;
        end
        else if(mode==1)
        begin
            if(btn==1)//秒表的开关
            begin
                if(sec_cnt>=8'd59)
                begin
                    mil_sec_high=0;
                    mil_sec_low=0;
                    sec_high1=0;
                    sec_low1=0;
                    sec_cnt=0;
                    mil_sec_cnt=0;
                end
                else
                begin
                    if(mil_sec_cnt>=12'd999)
                    begin
                    sec_cnt<=sec_cnt+1;
                    led<=1;
                    mil_sec_high<=0;
                    mil_sec_low<=0;
                    sec_high1 = sec_cnt/10;
                    sec_low1= sec_cnt%10;
                    mil_sec_cnt<=0;
                    end
                    else
                    begin
                        led<=0;
                        mil_sec_cnt<=mil_sec_cnt+1;
```

```
                    mil_sec_high = mil_sec_cnt/100;
                    mil_sec_low = (mil_sec_cnt/10)%10;
                    sec_high1 = sec_cnt/10;
                    sec_low1 = sec_cnt%10;
                end
            end
        end
    end
    else if(mode==0)
    begin
        mil_sec_cnt=0;
        mil_sec_high=0;
        mil_sec_low=0;
    end
  end
endmodule
```

该模块生成的 RTL 图如下：



7.顶层模块

```
module stopwatch(//秒表
    input wire clk_1ms,reset1,mode,btn,
    output reg [3:0]mil_sec_high,
    output reg [3:0]mil_sec_low,
    output reg [3:0]sec_high1,
    output reg [3:0]sec_low1,
    output reg led
    );
        reg [12:0]mil_sec_cnt;
    reg [7:0]sec_cnt;
    reg carry;
    always@(posedge clk_1ms or posedge reset1)
    begin
        if(reset1==1)
        begin
          mil_sec_high=0;
          mil_sec_low=0;
          sec_high1=0;
          sec_low1=0;
          carry=0;
          mil_sec_cnt=0;
        end
        else if(mode==1)
        begin
                    mil_sec_high=0;
                    mil_sec_low=0;
                    sec_high1=0;
                    sec_low1=0;
                    sec_cnt=0;
                    mil_sec_cnt=0;
                end
                else
                begin
                    if(mil_sec_cnt>=12'd999)
                    begin
                    sec_cnt<=sec_cnt+1;
                    led<=1;
                    mil_sec_high<=0;
                    mil_sec_low<=0;
                    sec_high1 = sec_cnt/10;
                    sec_low1= sec_cnt%10;
                    mil_sec_cnt<=0;
                    end
                    else
                    begin
                        led<=0;
                        mil_sec_cnt<=mil_sec_cnt+1;
                        mil_sec_high = mil_sec_cnt/100;
                        mil_sec_low = (mil_sec_cnt/10)%10;
                        sec_high1 = sec_cnt/10;
                        sec_low1 = sec_cnt%10;
                    end
                end
          end
        end
        else if(mode==0)
        begin
          mil_sec_cnt=0;
          mil_sec_high=0;
          mil_sec_low=0;
        end
    end
endmodule
module clock_top(
    input wire alarm_on,
    input wire alarm_ad,     //用于是否显示时钟调节
    input wire change,
    input wire set_min,set_hour,
```

```
input wire clk,reset,
output wire [6:0]a_to_g,
output wire [3:0]an,
input wire reset1,    //闹钟停止
output wire [3:0]led,
output wire led_stw,
input wire mode,
input wire btn
);
wire clk_1s;
wire clk_1ms;
wire [3:0]sec_high;
wire [3:0]sec_low;
wire clk_1m;
wire [3:0]min_high;
wire [3:0]min_low;
wire clk_1h;
wire [3:0]hour_high;
wire [3:0]hour_low;
wire clk_1d;
wire clk190;
wire [3:0] add_hour_high;
wire [3:0] add_hour_low;
wire [3:0] add_min_high;
wire [3:0] add_min_low;
clk_1s U1(
      .clk(clk),
      .reset(reset),      //clk 接 w5
      .clk_1s(clk_1s)
      );

clkdiv U2(
      .clk(clk),
      .clr(reset),
      .clk190(clk190)
            );

 sec U3(
      .clk(clk_1s),
      .reset(reset),
      .sec_high(sec_high),
      .sec_low(sec_low),
      .carry(clk_1m)
      );
```

```
  min U4(
     .alarm_ad(alarm_ad),
     .clk_1s(clk_1s),
     .set_min(set_min),
     .clk(clk_1m),
     .reset(reset),    //clk 输入秒计数模块的进位
     .min_high(min_high),
     .min_low(min_low),
     .carry(clk_1h)
    );

    hour U5(
          .alarm_ad(alarm_ad),
          .clk_1s(clk_1s),
         .set_hour(set_hour),
         .clk(clk_1h),
         .reset(reset),    //clk 输入分计数模块的进位
         .hour_high(hour_high),
         .hour_low(hour_low),
         .carry(clk_1d)
         );


     set_alarm U8(
          .clk_1s(clk_1s),
          .alarm_ad(alarm_ad),
          .set_alr_min(set_min),
          .set_alr_hour(set_hour),    //调整按键
          .add_hour_high(add_hour_high),
          .add_hour_low(add_hour_low),
          .add_min_high(add_min_high),
          .add_min_low(add_min_low)
          );

          alarm U9(
          .alarm_on(alarm_on),
          .hour_high(hour_high),
          .hour_low(hour_low),
          .min_high(min_high),
          .min_low(min_low),
          .add_hour_high(add_hour_high),
          .add_hour_low(add_hour_low),
          .add_min_high(add_min_high),
          .add_min_low(add_min_low),
```

```
        .clk(clk_1s),
        .clk190(clk190),     //穿入 1s
        .reset(reset1),     //停止灯亮
        .led(led)
        );

    clk_1ms U10(
        .clk(clk),
        .reset(reset),     //clk 接 w5
        .clk_1ms(clk_1ms)
        );

        wire [3:0] mil_sec_high;
        wire [3:0] mil_sec_low;
        wire [3:0] sec_high1;
        wire [3:0] sec_low1;

    stopwatch U7(
        .clk_1ms(clk_1ms),
        .reset1(reset1),
        .mode(mode),
        .btn(btn),
        .mil_sec_high(mil_sec_high),
        .mil_sec_low(mil_sec_low),
        .sec_high1(sec_high1),
        .sec_low1(sec_low1),
        .led(led_stw)
        );
    reg [3:0]digit3;
    reg [3:0]digit2;
    reg [3:0]digit1;
    reg [3:0]digit0;

//时分到分秒的切换
    always@(posedge clk)
    begin
        if(change==1&&alarm_ad==0&&mode==0)
        begin
            digit3<=hour_high;
            digit2<=hour_low;
            digit1<=min_high;
            digit0<=min_low;
        end
        else if(change==0 && alarm_ad==0&&mode==0)
        begin
            digit3<=min_high;
            digit2<=min_low;
            digit1<=sec_high;
            digit0<=sec_low;
        end
        else if(alarm_ad==1&&mode==0)
        begin
            digit3<=add_hour_high;     //显示闹钟调整
            digit2<=add_hour_low;
            digit1<=add_min_high;
            digit0<=add_min_low;
        end
        else if(mode==1)
        begin
            digit3<=sec_high1;
            digit2<=sec_low1;
            digit1<=mil_sec_high;
            digit0<=mil_sec_low;
        end
    end

    x7seg_msg U6(
        .clk(clk190),     //传入 clk190
        .clr(reset),
        .digit3(digit3),     //需要改成分钟和时钟
        .digit2(digit2),
        .digit1(digit1),
        .digit0(digit0),
        .a_to_g(a_to_g),    //段选段
        .an(an)    //位选端
            );

endmodule
```

注：顶层模块中通过按键 change 实现了时分到分秒的切换；

三、问题分析

（1）要注意编写程序的过程中 begin 和 end 配对问题，类似于 C 语言中的括号匹配问题，在编写计数模块式编译不通过，最后检查是缺少一个 end 结束符号，经修改后通过编译。
（2）Verilog HDL 语言编写时的语法问题。在最初的计时模块的程序设计中，将小时、分钟的调节信号放在了另外的一个 always 语句块中，编译无法通过，经查阅资料，在 Verilog HDL 语言的编写中应该注意不同的 always 语句块不可以对同一个变量进行操作，即一个变量不可以经过两个 always 语句块操作。将对小时和分钟调节信号的操作与计时放在同一个语句块中，编译通过。

四、心得体会

在这次的设计中我学会了很多东西。首先是对 Verilog HDL 语言的设计思想有了深入理解，将这种自顶向下的设计理念运用于实践中，设计多功能数字钟，突出了 Verilog HDL 作为硬件描述语言的良好可读性和可移植性，对理论知识有了深刻的理解。

其次是对 Verilog HDL 语言的语法熟悉，在这次的课程设计中，我学习到很多关于 Verilog HDL 语言的语法知识，比如在两个不同的语句块中不能对同一个变量进行操作，比如在用 Verilog HDL 语言中编写程序时要注意 begin 和 end 语句的匹配问题，在使用 Verilog HDL 语言时不可以使用中文注释等等。对于这种语言的学习也有了很大的帮助。

最后是设计作品时的设计逻辑和设计思想，在选择不同的系统方案时要综合考虑，选择最优方案。各个模块的实现也要考虑综合情况而制定出最符合实际情况的实现方案，方案间要进行对比、实践，最终确定。

在这次的课程设计中我不仅学习到有关程序编写以及设计方面的逻辑思维，对系统功能的实现也有了较为深入的了解，对各模块的调试等也学习到不少东西，总之，从这次设计中学到很多东西，也巩固了我的理论学习。

五、参考文献

1、赵倩、叶波、林丽萍、周多、王晓华编著，Verilog 数字系统设计与 FPGA 应用，清华大学出版社；
2、王金明编著，徐志军主审，Verilog HDL 程序设计教程，人民邮电出版社。