EDA 大作业

学生姓名：章星宇

学　　号：19200300029

班　　级：1920031

授课教师：白　勃

日　　期：2021 年 10 月 31 日

目录

一、实验目的

设计一个电子闹钟。要求电路上电后自动计时，到达预置的闹响时刻后，由蜂鸣器发出音乐报警。闹响时刻可利用按键设置，设置范围 0~999999。

此次实验除了满足上述基本功能外，额外添置了流水灯功能，当到达预置的闹响时刻后，不仅蜂鸣器会发出音乐报警，并且 LED 会形成流水灯。

二、实验环境

2.1 硬件环境

本实验采用的开发板是正点原子的开拓者 FPGA 开发板。

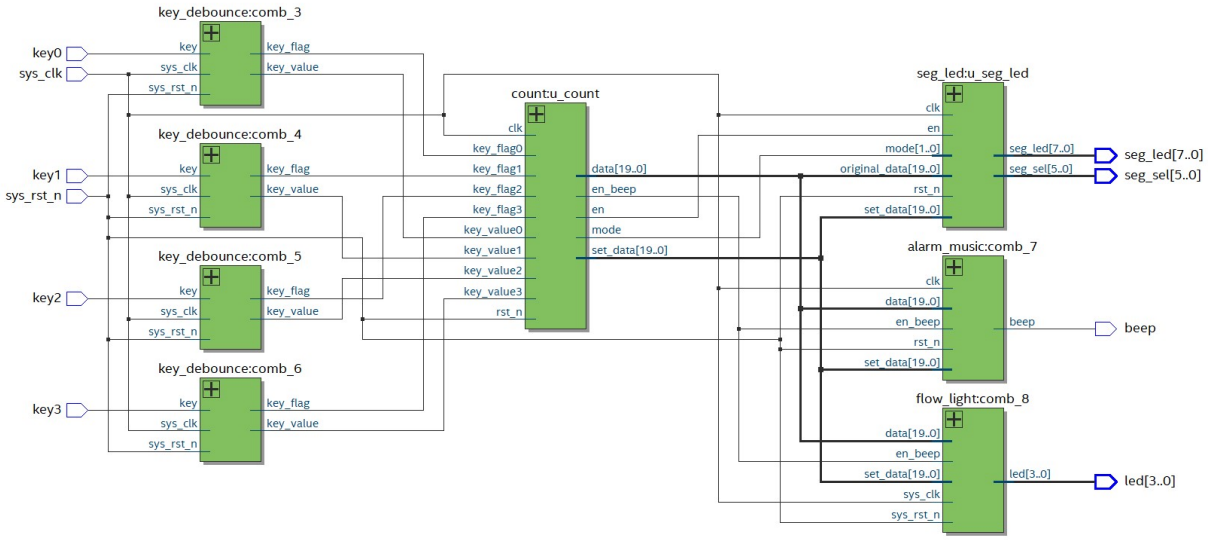
2.2 软件环境

使用软件：Quartus Ⅱ（18.1）、ModelSim（10.5）

操作系统：Windows 10（64 位）

三、方案设计及理论计算

3.1 原理框图



输入：时钟信号，重置信号，按键信号。

输出：数码管位选信号、数码管段选信号、流水灯控制信号、蜂鸣器控制信号。

3.2 分频器

输入：时钟信号，重置信号。

输出：分频时钟信号。

功能：对 50MHz 分频。

板子上电之后需每隔 1s 进行计数，板子的时钟频率为 50MHz，为满足这一功能，需要设计一个分频器，对板子的 50MHz 频率进行分频，从而输出一个 1Hz 的时钟信号。

分频器模值、系统时钟和期望输出时钟频率关系为

$$\text{分频器模值}=\frac{\text{系统时钟频率}}{\text{期望输出时钟频率}}\text{-1。}$$

所以，把 50MHz 时钟分频，输出 1Hz 的时钟，分频器的模值为

$$M=\frac{50000000}{1}\text{-1=49999999}$$

为了保证分频器正常工作，计数器寄存器所能表示的最大值必须大于分频器的模值。这里，设置把计数器寄存器的位数设定为 26 位。

流程图如图 1 所示。

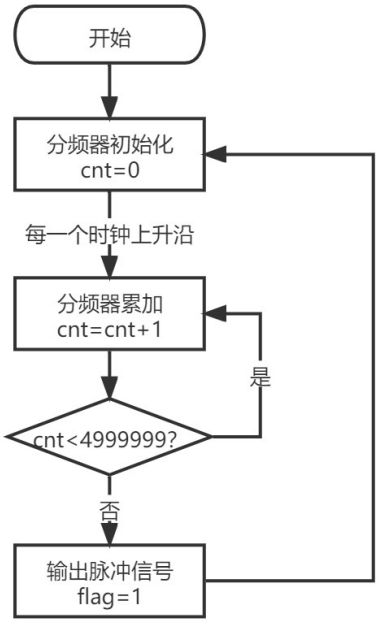


图 1 分频器

数码管的动态显示和流水灯也同样用到分频器，原理一样。其中，数码管对系统时钟频率进行了 10 分频，流水灯对系统时钟频率进行了 5 分频。

3.3 按键消抖

输入：时钟信号，重置信号，按键信号。

输出：按键数据有效信号，按键消抖后数据。

功能：消除按键抖动。

由于每次按下按键时存在抖动，容易引起按键的多次触发，因此按键消抖模块就可以解决这个问题。

按键消抖主要通过延时来实现，即当按键的一个状态保持 20ms 以上，即锁存按键的状态，这样既保证按键消抖的稳定又保证了一定的灵敏性。原理图如图 2 所示。

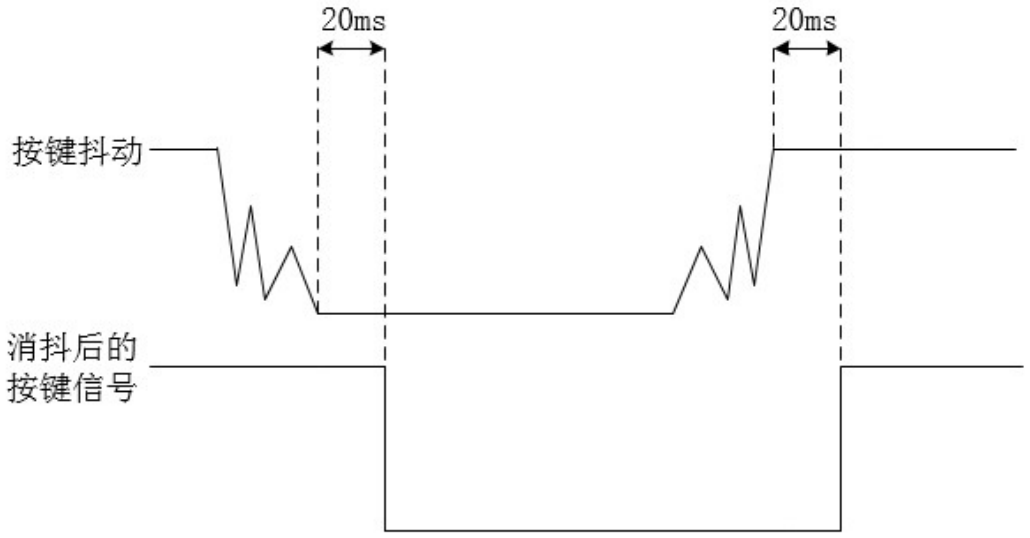


图 2 按键消抖原理图

3.4 状态机

实验中需要通过按键实现时间显示和时间设定两个模式的切换，本实验采用了一位寄存器作为状态机，当按键按下时，状态机可以从 0、1 两个状态之间来回切换。原理如图 3 所示：

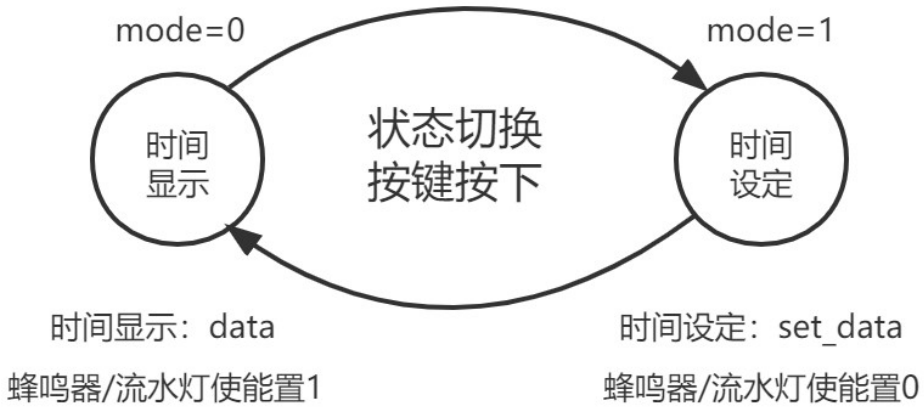


图 3 模式切换图

为了防止时间设定模式时对蜂鸣器和流水灯的干扰，本实验单独设置了一个寄存器信号，用于隔离两种模式。同时，显示时间和设定时间采用不同的寄存器存储，这样方便比较并且当时间设定时，时间显示会处于暂停的状态。

3.5 数码管动态显示

输入：重置信号，时钟信号，计数时间，设定时间，数码管使能信号、模式信号。

输出：数码管位选信号，数码管段选信号。

功能：将计数时间/设定时间在数码管上进行显示。

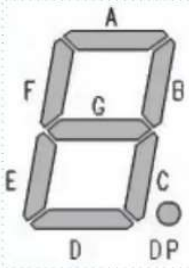
在数码管动态显示模块中，首先需要根据模式信号来判断需要显示的时间，即时间显示模式显示计数时间，时间设定模式显示设定时间。之后，通过整除去尾、取模取尾的计算方式，将时间每个位数上的数据提取出来。例如，提取十位数的数据公式为：

$$十位数 = Data/10\%10$$

将每一位上的数据提取出来后，将其转换成 8421BCD 码，与每一个数码管进行绑定。通过对系统时钟进行 10 分频，得到的频率为 5MHz 的数码管驱动时钟，用来控制数码管的位选信号，使每一个数码管以 1ms 的时间周期轮流显示。

当数码管需要显示时，通过段选信号将每一位数码管绑定的数据进行转换，从而显示出正确的数值。本实验的开发板采用的是共阳极数码管，段选真值表如图 4 所示。

共阳极八段数码管真值表：



字形	Dp	g	f	e	d	c	b	a
.	0	1	1	1	1	1	1	1
0	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	1
2	1	0	1	0	0	1	0	0
3	1	0	1	1	0	0	0	0
4	1	0	0	1	1	0	0	1
5	1	0	0	1	0	0	1	0
6	1	0	0	0	0	0	1	0
7	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0
A	1	0	0	0	1	0	0	0
B	1	0	0	0	0	0	1	1
C	1	1	0	0	0	1	1	0
D	1	0	1	0	0	0	0	1
E	1	0	0	0	0	1	1	0
F	1	0	0	0	1	1	1	0

图 4 共阳极八段数码管真值表

3.6 引脚分配

根据开发手册配置相关引脚如图 5 所示。

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
beep	Output	PIN_D12	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
key0	Input	PIN_E16	6	B6_N0	2.5 V (default)		8mA (default)			
key1	Input	PIN_E15	6	B6_N0	2.5 V (default)		8mA (default)			
key2	Input	PIN_M2	2	B2_N0	2.5 V (default)		8mA (default)			
key3	Input	PIN_M16	5	B5_N0	2.5 V (default)		8mA (default)			
led[3]	Output	PIN_F9	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
led[2]	Output	PIN_E10	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
led[1]	Output	PIN_C11	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
led[0]	Output	PIN_D11	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[7]	Output	PIN_D9	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[6]	Output	PIN_P11	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[5]	Output	PIN_N11	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[4]	Output	PIN_M10	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[3]	Output	PIN_N13	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[2]	Output	PIN_C9	7	B7_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[1]	Output	PIN_N12	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_led[0]	Output	PIN_M11	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[5]	Output	PIN_T15	4	B4_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[4]	Output	PIN_R16	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[3]	Output	PIN_P15	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[2]	Output	PIN_P16	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[1]	Output	PIN_N15	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
seg_sel[0]	Output	PIN_N16	5	B5_N0	2.5 V (default)		8mA (default)	2 (default)		
sys_clk	Input	PIN_E1	1	B1_N0	2.5 V (default)		8mA (default)			
sys_rst_n	Input	PIN_M1	2	B2_N0	2.5 V (default)		8mA (default)			

图 5 引脚分配图

四、波形仿真

4.1 仿真参数

sys_clk：时钟信号

sys_rst_n：复位信号

key0：按键信号，用于切换模式

key1：按键信号，在设定模式下，按一次设定时间加一秒

key2：按键信号，在设定模式下，按一次设定时间减一秒

key3：按键信号，在设定模式下，按一次设定时间加十秒

beep：蜂鸣器信号

seg_sel：数码管位选信号

seg_led：数码管段选信号

led：流水灯信号

4.2 仿真波形图

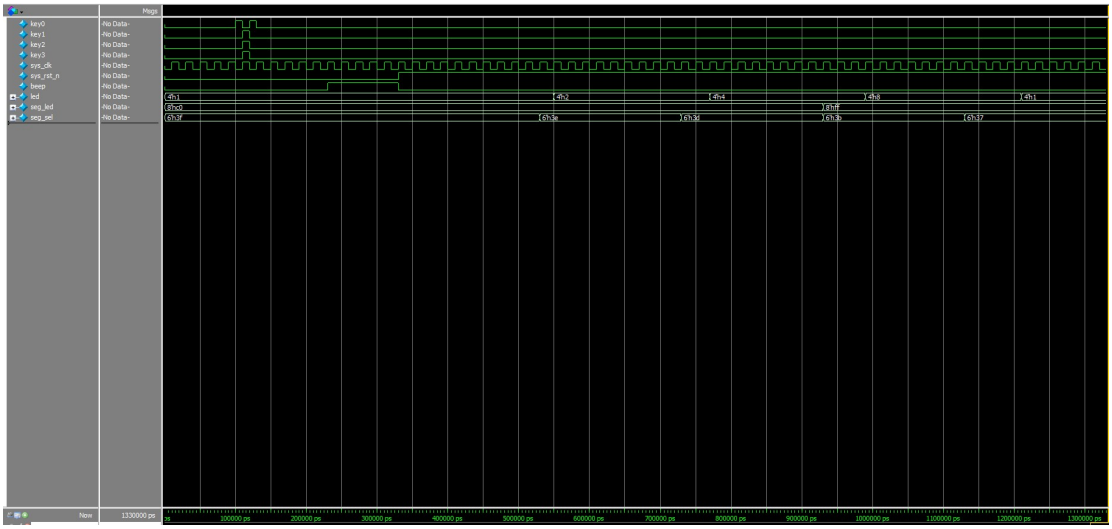


图 6 波形仿真图

如图 6 可以看到上电之后，程序开始正常计时，首先按下模式切换按键 key0，切换到设定时间模式，之后按下 key1，设定时间加一，再按下 key2，设定时间减一，再按下 key3，设定时间加十，共设计十秒时间。最后再按一次 key0，重新切换到计时模式。当计时时间到达设定时间时，蜂鸣器的 beep 信号变为高电平，流水灯开始工作。按下复位信号后，蜂鸣器的 beep 信号变为低电平，停止工作，流水灯也回复到初始状态。

五、实验结果

将程序通过驱动下载到开发板上后，开发板开始自动计数，按下 key0 进入设定模式，通过其它三个按键设定好闹钟时间。再按 key0，返回计数模式。当时间到达设定时间时，蜂鸣器播放音乐，流水灯开始工作，达到预期效果。

实验过程图如图 7 所示。

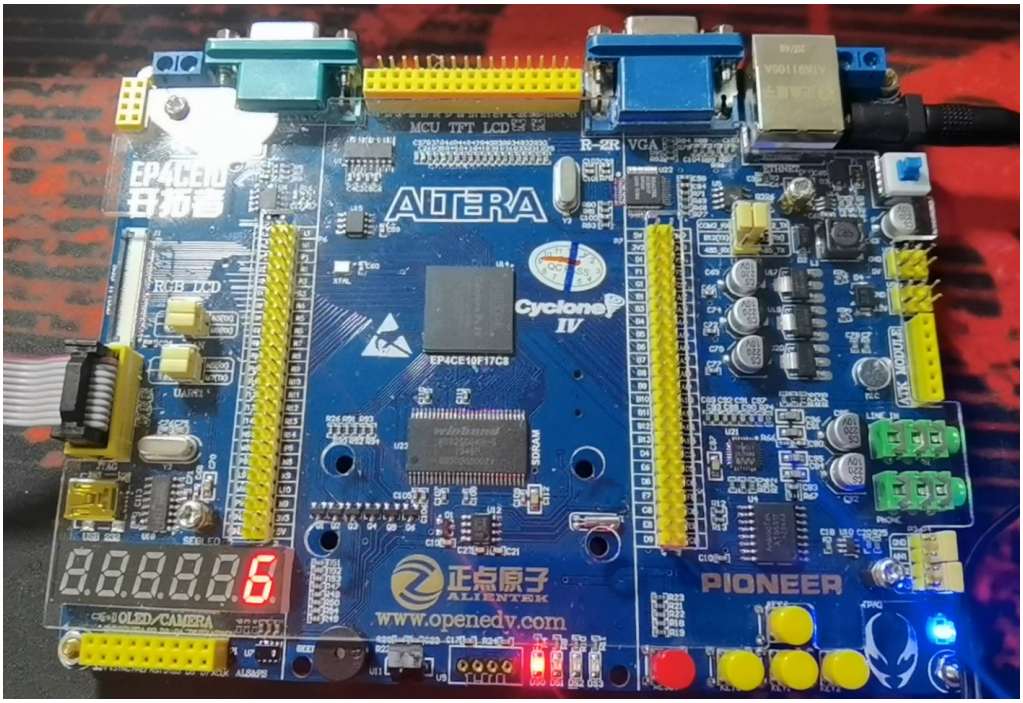


图 7 实验过程图

六、附录

顶层模块

```
module top_alarm(
    //global clock
    input            sys_clk   ,       // 全局时钟信号
    input            sys_rst_n,       // 复位信号（低有效）
    input           key0,           //按键信号
    input           key1,
    input           key2,
    input           key3,
    //seg_led interface
        output    [5:0]  seg_sel  ,       // 数码管位选信号
        output    [7:0]  seg_led  ,        // 数码管段选信号
        output   [3:0]  led ,       //4 个 LED 灯
        output beep
    );

    //wire define
    wire    [19:0]  data;                   // 数码管正常计数显示的数值
    wire    [19:0]  set_data;          // 数码管设定闹钟显示的数值
    wire   en;                                  // 数码管显示使能信号
    wire    en_beep;                                    // 蜂鸣器使能信号
    wire   key_value0;
```

```
22.  wire   key_value1;
23.  wire   key_value2;
24.  wire   key_value3;
25.  wire   key_flag0;
26.  wire   key_flag1;
27.  wire   key_flag2;
28.  wire   key_flag3;
29.
30.  // 对四个按键分别进行消抖
31.  key_debounce k0 (
32.      .sys_clk(sys_clk),           //外部 50M 时钟
33.      .sys_rst_n(sys_rst_n),         //外部复位信号，低有效
34.      .key(key0),                //外部按键输入
35.      .key_flag(key_flag0),          //按键数据有效信号
36.      .key_value(key_value0)        //按键消抖后的数据
37.      );
38.
39.  key_debounce k1(
40.      .sys_clk(sys_clk),           //外部 50M 时钟
41.      .sys_rst_n(sys_rst_n),         //外部复位信号，低有效
42.      .key(key1),                //外部按键输入
43.      .key_flag(key_flag1),          //按键数据有效信号
44.      .key_value(key_value1)        //按键消抖后的数据
45.  );
46.
47.   key_debounce k2(
48.      .sys_clk(sys_clk),           //外部 50M 时钟
49.      .sys_rst_n(sys_rst_n),         //外部复位信号，低有效
50.      .key(key2),                //外部按键输入
51.      .key_flag(key_flag2),          //按键数据有效信号
52.      .key_value(key_value2)        //按键消抖后的数据
53.   );
54.
55.   key_debounce k3(
56.      .sys_clk(sys_clk),           //外部 50M 时钟
57.      .sys_rst_n(sys_rst_n),         //外部复位信号，低有效
58.      .key(key3),                //外部按键输入
59.      .key_flag(key_flag3),          //按键数据有效信号
60.      .key_value(key_value3)        //按键消抖后的数据
61.   );
62.
63.  //计数器模块，产生数码管需要显示的数据
64.  count u_count(
65.      .clk           (sys_clk  ),        // 时钟信号
66.      .rst_n         (sys_rst_n),        // 复位信号
67.      .key_flag0   (key_flag0) ,     //按键有效信号
68.      .key_value0 (key_value0),    //消抖后的按键信号
69.      .key_flag1   (key_flag1) ,     //按键有效信号
70.      .key_value1 (key_value1),    //消抖后的按键信号
71.      .key_flag2   (key_flag2) ,     //按键有效信号
72.      .key_value2 (key_value2),    //消抖后的按键信号
```

```
        .key_flag3   (key_flag3) ,     //按键有效信号
        .key_value3 (key_value3),    //消抖后的按键信号
        .set_data(set_data) ,       // 设定 0～999999 时间
        .data          (data),        // 6 位数码管要显示的数值
        .en            (en),        // 数码管使能信号
        .en_beep (en_beep ),
        .mode   (mode)
    );


    //数码管动态显示模块
    seg_led u_seg_led(
        .clk           (sys_clk  ),        // 时钟信号
        .rst_n         (sys_rst_n),        // 复位信号

        .original_data          (data),        // 显示的数值
        .set_data(set_data),
        .en            (en       ),        // 数码管使能信号
        .mode   (mode),

        .seg_sel       (seg_sel  ),        // 位选
        .seg_led       (seg_led  )         // 段选
    );

    // 蜂鸣器音乐模块
    alarm_music alarm(
        .clk           (sys_clk  ),            // 时钟信号
        .rst_n         (sys_rst_n),        // 复位信号
        .data          (data) ,                // 实际的时间
        .set_data (set_data ),      // 设定的时间
        .en_beep (en_beep ),
        .beep(beep)
    );

    flow_light light(
        .sys_clk  (sys_clk) ,  //系统时钟
        .sys_rst_n (sys_rst_n) ,  //系统复位，低电平有效
        .data          (data) ,                // 实际的时间
        .set_data (set_data ),      // 设定的时间
        .en_beep (en_beep ),
        .led (led)          //4 个 LED 灯
        );
    endmodule
```

按键消抖模块

```
1. module key_debounce(
2.     input            sys_clk,          //外部 50M 时钟
3.     input            sys_rst_n,        //外部复位信号，低有效
4.
```

```
    input            key,              //外部按键输入
    output reg       key_flag,         //按键数据有效信号
    output reg       key_value         //按键消抖后的数据
    );

    //reg define
    reg [31:0] delay_cnt;
    reg        key_reg;


    always @(posedge sys_clk or negedge sys_rst_n) begin
        if (!sys_rst_n) begin
            key_reg   <= 1'b1;
            delay_cnt <= 32'd0;
        end
        else begin
            key_reg <= key;
            if(key_reg != key)             //一旦检测到按键状态发生变化(有按键被按下或释放)
                delay_cnt <= 32'd1000000;  //给延时计数器重新装载初始值(计数时间为20ms)
            else if(key_reg == key) begin  //在按键状态稳定时,计数器递减,开始 20ms 倒计时
                    if(delay_cnt > 32'd0)
                        delay_cnt <= delay_cnt - 1'b1;
                    else
                        delay_cnt <= delay_cnt;
                end
        end
    end

    always @(posedge sys_clk or negedge sys_rst_n) begin
        if (!sys_rst_n) begin
            key_flag  <= 1'b0;
            key_value <= 1'b1;
        end
        else begin
            if(delay_cnt == 32'd1) begin   //当计数器递减到 1 时，说明按键稳定状态维持了 20ms
                key_flag  <= 1'b1;         //此时消抖过程结束,给出一个时钟周期的标志信号
                key_value <= key;          //并寄存此时按键的值
            end
            else begin
                key_flag  <= 1'b0;
                key_value <= key_value;
            end
        end
    end

    endmodule
```

计数模块

```
module count(
    //mudule clock
    input                   clk   ,      // 时钟信号
    input                   rst_n,      // 复位信号
    input        key_flag0,    //按键有效信号
    input        key_value0,    //消抖后的按键信号
    input        key_flag1,    //按键有效信号
    input        key_value1,    //消抖后的按键信号
    input        key_flag2,    //按键有效信号
        input        key_value2,    //消抖后的按键信号
        input        key_flag3,    //按键有效信号
        input        key_value3,    //消抖后的按键信号

        //user interface
        output   reg [19:0]     data ,      // 6 个数码管要显示的数值
        output   reg [19:0]     set_data ,      // 设定 0～999999 时间
        output   reg     en   ,     // 数码管使能信号
        output   reg     en_beep   ,     // 蜂鸣器使能信号
        output   reg     mode                //为了控制不同状态的显示，同时需输出当前模式
        );

    //parameter define
    parameter  MAX_NUM = 26'd5000_0000;      // 计数器计数的最大值 1s/20ns

    //reg define
    reg    [25:0]   cnt ;                   // 计数器，用于计时 1s
    reg             flag;                   // 标志信号
    reg             flag_key;                   // 标志信号

    //模式选择
    always @ (posedge clk or negedge rst_n) begin
        if(!rst_n)
            mode <= 1'd0;
        else if(key_flag0&& (~key_value0)) begin //判断按键是否有效按下
            mode <= mode + 1'b1;   // 一位状态机，0、1 循环
            end
        else begin
                mode <= mode;
         end
     end

    //计数器对系统时钟计数达 1s 时，输出一个时钟周期的脉冲信号
    always @ (posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            cnt <= 26'b0;
            flag<= 1'b0;
        end
```

```
        else if (cnt < MAX_NUM - 1'b1) begin
            cnt <= cnt + 1'b1;
            flag<= 1'b0;
        end
        else begin
            cnt <= 26'b0;
            flag <= 1'b1;
        end
    end


    //数码管需要显示的数据，从 0 累加到 999999
    always @ (posedge clk or negedge rst_n) begin
        if (!rst_n)begin
            data  <= 20'b0;
            set_data <=20'b0;
            en    <= 1'b0;
        end
        // 按键 0 切换状态，mode=0 时正常计数
        else if ( mode == 1'b0) begin
            en    <= 1'b1;                  //打开数码管使能信号
            en_beep    <= 1'b1;
            if (flag) begin                 //显示数值每隔 1s 累加一次
                if(data < 20'd999999)
                    data <= data +1'b1;
                else
                    data <= 20'b0;
            end
        else
               data <= data;
        end
        // 按键 0 切换状态，mode=1 时进入设定模式
        else if (mode == 1'b1) begin
            en    <= 1'b1;
            en_beep    <= 1'b0;
            //按下按键 1 设定数+1
            if (key_flag1&& (~key_value1))  begin
                    if(set_data < 20'd999999)
                            set_data <= set_data + 1'b1;
                     else
                            set_data <= 20'b0;
                     end
              //按下按键 3 设定数+10
            else if (key_flag3&& (~key_value3))  begin
                    if(set_data < 20'd999999)
                            set_data <= set_data + 4'd10;
                     else
                            set_data <= 20'b0;
                     end
            //按下按键 2 设定数-1
            else if (key_flag2&& (~key_value2))  begin
```

```
                    if(set_data > 20'd0)
                            set_data <= set_data - 1'b1;
                     else
                            set_data <= 20'b0;
                    end
             //不按键的时设定数维持不变
            else
                    set_data <=  set_data;
            end
end

endmodule
```

数码管显示模块

```
module seg_led(
    input                   clk    ,        // 时钟信号
    input                   rst_n  ,        // 复位信号

    input         [19:0]    original_data   ,        // 6 位数码管要显示的计数时间数值
    input         [19:0]    set_data  ,        // 6 位数码管要显示的设定时间数值
    input                   en     ,        // 数码管使能信号
    input         [1:0] mode ,

        output   reg  [5:0]     seg_sel,        // 数码管位选，最左侧数码管为最高位
        output   reg  [7:0]     seg_led         // 数码管段选
        );

    //parameter define
    localparam  CLK_DIVIDE = 4'd10     ;        // 时钟分频系数
    localparam  MAX_NUM    = 13'd5000  ;        // 对数码管驱动时钟(5MHz)计数 1ms 所需的计数值

    //reg define
    reg    [ 3:0]             clk_cnt  ;        // 时钟分频计数器
    reg                       dri_clk  ;        // 数码管的驱动时钟,5MHz
    reg    [23:0]             num      ;        // 24 位 bcd 码寄存器
    reg    [12:0]             cnt0     ;        // 数码管驱动时钟计数器
    reg                       flag     ;        // 标志信号（标志着 cnt0 计数达 1ms）
    reg    [2:0]              cnt_sel  ;        // 数码管位选计数器
    reg    [3:0]              num_disp ;        // 当前数码管显示的数据


    //wire define
    wire   [19:0]              data     ;        // 需要生成的 data
    wire   [3:0]               data0    ;        // 个位数
    wire   [3:0]               data1    ;        // 十位数
```

```
33.  wire   [3:0]              data2    ;        // 百位数
34.  wire   [3:0]              data3    ;        // 千位数
35.  wire   [3:0]              data4    ;        // 万位数
36.  wire   [3:0]              data5    ;        // 十万位数
37.
38.
39.  //确定选择的数据
40.
41.  assign data = (mode== 1'b0)? original_data : set_data;
42.
43.  //提取显示数值所对应的十进制数的各个位
44.  assign  data0 = data % 4'd10;                 // 个位数
45.  assign  data1 = data / 4'd10 % 4'd10   ;     // 十位数
46.  assign  data2 = data / 7'd100 % 4'd10  ;    // 百位数
47.  assign  data3 = data / 10'd1000 % 4'd10 ;   // 千位数
48.  assign  data4 = data / 14'd10000 % 4'd10;   // 万位数
49.  assign  data5 = data / 17'd100000;          // 十万位数
50.
51.  //对系统时钟 10 分频，得到的频率为 5MHz 的数码管驱动时钟 dri_clk
52.  always @(posedge clk or negedge rst_n) begin
53.     if(!rst_n) begin
54.         clk_cnt <= 4'd0;
55.         dri_clk <= 1'b1;
56.     end
57.     else if(clk_cnt == CLK_DIVIDE/2 - 1'd1) begin
58.         clk_cnt <= 4'd0;
59.         dri_clk <= ~dri_clk;
60.     end
61.     else begin
62.         clk_cnt <= clk_cnt + 1'b1;
63.         dri_clk <= dri_clk;
64.     end
65.  end
66.
67.  //将 20 位 2 进制数转换为 8421bcd 码(即使用 4 位二进制数表示 1 位十进制数）
68.  always @ (posedge dri_clk or negedge rst_n) begin
69.      if (!rst_n)
70.          num <= 24'b0;
71.      else begin
72.          if (data5) begin     //如果显示数据为 6 位十进制数，
73.              num[23:20] <= data5;         //则依次给 6 位数码管赋值
74.              num[19:16] <= data4;
75.              num[15:12] <= data3;
76.              num[11:8]  <= data2;
77.              num[ 7:4]  <= data1;
78.              num[ 3:0]  <= data0;
79.          end
80.          else begin
81.              if (data4) begin //如果显示数据为 5 位十进制数，则给低 5 位数码管赋
值
82.                  num[19:0] <= {data4,data3,data2,data1,data0};
```

```
                num[23:20] <= 4'd10; //不需要显示负号时，则第 6 位不显示任何字符
            end
            else begin                  //如果显示数据为 4 位十进制数，则给低 4 位数码管赋值
                if (data3) begin
                    num[15: 0] <= {data3,data2,data1,data0};
                    num[23:20] <= 4'd10; //第 6 位不显示任何字符
                    num[19:16] <= 4'd10;
                end
                else begin              //如果显示数据为 3 位十进制数，则给低 3 位数码管赋值
                    if (data2) begin
                        num[11: 0] <= {data2,data1,data0};
                                        //第 6、5 位不显示任何字符
                        num[23:16] <= {2{4'd10}};
                        num[15:12] <= 4'd10;
                    end
                    else begin          //如果显示数据为 2 位十进制数，则给低 2 位数码管赋值
                        if (data1) begin
                            num[ 7: 0] <= {data1,data0};
                                        //第 6、5、4 位不显示任何字符
                            num[23:12] <= {3{4'd10}};
                            num[11:8] <=  4'd10;
                        end
                        else begin      //如果显示数据为 1 位十进制数，则给最低位数码管赋值
                            num[3:0] <= data0;
                                        //第 6、5 位不显示任何字符
                            num[23:8] <= {4{4'd10}};
                            num[7:4] <= 4'd10;
                        end
                    end
                end
            end
        end
    end
end

//每当计数器对数码管驱动时钟计数时间达 1ms，输出一个时钟周期的脉冲信号
always @ (posedge dri_clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        cnt0 <= 13'b0;
        flag <= 1'b0;
     end
    else if (cnt0 < MAX_NUM - 1'b1) begin
        cnt0 <= cnt0 + 1'b1;
        flag <= 1'b0;
     end
    else begin
        cnt0 <= 13'b0;
```

```
            flag <= 1'b1;
        end
end

//cnt_sel 从 0 计数到 5，用于选择当前处于显示状态的数码管
always @ (posedge dri_clk or negedge rst_n) begin
    if (rst_n == 1'b0)
        cnt_sel <= 3'b0;
    else if(flag) begin
        if(cnt_sel < 3'd5)
            cnt_sel <= cnt_sel + 1'b1;
        else
            cnt_sel <= 3'b0;
    end
    else
        cnt_sel <= cnt_sel;
end

//控制数码管位选信号，使 6 位数码管轮流显示
always @ (posedge dri_clk or negedge rst_n) begin
    if(!rst_n) begin
        seg_sel  <= 6'b111111;              //位选信号低电平有效
        num_disp <= 4'b0;
    end
    else begin
        if(en) begin
            case (cnt_sel)
                3'd0 :begin
                    seg_sel  <= 6'b111110;  //显示数码管最低位
                    num_disp <= num[3:0] ;  //显示的数据
                end
                3'd1 :begin
                    seg_sel  <= 6'b111101;  //显示数码管第 1 位
                    num_disp <= num[7:4] ;
                end
                3'd2 :begin
                    seg_sel  <= 6'b111011;  //显示数码管第 2 位
                    num_disp <= num[11:8];
                end
                3'd3 :begin
                    seg_sel  <= 6'b110111;  //显示数码管第 3 位
                    num_disp <= num[15:12];
                end
                3'd4 :begin
                    seg_sel  <= 6'b101111;  //显示数码管第 4 位
                    num_disp <= num[19:16];
                end
                3'd5 :begin
                    seg_sel  <= 6'b011111;  //显示数码管最高位
                    num_disp <= num[23:20];
                end
```

```
                default :begin
                    seg_sel  <= 6'b111111;
                    num_disp <= 4'b0;
                end
            endcase
        end
        else begin
            seg_sel  <= 6'b111111;          //使能信号为0时，所有数码管均不显示
            num_disp <= 4'b0;
        end
    end
end


//控制数码管段选信号，显示字符
always @ (posedge dri_clk or negedge rst_n) begin
    if (!rst_n )
        seg_led <= 8'hc0;
    else begin
        case (num_disp)
            4'd0 : seg_led <= 8'b11000000; //显示数字 0
            4'd1 : seg_led <= 8'b11111001; //显示数字 1
            4'd2 : seg_led <= 8'b10100100; //显示数字 2
            4'd3 : seg_led <= 8'b10110000; //显示数字 3
            4'd4 : seg_led <= 8'b10011001; //显示数字 4
            4'd5 : seg_led <= 8'b10010010; //显示数字 5
            4'd6 : seg_led <= 8'b10000010; //显示数字 6
            4'd7 : seg_led <= 8'b11111000; //显示数字 7
            4'd8 : seg_led <= 8'b10000000; //显示数字 8
            4'd9 : seg_led <= 8'b10010000; //显示数字 9
            4'd10: seg_led <= 8'b111111111;           //不显示任何字符
            default:
                   seg_led <= 8'b11000000;
        endcase
    end
end

endmodule
```

蜂鸣器模块

```
module alarm_music(
    input clk,
    input rst_n,
    input   [19:0]data ,      // 6个数码管要显示的数值
    input   [19:0]set_data ,      // 设定0~999999时间
    input  en_beep,
    output  beep
);
```

```
9.
10.   reg  music_flag        ;       // 标志音频产生信号
11.
12.   always @ (posedge clk or negedge rst_n) begin
13.       if(!rst_n)
14.           music_flag  <= 1'b0;
15.       else if((data ==set_data&& set_data>1'b0)&& (en_beep==1'b1))  //判断按
键是否有效按下
16.           music_flag  <= 1'b1;
17.   end
18.
19.   //参数定义
20.
21.   localparam    M1 = 95600,
22.                             M2 = 85150,
23.                             M3 = 75850,
24.                             M4 = 71600,
25.                             M5 = 63750,
26.                             M6 = 56800,
27.                             M7 = 50600;
28.
29.           parameter   TIMES   = 500;
30.
31.   //信号定义
32.       reg             buzzer_r    ;
33.
34.       reg [16:0]  cnt0    ;//计数每个音符对应的周期
35.       reg [8:0]       cnt1    ;//计数每个音符重复多少次
36.       reg [5:0]       cnt2    ;//一共有多少个音符
37.
38.       reg [16:0]  pre_set;//预装载值
39.       reg     [16:0]  pre_div;//设定占空比
40.
41.       always@(posedge clk or negedge rst_n)begin
42.           if(~rst_n)begin
43.               cnt0 <= 0;
44.           end
45.           else begin
46.               if(cnt0 == pre_set-1)
47.                   cnt0 <= 0;
48.               else
49.                   cnt0 <= cnt0 + 1;
50.           end
51.       end
52.
53.       always@(posedge clk or negedge rst_n)begin
54.           if(~rst_n)begin
55.               cnt1 <= 0;
56.           end
57.           else if(cnt0 == pre_set-1)begin
58.               if(cnt1 == TIMES-1)
```

```
                cnt1 <= 0;
            else
                cnt1 <= cnt1 + 1;
        end
    end

    always@(posedge clk or negedge rst_n)begin
        if(~rst_n)begin
            cnt2 <= 0;
        end
        else if(cnt1 == TIMES-1 && cnt0 == pre_set-1)begin
            if(cnt2 == 32-1)
                cnt2 <= 0;
            else
                cnt2 <= cnt2 + 1;
        end
    end

    //pre_set  查找表 选择每个音频对应的计数周期
    always@(posedge clk or negedge rst_n)begin
        if(~rst_n)begin
            pre_set <= 0;
        end
        else begin
            case(cnt2)
            0 :pre_set <= M1;
            1 :pre_set <= M2;
            2 :pre_set <= M3;
            3 :pre_set <= M1;
            4 :pre_set <= M1;
            5 :pre_set <= M2;
            6 :pre_set <= M3;
            7 :pre_set <= M1;
            8 :pre_set <= M3;
            9 :pre_set <= M4;
            10:pre_set <= M5;
            11:pre_set <= M3;
            12:pre_set <= M4;
            13:pre_set <= M5;
            14:pre_set <= M5;
            15:pre_set <= M6;
            16:pre_set <= M5;
            17:pre_set <= M4;
            18:pre_set <= M3;
            19:pre_set <= M1;
            20:pre_set <= M5;
            21:pre_set <= M6;
            22:pre_set <= M5;
            23:pre_set <= M4;
            24:pre_set <= M3;
            25:pre_set <= M1;
```

```
                26:pre_set <= M2;
                27:pre_set <= M5;
                28:pre_set <= M1;
                29:pre_set <= M2;
                30:pre_set <= M5;
                31:pre_set <= M1;
                default:pre_set <= M1;
                endcase
            end
        end


        always @ (posedge clk or negedge rst_n)begin
            if(~rst_n)begin
                pre_div <= 0;
            end
            else begin
                pre_div <= pre_set>>1;
            end
        end

        //assign pre_div = pre_set>>1;//  /2; 设定占空比为 50%

        always @ (posedge clk or negedge rst_n)begin
            if(~rst_n)begin
                buzzer_r <= 1'b1;
            end
            else if(cnt0 < pre_div)begin
                buzzer_r <= 1'b1;
            end
            else begin
                buzzer_r <= 1'b0;
            end
        end

    assign beep = music_flag ? buzzer_r : 1'b0;



endmodule
```

流水灯模块

```
module flow_light(
    input               sys_clk   ,  //系统时钟
    input               sys_rst_n,  //系统复位，低电平有效
    input   [19:0]data ,      // 6 个数码管要显示的数值
    input   [19:0]set_data ,      // 设定 0～999999 时间
    input  en_beep,
```

```
7.      output  reg  [3:0]  led          //4 个 LED 灯
8.      );
9.
10.     //reg define
11.     reg [23:0] counter;
12.     reg led_flag;

13.     //计数器对系统时钟计数，计时 0.2 秒
14.     always @(posedge sys_clk or negedge sys_rst_n) begin
15.         if (!sys_rst_n)
16.             counter <= 24'd0;
17.         else if (counter < 24'd1000_0000 )
18.             counter <= counter + 1'b1;
19.         else
20.             counter <= 24'd0;
21.     end
22.
23.     always @(posedge sys_clk or negedge sys_rst_n) begin
24.         if (!sys_rst_n)
25.             led_flag <= 1'd0;
26.         else if ((data ==set_data&& set_data>1'b0) && (en_beep==1'b1))
27.             led_flag <= 1'b1;
28.         else
29.             led_flag <= led_flag;
30.     end
31.
32.     //通过移位寄存器控制 IO 口的高低电平，从而改变 LED 的显示状态
33.     always @(posedge sys_clk or negedge sys_rst_n) begin
34.         if (!sys_rst_n)
35.             led <= 4'b0001;
36.         else if((counter == 24'd1000_0000 )&& (led_flag == 1'b1))
37.             led[3:0] <= {led[2:0],led[3]};
38.         else
39.             led <= led;
40.     end
41.
42.     endmodule
```