基于Verilog和FPGA/CPLD的多功能秒表设计
- 杨子超 517030910330 F1703013
- 实验目的
- 实验所用仪器及元器件
- 实验任务
  - 实验内容和任务
  - 设计过程
    - 上电后初始化工作
    - “计时复位”、“计数/暂停”、“显示暂停/显示继续”三个功能的实现
    - 功能状态间切换（包括防抖设计）
    - 如何满足马拉松或长跑运动员的计时需要 （LED灯的使用)
- 实验步骤
- Verilog程序代码
- 实验总结

基于Verilog和FPGA/CPLD的多功能秒表设计

杨子超 517030910330 F1703013

实验目的

- **初步掌握利用Verilog硬件描述语言进行逻辑功能设计的原理和方法。**
- **理解硬件实现方法中的并行性，联系软件实现方法中的并发性。**
- **理解硬件和软件是相辅相成、并在设计和应用方法上的优势互补的特点。**

- **本实验学习积累的Verilog硬件描述语言和对FPGA/CPLD的编程操作，是进行后续《计算机组成原理》部分课程实验，设计实现计算机逻辑的基础。**

实验所用仪器及元器件

DE1-SOC实验板 1套

实验任务

实验内容和任务

- **运用Verilog硬件描述语言，基于DE1-SOC实验板，设计实现一个具有较多功能的计时秒表。**
- **要求将6个数码管设计为具有“分：秒：毫秒”显示，按键的控制动作有：“计时复位”、“计数/暂停”、“显示暂停/显示继续”等。功能能够满足马拉松或长跑运动员的计时需要。**
- **利用示波器观察按键的抖动，设计按键电路的消抖方法。**
- **在实验报告中详细报告自己的设计过程、步骤及Verilog代码。**

设计过程

王赓老师已经给出了部分代码，需要我实现的部分包括上电后的初始化工作，“计时复位”、“计数/暂停”、“显示暂停/显示继续”三个功能的实现，以及功能状态间切换（包括防抖设计）,如何满足马拉松或长跑运动员的计时需要 （LED灯的使用)。

上电后初始化工作

上电后，如无特别用途，所有寄存器的值均清0，但记录显示工作状态寄存器的值为1，即上电后，显示为00:00:00，但不计时。用initial begin end 具体实现。

“计时复位”、“计数/暂停”、“显示暂停/显示继续”三个功能的实现

- **“计时复位”功能的实现**:

计时复位所做工作和上电后初始化工作基本完全相同，即所有寄存器的值均清0，但记录显示工作状态的寄存器值为1，显示00:00:00,但不计时。

- **“计数/暂停”功能的实现**

计数功能的实现，板上的时钟为50MHz，而数显的最小单位为10ms，所以对于板上的时钟来说，每500000个时钟周期，对应着现实中的10ms，记录毫秒低位的寄存器的值应该加1。而加1后，可能涉及到进位问题，由此可能引发连锁的进位问题。同时每计时60分钟，需要在小时的单位上进行处理，这里通过对应的LED灯亮起实现。此部分通过if begin end语句的嵌套具体实现。而暂停的功能则相对简单，即不做任何操作。

- **"显示暂停/显示继续"功能的实现**

如果显示继续，那么将记录计时数据的寄存器的值相应地赋给记录显示数据的寄存器的值即可，通过实例化的六个数显元件即可显示出来。如果显示暂停，那么记录显示数据寄存器的值不变。

功能状态间切换（包括防抖设计）

如果不考虑防抖，那么功能状态间的切换应该是，按键不按下时，对应的输入是高电平，按键按下时，对应的输入是低电平。所以当我们检测到相应的下降沿时，进行状态的切换。而所谓的检测相应的下降沿既可以通过always敏感信号列表中的negedge实现，也可以通过记录上一个时钟周期按键状态的值实现。

考虑防抖，最终采用的是王赓老师在课上提到的一种方法，在上升沿的时候，检测低电平维持的时间是否超过了设定的阈值，如果超过了该阈值那么相应的状态进行切换，否则，认为是振荡引起的高低电平的变化，状态不进行改变。也就是，按下按键，如果一直不弹起，那么功能状态始终不切换，只有按键被松开，同时按键的时间超过设定的阈值，状态才切换。刚开始设定的阈值是200ms，但是由于DE1-SOC的板上已经使用施密特触发器实现了一定的防抖，再加上自己的防抖，导致有的时候按键按快了就会没有反应，所以最终阈值设定的是200ms。这部门的代码实现主要是利用三个独立的if begin end的嵌套。

如何满足马拉松或长跑运动员的计时需要 （LED灯的使用）

马拉松的时长一般在两个小时以上，所以当计时超过一个小时的时候，LEDR3会亮起，当计时超过两个小时的时候，LEDR4会亮起，而如果计时时间小于一个小时，那么LEDR3和LEDR4都处于熄灭状态。所以一共可以完成接近三个小时的计时。当LEDR0用来指示清零键是否被按下，如果被按下那么LEDR0亮起，否则熄灭。而LEDR1和LEDR2不同，当处于计时状态时，LEDR1亮起，否则熄灭；当显示继续的时候LEDR2亮起，当显示暂停的时候LEDR2熄灭。

实验步骤

- 编写Verilog代码，检查无误后，生成相应的symbol；
- 在.bdf顶层文件中，插入生成的symbol，同时连接输入和输出，并对管脚进行合理的命名；
- 通过执行.pcl文件实现管脚的批量分配；
- 进行编译，生成.sof文件；
- 连接DE1-SOC至电脑，设置好，将.sof文件烧录至DE1-SOC上；
- 在DE1-SOC上进行测试。

注： 本次实验实际上没有使用Modelsim ALTERA 10.1d仿真软件进行设计功能验证，但仔细阅读并实践了王赓老师给出的附录A：ModelSim-Altera 10.1d使用方法简介。

Verilog程序代码

```
module stopwatch_01(clk,key_reset,key_start_pause,key_display_stop,
// 时钟输入+ 3个按键；按键按下为0 。板上利用施密特触发器做了一定消抖，效果待测试。
hex0,hex1,hex2,hex3,hex4,hex5,
// 板上的6个7段数码管，每个数码管有7位控制信号。
led0,led1,led2,led3,led4);
// LED发光二极管指示灯，用于指示/测试程序按键状态，若需要，可增加。高电平亮。
input clk,key_reset,key_start_pause,key_display_stop;
output [6:0] hex0,hex1,hex2,hex3,hex4,hex5;
output led0,led1,led2,led3,led4;
reg led0,led1,led2,led3,led4;
reg display_work;
// 显示刷新，即显示寄存器的值实时 更新为计数寄存器的值。
reg counter_work;
// 计数（计时）工作状态，由按键“计时/暂停” 控制。
parameter DELAY_TIME = 10000000;
```

```
// 定义一个常量参数。 10000000 ->200ms;
// 定义6个显示数据（变量）寄存器:
reg [3:0] minute_display_high;
reg [3:0] minute_display_low;
reg [3:0] second_display_high;
reg [3:0] second_display_low;
reg [3:0] msecond_display_high;
reg [3:0] msecond_display_low;
// 定义6个计时数据（变量）寄存器:
reg [3:0] minute_counter_high;
reg [3:0] minute_counter_low;
reg [3:0] second_counter_high;
reg [3:0] second_counter_low;
reg [3:0] msecond_counter_high;
reg [3:0] msecond_counter_low;
reg [31:0] counter_50M; // 计时用计数器， 每个50MHz的clock 为20ns。
// DE1-SOC板上有4个时钟， 都为 50MHz，所以需要500000次20ns之后，才是10ms。
reg reset_1_time; // 消抖动用状态寄存器-- for reset KEY
reg [31:0] counter_reset; // 按键状态时间计数器
reg start_1_time; //消抖动用状态寄存器-- for counter/pause KEY
reg [31:0] counter_start; //按键状态时间计数器
reg display_1_time; //消抖动用状态寄存器-- for KEY_display_refresh/pause
reg [31:0] counter_display; //按键状态时间计数器
reg start; // 工作状态寄存器
reg display; // 工作状态寄存器
// sevenseg模块为4位的BCD码至7段LED的译码器，
//下面实例化6个LED数码管的各自译码器。

sevenseg LED8_minute_display_high ( minute_display_high, hex5 );
sevenseg LED8_minute_display_low ( minute_display_low, hex4 );
sevenseg LED8_second_display_high( second_display_high, hex3 );
sevenseg LED8_second_display_low ( second_display_low, hex2 );
sevenseg LED8_msecond_display_high( msecond_display_high, hex1 );
sevenseg LED8_msecond_display_low ( msecond_display_low, hex0 );



initial
begin   // 上电后 做好准备工作  不计时  显示00:00:00
    start_1_time = 1;
    display_1_time = 1;
    reset_1_time = 1;
```

```
    counter_work = 0;
    display_work = 1;
    led0=0;led1=0;led2=1;
    counter_display=0;
    counter_reset=0;
    counter_start=0;
    counter_50M=0;
    led3=0;
    led4=0;
    msecond_counter_low = 0; //清0
    msecond_counter_high = 0;
    second_counter_low = 0;
    second_counter_high = 0;
    minute_counter_low = 0;
    minute_counter_high = 0;


end
always @ (posedge clk) // 每一个时钟上升沿开始触发下面的逻辑,


begin

    if(key_reset==1)
    begin
        if(reset_1_time==0)    // 按键弹起
        begin
            if(counter_reset>=DELAY_TIME/10)  // 按键超过20ms
            begin
                msecond_counter_low = 0; //计数 显示 清0
                msecond_counter_high = 0;
                second_counter_low = 0;
                second_counter_high = 0;
                minute_counter_low = 0;
                minute_counter_high = 0;
                counter_work = 0;   //不工作
                display_work = 1;  //显示
                start_1_time = 1;
                display_1_time = 1;
                led3=0;led4=0;
            end
            counter_reset=0;//按键计时状态清0
```

```
        end
        reset_1_time = 1; //缓冲为1
        led0=0;  // reset灯不亮
    end
    else // key_reset=0; 清零键被按下
    begin
        led0=1; // reset灯亮
        reset_1_time = 0;  // 缓冲为0
        counter_reset=counter_reset+1; //按键计时状态+1
    end

    if(key_start_pause==1)
    begin
        if(start_1_time==0)
        begin
            if(counter_start>=DELAY_TIME/10)
            begin
                counter_work = 1-counter_work ;   //翻转
            end
            counter_start=0;//按键计时状态清0
        end
        start_1_time = 1; //缓冲为1

    end
    else // key_start_pause=0; 停止计时被按下
    begin

        start_1_time = 0;  // 缓冲为0
        counter_start=counter_start+1; //按键计时状态+1
    end


        if(key_display_stop==1)
    begin
        if(display_1_time==0)
        begin
            if(counter_display>=DELAY_TIME/10)
            begin
                display_work = 1-display_work ;   //翻转
            end
            counter_display=0;//按键计时状态清0
        end
```

```
142         display_1_time = 1; //缓冲为1
143 
144     end
145     else // key_display_stop=0; 不显示键 被按下
146     begin
147 
148         display_1_time = 0;  // 缓冲为0
149         counter_display=counter_display+1; //按键计时状态+1
150     end
151 
152 
153     if (counter_work == 1)
154     begin  // 如果计时
155         counter_50M = counter_50M + 1;
156         led1=1;
157     end
158     else
159         led1=0;   //计时灯亮
160 
161 
162     if (counter_50M == 500000) //10ms到了
163     begin
164         counter_50M = 0;
165         msecond_counter_low = msecond_counter_low + 1;
166         if (msecond_counter_low == 10)   // 可能需要进位
167         begin
168             msecond_counter_low = 0;
169             msecond_counter_high = msecond_counter_high + 1;
170             if (msecond_counter_high == 10)
171             begin
172                 msecond_counter_high = 0;
173                 second_counter_low = second_counter_low + 1;
174                 if (second_counter_low == 10)
175                 begin
176                     second_counter_low = 0;
177                     second_counter_high = second_counter_high + 1;
178                     if (second_counter_high == 6)
179                     begin
180                         second_counter_high = 0;
181                         minute_counter_low = minute_counter_low + 1;
182                         if (minute_counter_low == 10)
183                         begin
```

```
                                minute_counter_low = 0;
                                minute_counter_high = minute_counter_high
+ 1;
                                if (minute_counter_high == 6)
                                begin  // 60分钟后 led3亮 120分钟后 led4亮
                                    minute_counter_high = 0;
                                    if(led3==0)
                                        led3=1;
                                    else
                                        if(led4==0)
                                            led4=1;
                                end
                            end
                        end
                    end
                end
            end
        end



    if (display_work)  //如果显示
    begin
        led2=1;   // 状态显示灯亮
        msecond_display_low = msecond_counter_low;
        msecond_display_high = msecond_counter_high;
        second_display_low = second_counter_low;
        second_display_high = second_counter_high;
        minute_display_low = minute_counter_low;
        minute_display_high = minute_counter_high;
    end
    else
        led2=0;
end
endmodule

//4bit的BCD码至7段LED数码管译码器模块
//可供实例化共6个显示译码模块
module sevenseg ( data, ledsegments);
input [3:0] data;
output ledsegments;
reg [6:0] ledsegments;
```

```
always @ (*)
case(data)
// gfe_dcba // 7段LED数码管的位段编号
// 654_3210 // DE1-SOC板上的信号位编号
0: ledsegments = 7'b100_0000; // DE1-SOC板上的数码管为共阳极接法。
1: ledsegments = 7'b111_1001;
2: ledsegments = 7'b010_0100;
3: ledsegments = 7'b011_0000;
4: ledsegments = 7'b001_1001;
5: ledsegments = 7'b001_0010;
6: ledsegments = 7'b000_0010;
7: ledsegments = 7'b111_1000;
8: ledsegments = 7'b000_0000;
9: ledsegments = 7'b001_0000;
default: ledsegments = 7'b111_1111; // 其它值时全灭。
endcase
endmodule
```

实验总结

本次实验基本上顺利实现了实验的任务，达到了实验的目的。初步掌握了利用Verilog硬件描述语言和大规模可编程逻辑器件进行逻辑功能设计的原理和方法。在一个相对较高的层次上，利用软件和硬件的相辅相成和优势互补，设计实现了一个具有较多功能的计时秒表，让我初窥门径，同时也让体会到了相当的乐趣。

稍具体的一些心得体会是，严格按照王赓老师给出的实验指导书进行操作，同时多向助教请教，多与同学交流，充分利用DE1-SOC官方给出的手册。更具体的一些体会是，在动手写Verilog代码前先想清楚，才能写明白；在编译出现问题的时候，验证自己的实验器材型号是否正确，输入和输出连接和命名是否正确；利用.pcl批量导入管脚。