多功能秒表实验报告

陈子轩 516030910545

2018 年 5 月 22 日

1 实验目的

1. 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。
2. 理解和掌握运用大规模可编程逻辑器件进行逻辑设计的原理和方法。
3. 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。
4. 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。
5. 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。

2 实验内容和任务

1. 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。
2. 要求将 8 个数码管设计为具有“时：分：秒：毫秒”显示，按键的基本控制动作有 3 个：“计时复位”、“计数/暂停”、“显示暂停/显示继续”。功能能够满足马拉松或长跑运动员的计时需要。
3. 利用示波器观察按键的抖动，设计按键电路的消抖方法。
4. 在实验报告中详细报告自己的设计过程、步骤及Verilog 代码。

3 实验仪器

Altera – DE1-SOC实验板1套，示波器1台，数字万用表1台。

4 实验设计思路

4.1 秒表设计思路

1表示工作状态 创建三个寄存器变量counter_work, display_work, 和reset_work来表示此时秒表的状态如果counter_work为1说明此时计时器是工作的；如果display_work为1说明此时显示器是工作的；如果reset_work说明此时需要将所有数码管置零。

2判定工作状态 接下来是如何判断这三个寄存器的工作状态。我的方法是设置三个寄存器保存上一个时钟上跳沿三个按键key_reset, key_start_pause, key_display_stop的状态。如果上一个状态按键没有按下(状态为1)，该时钟上跳沿按键按下（状态为0），说明是首次按下，我就将上述三个寄存器变量（counter_work, display_work, 和reset_work）的值反转。

3实现工作状态 最后就是对应个状态的功能实现了，对于“计数/暂停”功能，设置6个计时器分别对应6个数码管。设置一个计数器。如果计数器达到500000，由于时钟是50MHz的，所以说明距上次计时器状态更新已经过了10ms，则依次进位更新各计时器。对于“显示暂停/显示继续”功能，则简单的将计时器的值显示在数码管上。对于“计时复位”功能，则简单的将各计时器置零即可。

4.2 防抖设计思路

上述设计思路并没有考虑按键抖动的情况，所以接下来还需针对抖动进行设计。我的设计思路是创建三个按键状态时间计数器，它记录了假定本次按键按下距离上次判定按键按下的时间。如果这个时间小于某一个值，我们就判定这次按键按下的原因是因为抖动，所以实际上按键并未按下。所以我们的设计只需对4.1中的第二步做修改即可。

5 部分实验代码

```
always @ (posedge clk) //每一个时钟上升沿开始触发下面的逻辑
    begin
    //消抖寄存器更新
```

```
counter_reset = #1 counter_reset + 1;
counter_start = #1 counter_start + 1;
counter_display = #1 counter_display + 1;


//判断是否触发开关
if(key_start_pause == 0 && previous_key_start_pause == 1
    && counter_start >= DELAY_TIME)
    //如果开关从1变0，并且在200ms时间之内没有触发开关
    begin
    counter_start = #1 0;
    counter_work = #1 counter_work + 1;
    end
if(key_display_stop == 0 && previous_key_display_stop == 1
    && counter_display >= DELAY_TIME)
    begin
    display_work = #1 display_work + 1;
    counter_display = #1 0;
    end
if(key_reset == 0 && previous_key_reset == 1 && counter_reset >= DELAY_TIME)
    begin
    reset_work = #1 1;
    counter_reset = #1 0;
    end



//触发了“暂停计时”开关
if(counter_work)
    begin
    counter_50M = #1 counter_50M + 1;
    if(counter_50M == 500000)
        begin
```

```
        counter_50M = #1 0;
        msecond_counter_low = #1 msecond_counter_low + 1;
        if(msecond_counter_low == 9)
            begin
            msecond_counter_low = #1 0;
            msecond_counter_high = #1 msecond_counter_high + 1;
            if(msecond_counter_high == 9)
                begin
                msecond_counter_high = #1 0;
                second_counter_low = #1 second_counter_low + 1;
                if(second_counter_low == 9)
                    begin
                    second_counter_low = #1 0;
                    second_counter_high = #1 second_counter_high + 1;
                    if(second_counter_high == 5)
                        begin
                        second_counter_high = #1 0;
                        minute_counter_low = #1 minute_counter_low + 1;
                        if(minute_counter_low == 9)
                            begin
                            minute_counter_low = #1 0;
                            minute_counter_high = #1 minute_counter_high + 1;
                            if(minute_counter_high == 5)
                                begin
                                minute_counter_high = #1 0;
                                end
                            end
                        end
                    end
                end
            end
    end
```

```
//触发了“系统复位”开关
if(reset_work == 1)
    begin
    minute_counter_high = #1 0;
    minute_counter_low =#1 0;
    msecond_counter_high = #1 0;
    msecond_counter_low = #1 0;
    second_counter_high =#1  0;
    second_counter_low = #1 0;
    reset_work = #1 0;
    end

//触发了“暂停显示”开关
if(display_work == 0)
    begin
    minute_display_high = #1 minute_counter_high;
    minute_display_low = #1 minute_counter_low;
    second_display_high = #1 second_counter_high;
    second_display_low = #1 second_counter_low;
    msecond_display_high = #1 msecond_counter_high;
    msecond_display_low = #1 msecond_counter_low;
    end

//更新上周期的的寄存器状态的记录
previous_key_display_stop = #1 key_display_stop;
previous_key_start_pause = #1 key_start_pause;
previous_key_reset = #1 key_reset;
end
```

6 实验感想

本次实验我学到了Quartus平台的使用方法和verilog硬件编程语言的语

法以及如何运用可编程逻辑器件进行逻辑设计的方法。后者是我以前从没接触到的，所以一开始上手感觉比较新奇，我发现我们在硬件编程的所有逻辑都可以用逻辑器件进行表达，所以这比较强调我们软硬件的综合能力。