基于 Verilog 和 FPGA 的多功能秒表设计实验报告

毛咏

2018 年 5 月 14 日

目录

1 实验要求

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。

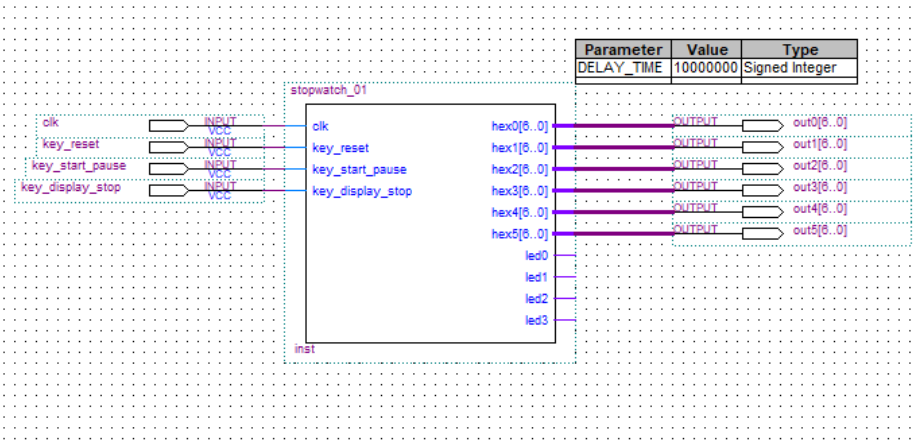
2 设计思路

2.1 整体思路

毫秒部分每位满 10 进 1，分秒低位满 10 进 1，高位满 6 进 1。计时按钮控制是否在时钟周期到达制定常数时加毫秒低位，显示按钮控制是否由变量改变导致高低电平变化，重置按钮控制变量清零以及计时与显示状态的复位。

2.2 BDF 设计

包含 4 个输入，一个是时钟脉冲，依据题意设计为 50MHz，即一个周期 20ns。另外三个输入分别为重置、计时、显示的电位高低，电位变化将导致计时、显示状态的改变。输出中的前 6 个 7 位分别对应了 6 个显示数字的各个针脚电位高低。led 可用于显示是否处于计时以及是否实时显示的状态，实验中未进行实现（因为实现过程与上述类似）。具体设计如下图。



2.3 verilog 代码设计

详细请见代码注释。消抖的实现方式是记录之前的电位高低，随时与当前的电位高低对比，如果之前基本不是高电平，那么现在的低电平将不会导致状态的改变；现在的低电平只有在之前是高电平时才会改变计时与显示的状态。当然 reset 复位不受影响，因为从肉眼效果来看几次高频率的复位可以基本等效成 1 次复位。

3 具体步骤

3.1 初始化

设备驱动安装、在新建设备时选择实验设备的型号、设置时钟脉冲频率等。

3.2 实现代码

根据设计要求把代码的主要功能部分全部实现，在给出的代码中自己填写的主要包括

1) 具体的 6 位数字更改解决方案

2) 计时状态更改、显示状态更改、复位解决方案

3) 消抖解决方案

具体代码实现请看本实验报告的第 4 部分完整代码

3.3 完成 BDF 设计

根据实验要求把四个输入和六个输出分配好，用 input 和 output 的 pin 连接，以及保证基本的美观性和可读性。

3.4 针脚分配

对于输入的 button 每个都有 1 个针脚，对于输出的数字每个都有 7 个针脚，按照实验手册进行匹配。

3.5 编译与写入

把文件编译，写入开发板中。

4 完整代码

注：由于 latex 排版代码不是很美观所以采用截图，如果需要完整代码，请至 Github 下载。地址 https://github.com/yyong119/EI332SourceCode

//**** 之中的注释是我认为比较关键的地方

```v
// ============================================================
//
// The counter is designed by a series mode. / asynchronous mode. 即异步进位
// use "=" to give value to hour_counter_high and so on. 异步操作/阻塞赋值方式
//
// 3 key: key_reset/系统复位, key_start_pause/暂停计时, key_display_stop/暂停显示
//
// ============================================================
module stopwatch_01(clk, key_reset, key_start_pause, key_display_stop,
        hex0, hex1, hex2, hex3, hex4, hex5, led0, led1, led2, led3);
        //key_reset复位,key_start_pause开始暂停计时,key_display_stop开始暂停显示

    input clk, key_reset, key_start_pause, key_display_stop;
    output[6:0] hex0, hex1, hex2, hex3, hex4, hex5;
    output led0, led1, led2, led3;
    reg led0, led1, led2, led3;
    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;//计数工作状态,由按键key_start_pause控制
    parameter DELAY_TIME = 10000000;//定义一个常量参数.10000000->200ms

    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;//定义6个显示数据(变量)寄存器

    reg[31:0] counter_50M;//计时用计数器,每个50MHz的clock为20ns.
    //若选择板上的50MHz时钟, 需要500000次20ns后,才是10ms.
    reg reset_1_time;//消抖动用状态寄存器--for reset KEY_display_stop
    reg[31:0] counter_reset;//按键状态时间计数器
    //**start_1_time为1计时0不计时**
    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
    //**display_1_time为1不实时显示0实时显示**
    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.
    //**有些设备连接时hex3,hex2和hex1,hex0需要换位**
    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);

    always@(posedge clk)
    begin
        if (key_start_pause == 0 && start_1_time == 1)//如果计时按钮被按下,且之前不是按下的
            counter_work = 1 - counter_work;//反转是否计时状态
        if (key_display_stop == 0 && display_1_time == 1)//如果显示按钮被按下,且之前不是按下的
            display_work = 1 - display_work;//反转是否显示状态
        if (key_reset == 1'b0)//如果复位被按下
        begin
            msecond_counter_low = 0;
            msecond_counter_high  = 0;
            second_counter_low = 0;
            second_counter_high = 0;
            minute_counter_low = 0;
            minute_counter_high = 0;
            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;
            //**下面这部分是根据自己的想法加的:复位后默认"不计时"且"显示计时"**
            counter_work = 0;
            display_work = 0;
        end

        //**记录这一次的button状态以供下次使用**
        start_1_time = key_start_pause;
        display_1_time = key_display_stop;

        counter_50M = counter_50M + 1;
```

```
        if (counter_50M == 500000)//如果已经经过了500000个时钟周期
        begin
            counter_50M = 0;//重新计算时钟周期

            //**如果处于计时开始状态,则计时器内部更新**
            if (counter_work == 1)
            begin
                msecond_counter_low =  msecond_counter_low + 1;
                if (msecond_counter_low == 4'b1010)//满10进1
                begin
                    msecond_counter_low = 0;
                    msecond_counter_high = msecond_counter_high + 1;
                    if (msecond_counter_high == 4'b1010)//满10进1
                    begin
                        msecond_counter_high = 0;
                        second_counter_low = second_counter_low + 1;
                        if (second_counter_low == 4'b1010)//满10进1
                        begin
                            second_counter_low = 0;
                            second_counter_high = second_counter_high + 1;
                            if (second_counter_high == 4'b0110)//满6进1
                            begin
                                second_counter_high = 0;
                                minute_counter_low = minute_counter_low + 1;
                                if (minute_counter_low == 4'b1010)//满10进1
                                begin
                                    minute_counter_low = 0;
                                    minute_counter_high = minute_counter_high + 1;
                                    if (minute_counter_high == 4'b0110)//满6进1(没有hour位因此清零)
                                        minute_counter_high = 0;
                                end
                            end
                        end
                    end
                end
            end
            //**如果处于显示更新状态,则显示更新为内部数值**
            if (display_work == 0)
            begin
                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
        end



    end
endmodule


//4bit的BCD码至7段LED数码管译码器模块
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  //DE2板上的信号位编号
            0:ledsegments = 7'b100_0000;//DE2C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;//其他值时全灭
        endcase
endmodule
```