计算机组成实验报告

实验 1：多功能秒表设计

姓 名: 郭 嘉 宋
学 号: 517030910374
班 号: F1703015

上海交通大学
IEEE 试点班

2020 年 2 月 8 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 实验所用的仪器以及元器件

DE1-SOC 实验板 1 套
示波器 1 台
数字万用表 1 台

3 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 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。
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。

4 实验原理和结果展示

4.1 完整代码

```
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 );
// 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;
 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（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 );



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

```
// 进行计时后各部分的刷新工作
 begin

        if (key_reset==0)

                begin
                        led2=1;
                        counter_reset = counter_reset+1;
                        if (counter_reset > DELAY_TIME)
                                reset_1_time=1;
                        else
                                reset_1_time=0;
                end
        else
                begin
                led2=0;
                counter_reset=0;
                reset_1_time=0;
                end

        if (key_start_pause==0)
                begin
                led1=1;
                counter_start = counter_start+1;
                if (counter_start == DELAY_TIME )
                                start_1_time = !start_1_time;

                end
        else
        begin
                counter_start=0;
                led1=0;
        end


        if (key_display_stop==0)
                begin
                led0=1;
                counter_display = counter_display+1;
                if (counter_display == DELAY_TIME )
                                display_1_time = !display_1_time;

                end
        else
        begin
                counter_display = 0;
                led0=0;
        end


if (reset_1_time==1)
```

```
                begin
                        minute_counter_high=0;
                        minute_counter_low = 0;
                        second_counter_high = 0;
                        second_counter_low =0;
                        msecond_counter_high=0;
                        msecond_counter_low=0;
                end


if (display_1_time==0)
begin
minute_display_high=minute_counter_high;
minute_display_low=minute_counter_low;
second_display_high=second_counter_high;
second_display_low=second_counter_low;
msecond_display_high=msecond_counter_high;
msecond_display_low=msecond_counter_low;
end
else
begin
minute_display_high=11;
minute_display_low=11;
second_display_high=11;
second_display_low=11;
msecond_display_high=11;
msecond_display_low=11;
end



if (start_1_time==0)
begin
        led3=1;
        if (counter_50M ==500000)
        begin
        counter_50M = 0;
                if (msecond_counter_low < 4'd9)
                msecond_counter_low = msecond_counter_low+1;
                else
                begin
                msecond_counter_low = 0;
                if (msecond_counter_high < 4'd9)
                msecond_counter_high = msecond_counter_high+1;
                else
                        begin
                        msecond_counter_high = 0;
                        if (second_counter_low < 4'd9)
                        second_counter_low = second_counter_low+1;
                        else
                                begin
```

```
                                        second_counter_low = 0;
                                        if (second_counter_high <4'd5)
                                        second_counter_high = second_counter_high+1;
                                        else
                        begin
                        second_counter_high = 0;
                        if (minute_counter_low < 4'd9)
                                minute_counter_low = minute_counter_low + 1;
                        else
                                begin
                                minute_counter_low = 0;
                                if (minute_counter_high < 4'd9)
                                        minute_counter_high = minute_counter_high + 1;
                                else
                                        begin
                                        minute_counter_high=0;
                                        second_counter_high = 0;
                                        second_counter_low =0;
                                        msecond_counter_high=0;
                                        msecond_counter_low=0;
                                        end
                                end
                        end
                                        end
                                end
                end
        end



        else
        counter_50M = counter_50M+1;
end


else
begin
counter_50M=0;
led3=0;
end

//此处功能代码省略，由同学自行设计。
 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
```

4.2 原理解释

1. 消抖

对每个按键进行判定，时间超过 200ms 后才判定按下。DELAY TIME = 10000000，因此 DELAY TIME 个 CLK 周期正好是 200ms。

```
        if (key_reset==0)

                begin
                        led2=1;
                        counter_reset = counter_reset+1;
                        if (counter_reset > DELAY_TIME)
                                reset_1_time=1;
                        else
                                reset_1_time=0;
                end
        else
                begin
                led2=0;
                counter_reset=0;
                reset_1_time=0;
                end
```

2. 基本秒表功能

多重 if 语句判断，只需在后一位需要进位时判断前一位的数字是否溢出进行分支即可，完整代码如上。

3. 功能键

key3 设定为复位键，若按下则会置于 0.

key2 设定为暂停键，可以暂停秒表计时与否.

key1 设定为显示键，按下将 HEX 的传入参数调为 10，这样即可在译码后不显示内容.

4. 功能键分离

此程序将 3 个功能键分离可以同时按下或在任意时刻按下某个功能键，因为此时将 3 个功能键存为 3 个状态，判断状态即可。

5. LED 显示

LED3 设定为秒表显示，若秒表在走则 LED3 亮。

LED2,LED1,LED0 分别与 KEY3，KEY2，KEY1 绑定，表示按下与否的状态。

5 实验总结

这次实验是我第一次使用 Quartus 制作 FPGA，现在慢慢熟悉了流程。逻辑层面我第一次使用 Verilog 编写硬件，对 Verilog 有了初步的了解，希望之后慢慢加深对 Verilog 的熟练程度。同时 FPGA 对系统 Debug 时并不像软件那样简单，很多问题需要反复耐心排查才可看出，也意识到硬件层面的困难，希望继续学习 FGPA 的设计方法，继续提升自己！