实验一：基于 Verilog 和 FPGA/CPLD 的多功能秒表设计

杨健邦 515030910223

一、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。

二、实验内容：

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。

三、预习内容：

1. 学习和掌握 Verilog HDL 硬件描述语言。
2. 熟悉针对 Altera 公司 FPGA 开发的 Quartus Ⅱ 13.1 软件开发界面。
3. 掌握利用 Modelsim ALTERA 10.1d 仿真软件进行设计功能验证的方法。

四、实验器材：

1. Altera-DE1-SOC 实验板套件 1 套
2. 示波器 1 台
3. 数字万用表 1 台

五、设计思想和方法：

1. 基本功能的实现：每次在时钟的上升沿时，给一个计数器加一，直到计数器加到 500000 次之后，给毫秒位加 1，之后一直判断进位。之后根据按键状态来决定是否将实际值付给显示值的寄存器，最后通过 sevenseg 转化为七段数码管的显示。当各位到达最大值之后，所有位都重新置为 0。重新开始计时。
2. 按键消抖：通过多次采样的方法，当第一次检测到低电平，经过一个 delay 之后，连续几个周期都检测到按键按下产生的低电平则判断按键按下，此时才会产生按键的效果。这样虽然会使得快速按键没有效果，但是却很有效地消除了按键抖动的影响，权衡利弊，故采用这种方法。
3. 长按的处理：在解决的按键消抖的问题之后，我发现持续不断地按下按键会导致秒表会不断地暂停又开始，这不符合日常我们的逻辑。在按下按键之后，再放开重按之前，应该只会产生执行一次的效果。基本的解决方法是，用一个寄存器去记录按键在按下之后是否被放开过，这样，当检测到按键按下之后，如果发现按键没有被放开过，就不产生此次按键的效果。

六、实验感想：

1. 这次实验让我熟悉了 Verilog HDL 硬件编程语言的语法，同时也熟悉了 FPGA 的操作流程。
2. 这次实验让我改变了以往编程语言顺序执行的传统观念，让我从不同的角度去思考问题，学会了并行逻辑的编写。

```
module stopwatch_01(clk, key_reset, key_start_pause, key_display_stop,
                    hex0, hex1, hex2, hex3, hex4, hex5,
                    led0, led1, led2, led3);
   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; // 0 for refresh, 1 for stop
   reg            counter_work; // 0 for refresh, 1 for stop
   parameter      DELAY_TIME = 10000000;

   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;

   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; //clock is 50MHz. 500000 x 20ns = 10ms
   reg [31:0]     counter_start_pause;
   reg [31:0]     counter_reset;
   reg [31:0]     counter_display_refresh;

   reg            reset_1_time; //for reset KEY
   reg [31:0]     counter_reset;

   reg            start_1_time; //for counter/pause KEY
   reg [31:0]     counter_start;

   reg            display_1_time; //for KEY_display_refresh/pause
   reg [31:0]     counter_display;

   //Used for testing long pressing
   reg            start; // 0 means released already
   reg            display; // 0 means released already

   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
      // check reset KEY
      if(reset_1_time == 0 && key_reset == 0) reset_1_time = 1;
      if(reset_1_time == 1) counter_reset = counter_reset + 1;
      if(reset_1_time == 1 && counter_reset >= DELAY_TIME)
         begin
            if(key_reset == 0)
               begin
                  display_work = 0;
                  counter_work = 0;

                  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= 0;
                  msecond_display_high= 0;
                  second_display_low= 0;
                  second_display_high= 0;
                  minute_display_low= 0;
                  minute_display_high= 0;

                  counter_50M = 0;
                  counter_start_pause= 0;
                  counter_reset = 0;
                  counter_display_refresh= 0;


                  reset_1_time = 0;
                  counter_reset = 0;

                  start_1_time = 0;
                  counter_start = 0;

                  display_1_time = 0;
                  counter_display = 0;

                  start = 0;
                  display = 0;
```

```
                end
            reset_1_time = 0;
            counter_reset = 0;
        end

    //check start pause KEY
    if(start_1_time == 0 && key_start_pause == 0) start_1_time = 1;
    if(start_1_time == 1) counter_start = counter_start + 1;
    if(start_1_time == 1 && counter_start >= DELAY_TIME) ↵
counter_start_pause= counter_start_pause+ 1;
    if(start_1_time == 1 && counter_start_pause>= 5)
        begin
            if(key_start_pause == 0 && start == 0)
                begin
                    counter_work = ~counter_work;
                    start = 1;
                end
            start_1_time = 0;
            counter_start = 0;
        end
    //testing long pressing
    if(start_1_time == 0 && key_start_pause == 1) start_1_time = 1;
    if(start_1_time == 1) counter_start = counter_start + 1;
    if(start_1_time == 1 && counter_start >= DELAY_TIME)
        begin
            if(key_start_pause == 1 && start == 1) start = 0;
            start_1_time = 0;
            counter_start = 0;
        end

    //check display stop KEY
    if(display_1_time == 0 && key_display_stop == 0) display_1_time = 1;
    if(display_1_time == 1) counter_display = counter_display + 1;
    if(display_1_time == 1 && counter_display >= DELAY_TIME)
        begin
            if(key_display_stop == 0 && display == 0)
                begin
                    display_work = ~display_work;
                    display = 1;
                end
            display_1_time = 0;
            counter_display = 0;
        end
    //testing long pressing
    if(display_1_time == 0 && key_display_stop == 1) display_1_time = 1;
    if(display_1_time == 1) counter_display = counter_display + 1;
    if(display_1_time == 1 && counter_display >= DELAY_TIME)
        begin
```

```
            if(key_display_stop == 1 && display == 1) display = 0;
            display_1_time = 0;
            counter_display = 0;
        end

    // count
    if(counter_work == 0)
        begin
            counter_50M = counter_50M + 1;
            if(counter_50M >= 500000)
                begin
                    counter_50M = 0;
                    msecond_counter_low= msecond_counter_low+ 1;

                    if(msecond_counter_low>= 10)
                        begin
                            msecond_counter_low= 0;
                            msecond_counter_high= msecond_counter_high+ 1;
                        end

                    if(msecond_counter_high>= 10)
                        begin
                            msecond_counter_high= 0;
                            second_counter_low= second_counter_low+ 1;
                        end

                    if(second_counter_low>= 10)
                        begin
                            second_counter_low= 0;
                            second_counter_high= second_counter_high+ 1;
                        end

                    if(second_counter_high>= 6)
                        begin
                            second_counter_high= 0;
                            minute_counter_low= minute_counter_low+ 1;
                        end

                    if(minute_counter_low>= 10)
                        begin
                            minute_counter_low= 0;
                            minute_counter_high= minute_counter_high+ 1;
                        end

                    if(minute_counter_high>= 10) minute_counter_high= 0;
                end
        end
    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

         //for LEDs
         led0 = ~counter_work;
         led1 = counter_work;
         led2 = ~display_work;
         led3 = display_work;
      end
endmodule
```