《基于 Verilog 和 FPGA/CPLD 的多功能秒表设计 》实验报告

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。

实验内容和任务：

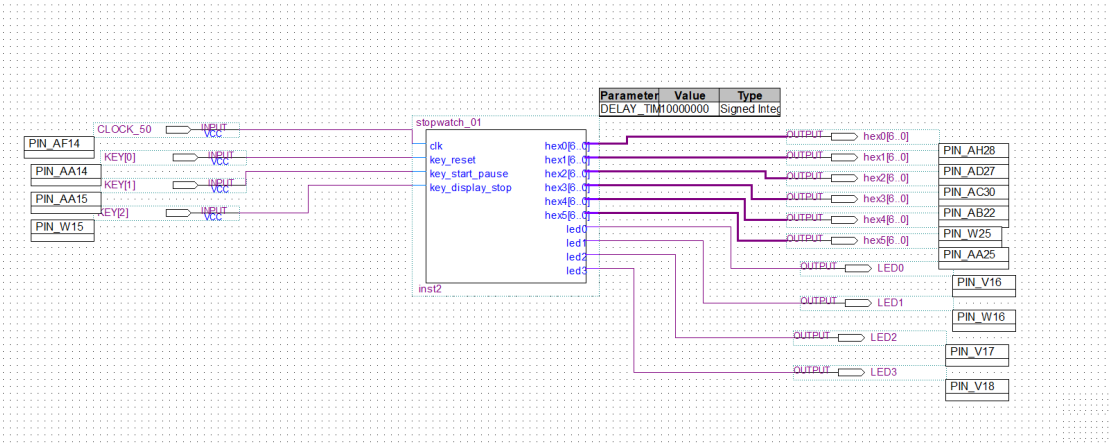
1. 运用 Verilog 硬件描述语言，基于 DE1-SOC 实验板，设计实现一个具有较多功能的计时秒表。
2. 要求将 6 个数码管设计为具有“分：秒：毫秒”显示，按键的控制动作有：“计时复位”、“计数/暂停”、“显示暂停/显示继续”等。功能能够满足马拉松或长跑运动员的计时需要。
3. 利用示波器观察按键的抖动，设计按键电路的消抖方法。
4. 在实验报告中详细报告自己的设计过程、步骤及 Verilog 代码。

实验仪器

- DE1-SOC 实验板
- 软件： Altera Quartus II 13.1

实验内容

电路设计



电路情况 stopwatch 模块有四个输入：时钟输入 CLOCK_50，3 个按键输入 KEY[0:3]分别对应功能按键“计时复位”、“计数/暂停”、“显示暂停/显示继续”。同时，模块将秒表的时间和调试状态输出到 hex0[6:0]~hex5[6:0]六个七段数码管和 4 个 LED 灯中。

硬件设计代码

按键状态及消抖

```
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
    //start 按键
        if(key_start_pause && !start_1_time)begin //消抖
            counter_start <= counter_start+1;
            if  (counter_start == 500000)begin//等10ms
                start_1_time <= ~start_1_time;
                counter_start<=0;

                start <= ~start;
            end

        end
        else if(!key_start_pause && start_1_time)begin
            counter_start <= counter_start+1;
            if(counter_start == 500000)begin
                start_1_time <= ~start_1_time;
                counter_start<=0;
```

```
            end
        end
        else begin
            counter_start <=0;
        end

    //reset 按键
        if(key_reset && !reset_1_time)begin //消抖
            counter_reset <= counter_reset+1;
            if  (counter_reset == 500000)begin//等10ms
                reset_1_time <= ~reset_1_time;
                counter_reset<=0;

                reset <= ~reset;
            end

        end
        else if(!key_reset && reset_1_time)begin
            counter_reset <= counter_reset+1;
            if(counter_reset == 500000)begin
                reset_1_time <= ~reset_1_time;
                counter_reset<=0;
            end
        end
        else begin
            counter_reset <=0;
        end
    //display 按键
        if(key_display_stop && !display_1_time)begin //消抖
            counter_display <= counter_display+1;
            if  (counter_display == 500000)begin//等10ms
                display_1_time <= ~display_1_time;
                counter_display<=0;

                display <= ~display;
            end
        end
        else if(!key_display_stop && display_1_time)begin //从按下状
态中恢复出来
            counter_display <= counter_display+1;
            if(counter_display == 500000)begin
                display_1_time <= ~display_1_time;
                counter_display<=0;
            end
```

```
        end
        else begin
            counter_display <=0;
        end
    end
```

秒表复位（RESET）

```
always @ (posedge clk) // 每一个时钟上升沿开始触发下面的逻辑，
    begin
    // ..... 按键部分代码
    if(reset) 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;
              reset <= 0;
              start <= 0;
              display<=1;
        end
end
```

计时

```
always @ (posedge clk) // 每一个时钟上升沿开始触发下面的逻辑，
    begin
    // ..... 按键部分代码
        if (start)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
                end
            end
end
```

展示

```
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
        //.... 按键部分代码
        if(display)
            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
    end
```

显示译码模块

```
//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

实验结果

将设计代码编译后，烧录进入开发板中，秒表计时准确，按键的消抖效果良好，响应及时准确。

实验过程中的问题

- **第一次烧录后发现，按键响应存在问题：**

  第一次的代码实现未对按键进行消抖，按键按下后状态位转换存在问题。

- **秒表时间略慢于设计预期:**

  检查后发现，计时部分代码存在问题：

```
verilog   if(counter_50M ==500000)begin//10ms
counter_50M =0;                  msecond_counter_low =
msecond_counter_low + 1;
```

由于采用了阻塞赋值的方式，导致计数更新阻塞，使得秒表计时慢于实际设计时间。改用非阻塞赋值后，计时恢复正常

- **按键按下后秒表计时停止：**

同样是由于阻塞赋值的方式，导致计时阻塞。

感想

通过本次实验我比较熟练的掌握了 Verilog 硬件描述语言和对 FPGA/CPLD 的编程操作，也熟悉了使用 Altera Quartus 和 ModelSim 进行 FPGA 开发和模拟的全过程。