基于 Verilog 和 FPGA/CPLD 的多功能秒表设计

曹金坤 515260910022

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

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. 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。

实验设计

1. 实验模块的顶层设计部分为一个stopwatch模块：

```
module
stopwatch_01(clk,key_reset,key_start_pause,key_display_stop,
                    hex0,hex1,hex2,hex3,hex4,hex5,
                    led0,led1,led2,led3 );
```

参数含义如下：

- clk：时钟参数，用于驱动程序的周期性运行；
- ket_reset：绑定了用于重置秒表当前计数的按钮的指示变量；
- key_start_pause：绑定了用于暂停和开始秒表计数的按钮的指示变量；
- ket_display_stop：绑定了用于暂停和开始秒表计数显示的按钮的指示变量；
- hex0 - hex5：对应于用于显示计时的六个数字显示器的驱动变量，控制了显示器的

数字显示；
- led0 - led3：对应于开发板上的4个led指示灯

2. 基本操作逻辑：

```
always @ (posedge clk) // 每一个时钟上升沿开始触发下面的逻辑，
    begin
        // 利用counter_reset来指示当前的key_reset导致的状态
        if(key_reset == 0)
            begin
                counter_reset = 1;
            end
        else
            begin
                counter_reset = counter_reset + 1;
            end

        // 利用counter_start来指示当前的key_start_pause导致的状态
        if(key_start_pause == 0)
            begin
                counter_start = 1;
            end
        else
            begin
                counter_start = counter_start + 1;
            end

        // 利用counter_display来指示当前的key_display_stop导致的状态
        if(key_display_stop == 0)
            begin
                counter_display = 1;
            end
        else
            begin
                counter_display = counter_display + 1;
            end

        // 下面是对应于不同状态时的数码显示器的数字变化操作逻辑
        ...
    end
```

程序在时钟的每个下降沿进行一次逻辑响应，按键对应的指示变量为0时，说明该按键被触发了，应该做出相关的反应。counter_*为0的时候，说明对应按键为触发态。需要说明的是，在按钮没有触发的时候，对应的counter变量会一直增加，这种设计是为了之后的防抖动服务的。

程序的主体操作逻辑分为三个部分：

- 在一般运行情况下，每个时钟周期都检查6位数字，从最第一位可以递增，并且处理好向相邻高位进位的传递操作；
- 在key_display_stop被触发的时候，当前6位数码显示器的状态被锁定：

```
if(display == 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
```

_counter_low/high**是当前时钟时间下对应的计数数据，但是_display_low/high**才是当前显示的数码，如此就暂停了数码显示器的跳动，但是后台的真实计时时间在继续前进。

- 在key_reset被触发的时候，重置所有的6个数码显示器：

```
if(key_reset == 0)
    // 重置按钮生效了，重置所有的显示数字
    begin
        minute_display_high = 0;
        minute_display_low = 0;
        second_display_high = 0;
        second_display_low = 0;
        msecond_display_high = 0;
        msecond_display_low = 0;
        minute_counter_high = 0;
        minute_counter_low = 0;
        second_counter_high = 0;
        second_counter_low = 0;
        msecond_counter_high = 0;
        msecond_counter_low = 0;
        counter_50M = 0;
        start = 0;
        display = 0;
    end
```

利用两个指示变量说明当前是否处于计时状态和显示状态：

```
if((counter_start > DELAY_TIME)&&(key_start_pause == 0))
    // 指示当前秒表是否是运行状态，即对应的数字变量是否在增加
    begin
        start = 1 - start;
    end

if((counter_display > DELAY_TIME)&&(key_display_stop == 0))
    // 指示当前的秒表是否是数码跳动的状态
    begin
        display = 1 - display;
    end

```

如果当前处于运行状态和数码显示状态，分别进入相关的操作逻辑。相关的操作代码逻辑简单，但是比较冗长，在这里不贴出具体说明。

3. 防抖动设计：

因为时钟周期很短，导致在短时间内程序就可以做出大量的响应轮次。这就导致了按钮抖动的问题，使得程序在短时间做出多次不是预想中的按键响应。为了消除这种不利影响，我设计了一种简单但是有效的防抖动策略，基本逻辑如下：

```
parameter DELAY_TIME = 10000000; // 定义一个常量参数。 10000000 ->200ms;
```

定义一个时间参量，它对应于两次按键响应之间相间隔的最短时间。之后，不同于简单的对于**key_*=0**进行按键响应，我们使用：

```
if((counter_reset > DELAY_TIME)&&(key_reset == 0))
    // 下面的代码响应reset的按键操作逻辑

if((counter_start > DELAY_TIME)&&(key_start_pause == 0))
    // 下面的代码响应start/pause的按键操作逻辑

if((counter_display > DELAY_TIME)&&(key_display_stop == 0))
    // 下面的代码响应display/stop的按键操作逻辑
```

如前所述，在按键没有被触发的时候，我们递增**counter_***变量，这种方法就简单有效地消除了抖动带来的问题。

实验步骤

1. 在Quartes上创建新的project wizard；
2. 创建对应的.v文件，编写秒表的操作逻辑，并且把对应的module设置为porject的顶层设计，在pin planner中设置引脚；
3. 通过仿真查看当前的工程效果是否符合期望；

4. 如果仿真效果不满意，则重复2-3的操作，如果已经满意则编译工程生成.sof文件，添加到programmer中的开发板上；
5. 在开发板上操作验证，如果发现问题，则从步骤2开始循环操作。

总结和感悟

这次的大作业相较于其他课程的作业有很大的不同：

1. 是编程语言的逻辑不同，硬件上的不同线路是并发操作的，verilog类语言也没有所谓的main函数作为顺序执行的入口，这些不同都给我上手实验带来了一些困难，但是熟悉了之后，也得到了对于编程的完全不同的理解，受益颇多；
2. 不同于以往的体验，fpga的操作设计到硬件电路设计和仿真等操作，综合性更加明显，无法通过传统的工具进行debug和输入输出；
3. 不同于一般编程面对的简单计算机环境，硬件环境下的按键抖动等问题都更加具体和特殊，需要更加细致的逻辑设计和工程实现。

这次的实验，在硬件领域给我带来了全新的体验，尽管在初期上手比较困难，走了不少的弯路，但是这种体验让我对现代的电子信息科技都有了更加全面的了解，对我的帮助很大。