**数字部件秒表实验报告**

**515015910016 李嘉昊**

**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。

**三、功能简介：**

1. Key0——Reset按钮：将计时归零并暂停
2. Key1——Pause按钮：当秒表处于暂停状态时，按下pause按钮开始计时，LED屏幕上时间逐渐增长。当秒表处于计时状态时，按下pause按钮暂停计时，此时计时不变化，屏幕上显示的时间也不变化。
3. Key2——stop\_display按钮：当秒表处于计时状态时，按下stop\_display按钮，LED屏幕上的计时暂停，但是真实计时仍在增长。当秒表处于暂停显示的状态时，计时器其实仍然在增长，但是LED屏幕上显示的时间为上次按下按钮想要暂停显示时的计时，此时按下stop\_display按钮LED屏幕继续显示计时。
4. 注意：pause状态和stop\_display状态二者最多只能处于其中一种状态。

**四、过程中遇到的问题：**

1. 第一次编译时没有加入LED提示灯的变化，后为了方便调试，实现了LED提示灯和状态的同步。
2. 一开始弄反了按钮按下与松开的信号，导致不断的进入错误的if分支使pause等状态的state变量不断翻转。但是根据LED灯的不断跳动现象找出的错误并修复。
3. 初始设计中Reset之后不会暂停计时，后根据真实秒表的设计进行了改进，在执行reset后会进入pause状态。
4. 初始设计中可以同时处于pause和display\_stop两个状态，导致user按下其中一个按钮之后LED屏幕并不会发生变化，后对其进行了优化，两种状态至多只能处于其中一种。
5. 加入了防抖设计，按下按钮开始用一个局部计时器计时，当按下超过1秒时生效

**五、实验总结**

本次实验让我学会了verilog以及quantus的使用，学会了基本的语法。并在跟着教程完成了my\_first\_fpga、学习了verilog的语法的情况下完成了一个多功能秒表来通过实践加强了对与fpga的理解和verilog的编程能力。

其中有令我映像比较深刻的几个需要注意的地方。一是顶层文件名要和项目名称一致，否则编译的时候会提示相应的错误。这点也是在my\_first\_fpga教程中有提到过的。二是注意always会循环执行，不希望多次初始化的变量要放在其作用域外。三是if语句间的逻辑要清晰，代表秒表状态的寄存器设置和命名也要管理好方便之后的编程。

在代码编译通过后烧录的过程中并没有遇到太多的问题，感谢老师和助教提供的my\_first\_fpga教程，步骤和解释都十分的明确。要注意的是设置pin的时候pin对应的名称以及输入输出对应的pin千万不要搞错。

最后十分感谢老师和助教给我们提供的实验器材和环境，作为软件工程的学生能有机会接触此类与硬件相关的实验也非常有趣，而且在实验过程中我也体会到了实践才能出真知，输入信号的0/1、防抖以及pin的设置这些细节在自己动手实践后确实会有更深入的理解，

**六、时钟代码实现：**

module clock\_main(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 bool\_reset; // 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 bool\_start\_pause; //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 bool\_display\_stop; //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 );

/\*My Implementation\*/

reg bool\_do\_reset;//是否执行reset

reg bool\_do\_pause;//是否暂停

reg bool\_do\_display=1;//是否显示

//declare new reg

reg [31:0] per\_time\_counter;//最小单位时间计数器

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

//此处功能代码省略，由同学自行设计。

//按键判断：

//case1-1：按下reset,重置计时并暂停

if(!key\_reset && !bool\_reset)

begin

counter\_reset=counter\_reset+1;//按下过0.1s后生效

if(counter\_reset >= 500000)

begin

bool\_reset=~bool\_reset;//reverse the reset state

counter\_reset=0;//set the count to 0

bool\_do\_reset=1;//set the do\_reset to 1

bool\_do\_pause=0;

bool\_start\_pause=0;

end

end

//case1-2:松开reset

/\*

else if(!key\_reset && bool\_reset)

begin

counter\_reset=counter\_reset+1;//按下过0.1s后生效

if(counter\_reset >= 500000)

begin

bool\_reset=~bool\_reset;//reverse the reset state

counter\_reset=0;//set the count to 0

end

end

\*/

//case1-3:else

else

begin

counter\_reset=0;//clear the reset counter

end

//case2-1:按下start\_pause，尝试关闭暂停

if(!key\_start\_pause && !bool\_start\_pause)

begin

counter\_start=counter\_start+1;

if(counter\_start >= 5000000)

begin

bool\_start\_pause=~bool\_start\_pause;

counter\_start=0;

bool\_do\_pause=1;

bool\_display\_stop=1;

bool\_do\_display=1;

end

end

//case2-2：按下start\_pause，尝试启用暂停

else if(!key\_start\_pause && bool\_start\_pause)

begin

counter\_start=counter\_start+1;

if(counter\_start >= 5000000)

begin

bool\_start\_pause=~bool\_start\_pause;

counter\_start=0;

bool\_do\_pause=0;

end

end

//case2-3:else

else

begin

counter\_start=0;//clear the reset counter

end

//case3-1:按下display\_stop，尝试开启显示

if(!key\_display\_stop && !bool\_display\_stop)

begin

counter\_display=counter\_display+1;

if(counter\_display >= 5000000)

begin

bool\_display\_stop=~bool\_display\_stop;

counter\_display=0;

bool\_do\_display=1;

bool\_do\_pause=1;

bool\_start\_pause=1;

end

end

//case3-2:按下display\_stop，尝试暂停显示，但计时仍在继续

else if(!key\_display\_stop && bool\_display\_stop)

begin

counter\_display=counter\_display+1;

if(counter\_display >= 5000000)

begin

bool\_display\_stop=~bool\_display\_stop;

counter\_display=0;

bool\_do\_display=0;

end

end

//case3-3:else

else

begin

counter\_display=0;//clear the reset counter

end

//实现LED发光指示灯:led0=reset/led1=start\_pause/led2=display\_stop

if(bool\_do\_reset)

begin

led0=1;

end

else

begin

led0=0;

end

if(bool\_do\_pause)

begin

led1=1;

end

else

begin

led1=0;

end

if(bool\_do\_display)

begin

led2=1;

end

else

begin

led2=0;

end

//judge display&reset

//do reset

if(bool\_do\_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;

//finish reset,set do\_reset to 0

bool\_reset=0;

bool\_do\_reset=0;

end

//do\_display

if(bool\_do\_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

//update the clock counter

if(bool\_do\_pause)

begin

per\_time\_counter=per\_time\_counter+1;

if(per\_time\_counter==500000)

begin

msecond\_counter\_low=msecond\_counter\_low+1;//msecond\_low+1

per\_time\_counter=0;//reset the per\_time\_counter

//whether a carry happen on the msecond\_counter\_high

if(msecond\_counter\_low==10)

begin

msecond\_counter\_high=msecond\_counter\_high+1;

msecond\_counter\_low=0;

//whether a carry happen on the second\_counter\_low

if(msecond\_counter\_high==10)

begin

second\_counter\_low=second\_counter\_low+1;

msecond\_counter\_high=0;

//whether a carry happen on the second\_counter\_high

if(second\_counter\_low==10)

begin

second\_counter\_high=second\_counter\_high+1;

second\_counter\_low=0;

//whether a carry happen on the minute\_counter\_low

if(second\_counter\_high==6)

begin

minute\_counter\_low=minute\_counter\_low+1;

second\_counter\_high=0;

//whether a carry happen on the minute\_counter\_high

if(minute\_counter\_low==10)

begin

minute\_counter\_high=minute\_counter\_high+1;

minute\_counter\_low=0;

//whether a carry happen on the over\_minute\_counter\_high

if(minute\_counter\_high==10)

begin

minute\_counter\_high=10;

end

end

end

end

end

end

end

end

end

Endmodule