**实验四 综合电路设计实验报告**

**组长**：张驰  **学号**：1120191600

**班级：**07111904班  **手机：**18810575675

**组员**：魏慧聪  **学号**：1120191866

**班级**：07111904班 **手机**：15733906832

**组员**：徐幸波 **学号**：1120191232

**班级**：07111903班 **手机**：19967471965

1. **实验题目**

短跑计时器设计与实现（难度系数：0.9）

短跑计时器描述如下：

◼ 短跑计时器显示分、秒、毫秒；

◼ “毫秒”用两位数码管显示：百位、十位；

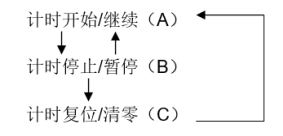
◼ “秒”用两位数码管显示：十位、个位；

◼ “分”用一位 LED 灯显示，LED 灯“亮”为 1 分；

◼ 最大计时为 1 分 59 秒 99，超限值时应可视或可闻报警；

◼ 三个按键开关：计时开始/继续（A）、计时停止/暂停（B）、复位/清零（C），

键控流程如下：



1. **电路设计**

**电路输入：**

时钟信号clk

计时开始/继续信号start\_key

计时停止/暂停信号pause\_key

计时复位/清零信号reset\_key

**电路输出：**

数码管的八位信号（包含七段数码管和一个小数点的LED灯）sig\_seg[7:0]

数码管的段选信号 CS[3:0]

一个LED灯输出 light

蜂鸣器超时警告输出 alert

**电路的概要设计：**

xc7a100tcsg324-1的内置系统时钟搭载一个100MHz的时钟信号，电路要求显示的是（十秒位）（秒位）.（百毫秒位）（十毫秒位）。

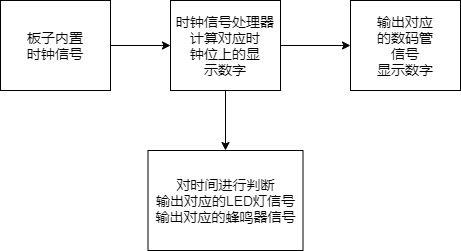
对于一个100MHz的时钟信号，clk跳动100000次为1ms，按下开始键，随着clk跳动100000次，毫秒位加一，当毫秒位到10时，进行进位，十毫秒位加一。当十毫秒位到10，进行进位，百毫秒位加一。当百毫秒位到10，进行进位，秒位加一。当秒位到10，进行进位，十秒位加一。

当十秒位到6，计时超过一分钟，这个时候要恢复十秒位、秒位、百毫秒位、十毫秒位为0，记录已经超过1分钟，变量min<=2’b01，让一盏led灯亮起，再次进行计时。

在进行计时过程中，按下暂停键，毫秒位不再变动，因此，数码管的各位显示值也不再变动，即停止计时。再次按下暂停键，毫秒位继续随着内置系统时钟信号clk变动。在进行计时过程中，按下复位键，此时所有显示数字全部置零，之后不再变动，当再次按下开始键，开始信号置为1，毫秒位再次随着clk信号而变动。

当计时达到两分钟，这时蜂鸣器响起，蜂鸣器的每隔50618个时钟信号（频率为低音7）响起一次，在板上的蜂鸣器表现为持续响起低音7的声音进行报警。当按下复位键，蜂鸣器停止蜂鸣。

**电路的模块图**：



1. **电路实现**

该部分通过Verilog语言对电路设计进行实现。关键模块的Verilog代码和注释如下所示：

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2019/05/06 09:12:15

// Design Name:

// Module Name: time\_counter

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module time\_counter(clk, pause\_key, reset\_key, start\_key, sig\_seg, CS, light, alert);

    input clk;              //板子上内置时钟，100MHz

    input pause\_key;        //暂停按键

    input reset\_key;        //重置按键

    input start\_key;        //开始计时按键

    output [7:0] sig\_seg;   //表示数码管

    output [3:0] CS;        //段选信号

    output light;       //表示分钟的LED灯

    output alert;

    reg min =2'b00;         //分钟

    reg started = 1'b0;     //表示当前状态，1表示正常计时，0表示暂停

    reg temp = 4'd1;        //temp与started相关，1和0表示另外一个时钟信号的增加或暂停状态

    reg [7:0] sig\_seg = 8'b0000\_0000;       //数码管

    reg [3:0] CS = 4'b1000;     //段选信号，1000 表示选择第一个数字

    reg light = 1'b0;

    reg alert = 1'b0;

    reg [16:0] ctr1 = 17'd0;

    reg [16:0] ctr0 = 17'd0;    //自定义时钟变量，每1ms循环一次

    reg [1:0] ctr2 = 2'd0;

    reg [3:0] ms = 4'd0;        //毫秒位

    reg [3:0] tms = 4'd0;       //十毫秒位

    reg [3:0] hms = 4'd0;       //百毫秒位

    reg [3:0] s = 4'd0;         //秒位

    reg [3:0] ts = 4'd0;        //十秒位

    reg [16:0] count\_end = 17'd50618; //蜂鸣器的频率 表示低音7

    reg [16:0] ctr\_beep = 17'd0;//蜂鸣器计数器

    always@ (posedge clk or negedge reset\_key or posedge start\_key or posedge pause\_key) begin      //clk上升沿、重置键下降沿、开始键上升沿和暂停键上升沿开始执行操作

        ctr\_beep <= ctr\_beep + 1'b1;

        //蜂鸣器按照一定的频率进行蜂鸣

        if(ctr\_beep == count\_end)begin

            ctr\_beep <= 17'd0;

            if(temp == 4'd0)begin

                alert <= !alert;

            end

        end

        if(min == 2'b01) begin      //如果分钟为1，则亮起LED灯光

            light <= 1;

        end

        if(!reset\_key) begin        //如果按下重置按键，则清空现有所有与时钟显示有关的寄存器变量

            started <= 1'b0;        //进入暂停状态，其余数字归零，按下开始键重新计时

            ctr1 <= 17'd0;

            ctr2 <= 2'd0;

            ms <= 4'd0;

            tms <= 4'd0;

            hms <= 4'd0;

            s <= 4'd0;

            ts <= 4'd0;

            min <= 2'b00;

            light <= 0;

            temp <= 4'd1;

            alert <= 0;

        end

        else begin

            if(start\_key) begin         //如果按下开始键，表示正常计时

                started <= 1'b1;

            end

            if(pause\_key) begin         //如果按下暂停按键，started寄存器变量置为0，表示暂停状态

                started <= 1'b0;

            end

            if(ctr0 == 17'd10\_0000) begin       //ctr0随着系统时钟变化，每跃变10\_0000次代表1ms

                ctr0 <= 17'd1;

                if(ctr2 != 2'd3)

                    ctr2 <= ctr2 + 2'd1;

                else

                    ctr2 <= 2'd0;

            end

            else begin

                ctr0 <= ctr0 + 1;

            end

            //==================

            if(ctr1 == 17'd10\_0000 && started == 1'b1 ) begin          //ctr1每跃变10\_0000次，代表过1ms，ms位加1

                ctr1 <= 17'd1;                                         //其实ctr1与ctr0代表的意义相同，只不过ctr0控制段选，不随着started变化

                ms <= ms + temp;

            end

            else if (ctr1 !=17'd10\_0000 && started == 1'b1 ) begin

                ctr1 <= ctr1 + 1;

            end

            //==================

            if(ms == 4'd10 && ctr1 == 17'd10\_0000 && started == 1'b1 ) begin       //计时状态且ms达到10，则十毫秒位进1

                ms <= 4'd0;

                tms <= tms + 4'd1;

            end

            if(tms == 4'd10 && ctr1 == 17'd10\_0000 && started == 1'b1 ) begin       //计时状态且tms达到10，则百毫秒位进1

                tms <= 4'd0;

                hms <= hms + 4'd1;

            end

            if(hms == 4'd10 && ctr1 == 17'd10\_0000 && started == 1'b1 ) begin       //计时状态且hms达到10，则秒位进1

                hms <= 4'd0;

                s <= s + 4'd1;

            end

            if(s == 4'd10 && ctr1 == 17'd10\_0000 && started == 1'b1 ) begin         //计时状态且s达到10，则十秒位进1

                s <= 4'd0;

                ts <= ts + 4'd1;

            end

            //计时状态且十秒位达到6，则其余位置0.分位进1

            if(ts == 4'd6 && ctr1 == 17'd10\_0000 && started == 1'b1 && min == 2'b00) begin

                min <= 2'b01;

                ms <= 4'd0;

                tms <= 4'd0;

                hms <= 4'd0;

                s <= 4'd0;

                ts <= 4'd0;

            end

            //计时状态且十秒位达到6，分位达到1，即处于两分钟位置

            if(ts == 4'd6 && ctr1 == 17'd10\_0000 && started == 1'b1 && min == 2'b01) begin

                temp <= 4'd0;

            end

            //==============================

            case (ctr2)     //ctr2代表段选信号，0，1，2，3分别表示选择一段数码管

            2'd0:begin      //ctr2=0时，选择第1段

                CS <= 4'b1000;

                case(ts)

                //////////////gfedcba////////////////////////////////////////////sig\_seg最后一位表示小数点

                    4'd0:sig\_seg <= 8'b0111\_1110;

                    4'd1:sig\_seg <= 8'b0000\_1100;//             a

                    4'd2:sig\_seg <= 8'b1011\_0110;//            \_\_

                    4'd3:sig\_seg <= 8'b1001\_1110;//         f/   /b

                    4'd4:sig\_seg <= 8'b1100\_1100;//           g

                    4'd5:sig\_seg <= 8'b1101\_1010;//          \_\_

                    4'd6:sig\_seg <= 8'b1111\_1010;//       e /   /c

                    4'd7:sig\_seg <= 8'b0000\_1110;//          \_\_

                    4'd8:sig\_seg <= 8'b1111\_1110;//           d

                    4'd9:sig\_seg <= 8'b1101\_1110;

                    default:;

                endcase

            end

            2'd1:begin      //ctr2=1时，选择第2段,

                CS <= 4'b0100;

                case(s)

                    4'd0:sig\_seg <= 8'b0111\_1111;

                    4'd1:sig\_seg <= 8'b0000\_1101;

                    4'd2:sig\_seg <= 8'b1011\_0111;

                    4'd3:sig\_seg <= 8'b1001\_1111;

                    4'd4:sig\_seg <= 8'b1100\_1101;

                    4'd5:sig\_seg <= 8'b1101\_1011;

                    4'd6:sig\_seg <= 8'b1111\_1011;

                    4'd7:sig\_seg <= 8'b0000\_1111;

                    4'd8:sig\_seg <= 8'b1111\_1111;

                    4'd9:sig\_seg <= 8'b1101\_1111;

                    default:;

                endcase

            end

            2'd2:begin      //ctr2=2时，选择第3段

                CS <= 4'b0010;

                case(hms)

                    4'd0:sig\_seg <= 8'b0111\_1110;

                    4'd1:sig\_seg <= 8'b0000\_1100;

                    4'd2:sig\_seg <= 8'b1011\_0110;

                    4'd3:sig\_seg <= 8'b1001\_1110;

                    4'd4:sig\_seg <= 8'b1100\_1100;

                    4'd5:sig\_seg <= 8'b1101\_1010;

                    4'd6:sig\_seg <= 8'b1111\_1010;

                    4'd7:sig\_seg <= 8'b0000\_1110;

                    4'd8:sig\_seg <= 8'b1111\_1110;

                    4'd9:sig\_seg <= 8'b1101\_1110;

                    default:;

                endcase

            end

            2'd3:begin      //ctr2=3时，选择第4段

                CS <= 4'b0001;

                case(tms)

                    4'd0:sig\_seg <= 8'b0111\_1110;

                    4'd1:sig\_seg <= 8'b0000\_1100;

                    4'd2:sig\_seg <= 8'b1011\_0110;

                    4'd3:sig\_seg <= 8'b1001\_1110;

                    4'd4:sig\_seg <= 8'b1100\_1100;

                    4'd5:sig\_seg <= 8'b1101\_1010;

                    4'd6:sig\_seg <= 8'b1111\_1010;

                    4'd7:sig\_seg <= 8'b0000\_1110;

                    4'd8:sig\_seg <= 8'b1111\_1110;

                    4'd9:sig\_seg <= 8'b1101\_1110;

                    default:;

                endcase

            end

            default:;

            endcase

        end

    end

endmodule

1. **电路验证**
   1. **TestBench**

该部分通过编写Verilog TestBench，来对实现的电路进行功能验证。Verilog代码和相关注释如下所示：

由于电脑空间内存不足，不能够运行仿真150s的波形图的结果，所以以下代码只能对慢跑计时器进行部分的代码仿真，具体仿真详见电路的上板视频。

`timescale 1ns / 1ps

module testbench ();

    reg clk\_100M = 1'b0;

    reg reset = 1'b1;

    reg pause = 1'b0;

    reg start = 1'b0;

    wire [7:0] sig;

    wire [3:0] cs;

    wire light\_60s;

    wire alert\_120s;

    always #1 clk\_100M <= ~clk\_100M;

    initial begin

        //初始状态，全按钮弹起

        #5000

        start <= 1'b1;          //按下开始按钮

        #100

        start <= 1'b0;          //松开开始按钮

        #5000

        pause <= 1'b1;          //按下暂停按钮

        #100

        pause <= 1'b0;          //松开暂停按钮

        #5000

        start <= 1'b1;          //按下继续按钮

        #100

        start <= 1'b0;          //松开继续按钮

        #5000

        pause <= 1'b1;          //按下暂停按钮

        #100

        pause <= 1'b0;          //松开暂停按钮

        #5000

        reset <= 1'b0;          //按下重置按钮

        #100

        reset <= 1'b1;          //松开重置按钮

        #5000

        start <= 1'b1;          //按下开始按钮

        #100

        start <= 1'b0;          //松开开始按钮

        #5000

        pause <= 1'b1;          //按下暂停按钮

        #100

        pause <= 1'b0;          //松开暂停按钮

        #5000

        reset <= 1'b0;          //按下重置按钮

        #100

        reset <= 1'b1;          //松开重置按钮

    end

    time\_counter TC(

        .clk( clk\_100M ),

        .pause\_key( pause ),

        .reset\_key( reset ),

        .start\_key( start ),

        .sig\_seg( sig ),

        .CS( cs ),

        .light( light\_60s ),

        .alert( alert\_120s )

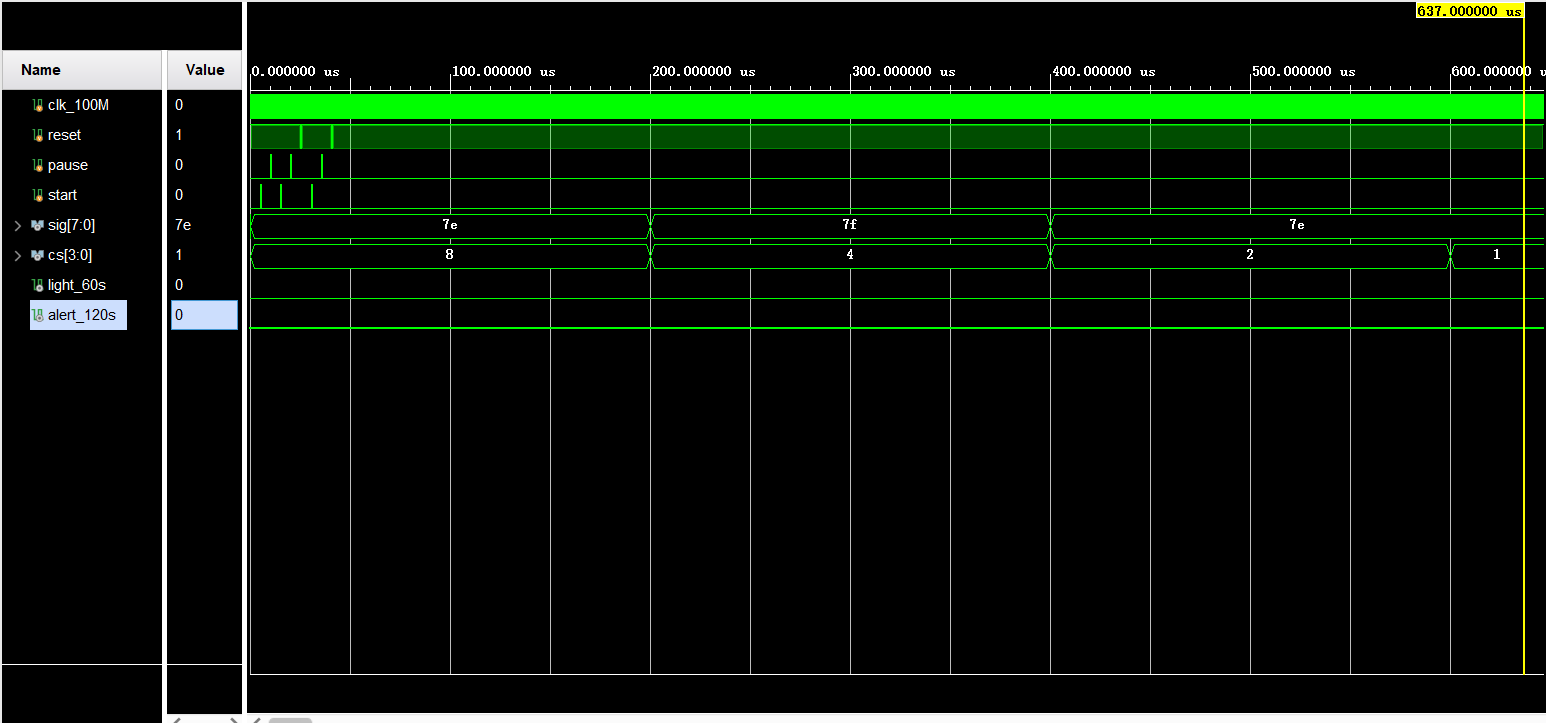
    );

endmodule

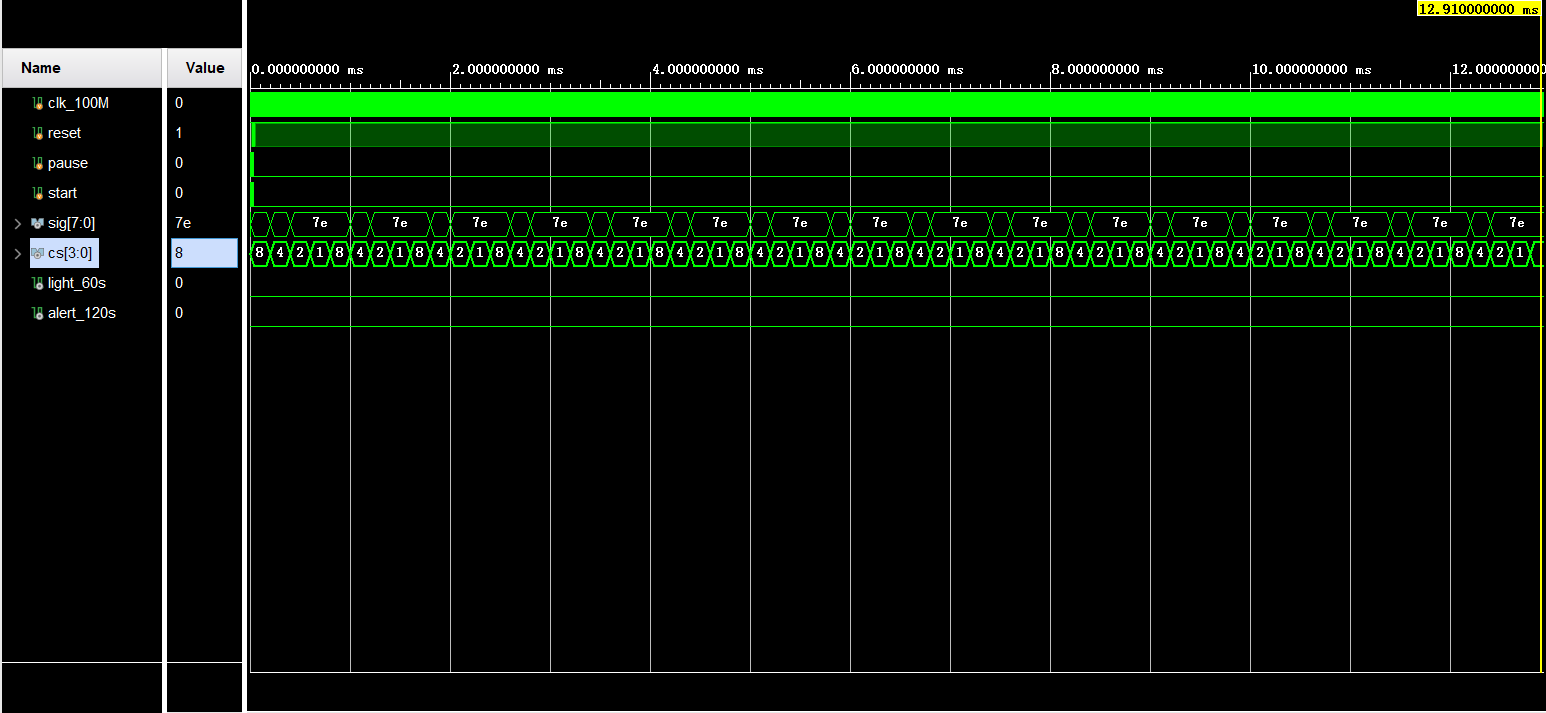
* 1. **仿真结果**

该部分通过Vivado仿真，得到仿真得到的Wave。波形图如下所示：

由于电脑空间内存不足，不能够运行仿真150s的波形图的结果，所以以下波形图仅展示电路的部分功能，具体功能详见上板视频。



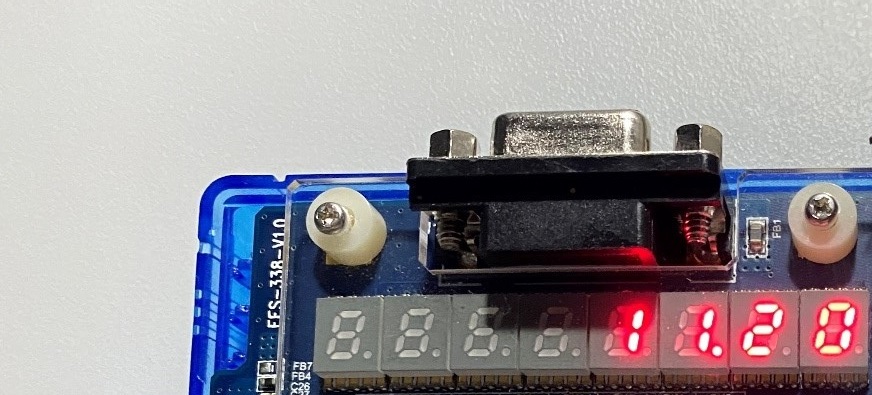
在本图中，我们可以看出reset、start、pause按钮在相应的时间都进行了对应的要求操作，由于电路仿真进行的时间较短（及每一块都显示的是0），所以对应的reset、start、pause操作不能在波形图上明显看出。但是，对应时间的复位、开始、暂停操作都有其对应的效果。



在本图中，我们可以看出，段选部分从8变为4变为2变为1，即二进制的变化为1000、0100、0010、0001，对应的段选即为第一个数字、第二个数字、第三个数字、第四个数字。

在波形图中，我们发现8、4、2、1对应的sig[7:0]中，8、2、1处对应显示的值为7e（二进制为0111 1110），对应在数码管上显示的数字为0（不带小数点），显示无误。在CS[3:0]的值为4时，二进制为0100，第二位数字显示时间时会带上小数点，显示的值为7f（二进制为0111 1111），对应在数码管上显示的数字为0（带小数点），显示无误。

可以看出，数码管的显示在模拟仿真时无误。

（如图，从左到右分别为 8 4 2 1对应的显示的数字）

1. **电路上板**
   * 1. 管脚配置代码：

set\_property IOSTANDARD LVCMOS33 [get\_ports{CS[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{CS[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{CS[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{CS[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{sig\_seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{clk}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{pause\_key}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{reset\_key}]

set\_property IOSTANDARD LVCMOS33 [get\_ports{start\_key}]

set\_property IOSTANDARD LVCMOS33 [get\_ports light]

set\_property IOSTANDARD LVCMOS33 [get\_ports alert]

set\_property PACKAGE\_PIN D5 [get\_ports {sig\_seg[0]}]

set\_property PACKAGE\_PIN B4 [get\_ports {sig\_seg[1]}]

set\_property PACKAGE\_PIN A4 [get\_ports {sig\_seg[2]}]

set\_property PACKAGE\_PIN A3 [get\_ports {sig\_seg[3]}]

set\_property PACKAGE\_PIN B1 [get\_ports {sig\_seg[4]}]

set\_property PACKAGE\_PIN A1 [get\_ports {sig\_seg[5]}]

set\_property PACKAGE\_PIN B3 [get\_ports {sig\_seg[6]}]

set\_property PACKAGE\_PIN B2 [get\_ports {sig\_seg[7]}]

set\_property PACKAGE\_PIN H1 [get\_ports {CS[3]}]

set\_property PACKAGE\_PIN C1 [get\_ports {CS[2]}]

set\_property PACKAGE\_PIN C2 [get\_ports {CS[1]}]

set\_property PACKAGE\_PIN G2 [get\_ports {CS[0]}]

set\_property PACKAGE\_PIN T5 [get\_ports clk]

set\_property PACKAGE\_PIN R15 [get\_ports pause\_key]

set\_property PACKAGE\_PIN P15 [get\_ports reset\_key]

set\_property PACKAGE\_PIN R17 [get\_ports start\_key]

set\_property PACKAGE\_PIN K2 [get\_ports light]

set\_property PACKAGE\_PIN G13 [get\_ports alert]

数码管HABCDEFG的管脚对应sig\_seg[0] — sig\_seg[7]

数码管的段选信号的管脚对应为 CS[3] — CS[0]

时钟信号管脚对应 clk

开始键的管脚对应的是板子中五个方向键的上键，信号为start\_key

暂停键的管脚对应的是板子中五个方向键的中心键，信号为pause\_key

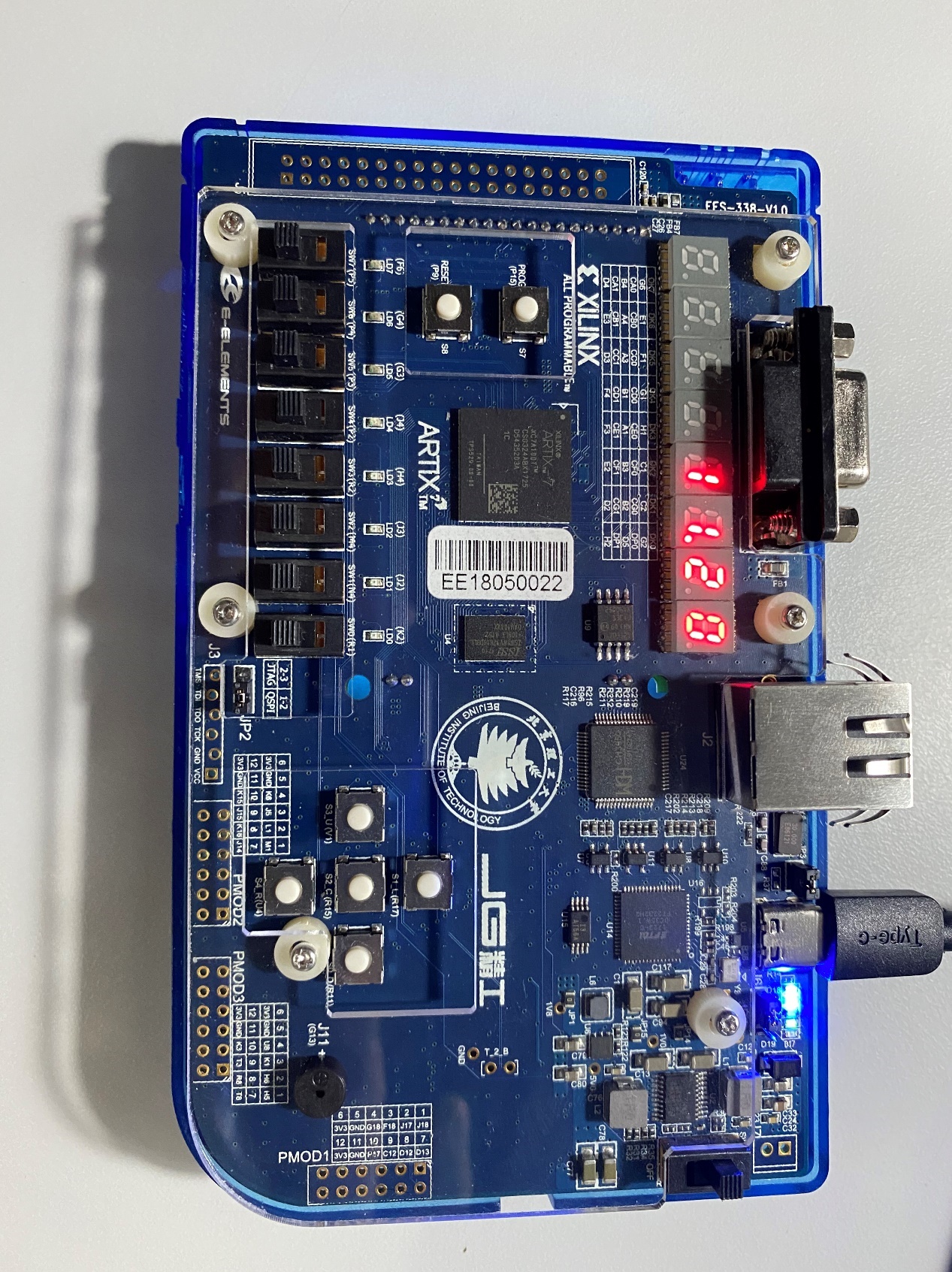
复位键的管脚对应的是板子中的RESET部位，信号为reset\_key

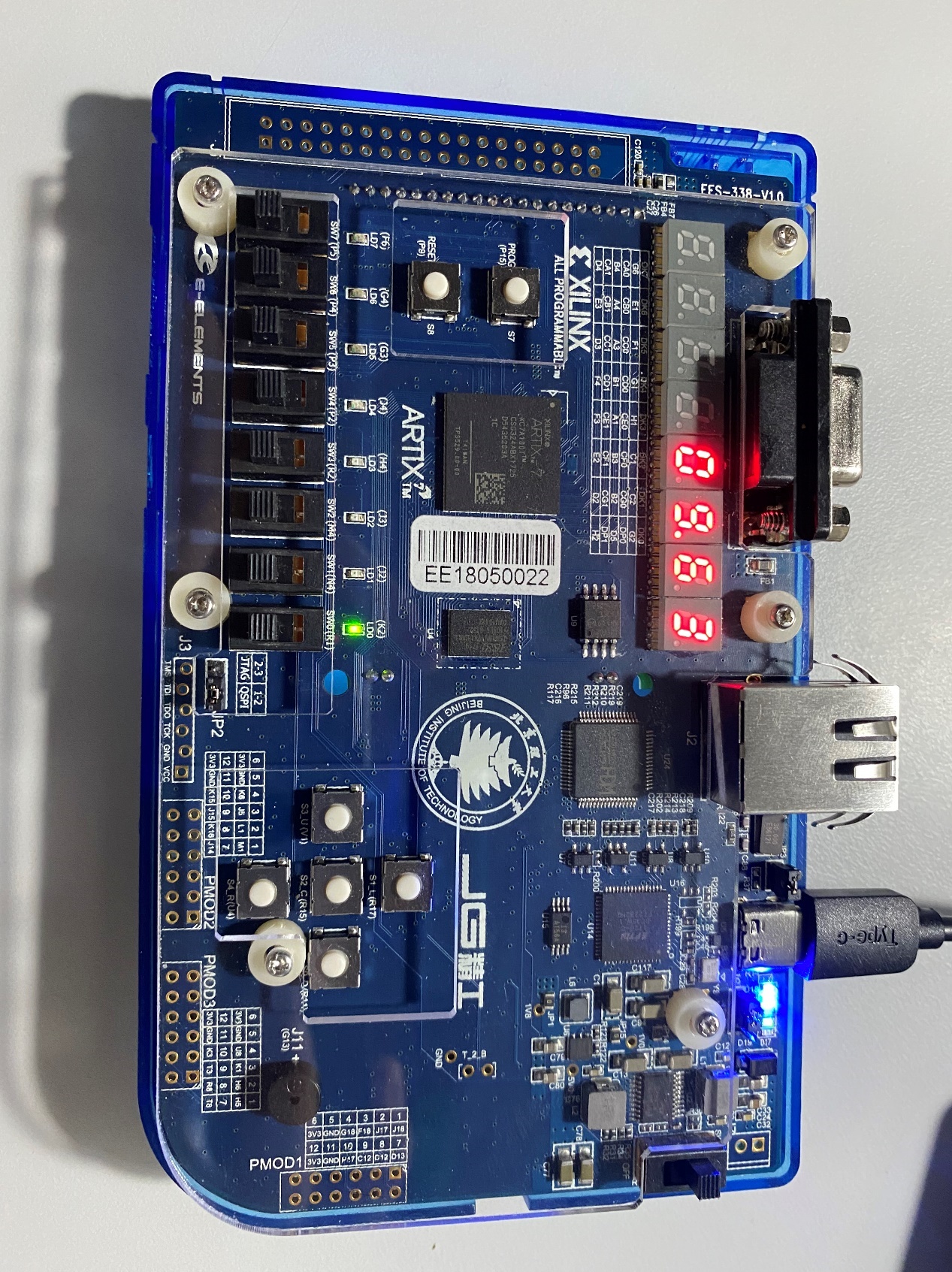
LED灯的信号的管脚对应的是八个LED灯最右侧的LED灯，信号为light

蜂鸣器管脚对应的信号为alert

* + 1. 电路上板情况：

上板成功图片如下，上板成功视频见附件所示。





1. **实验心得**

这次的数字逻辑实验四，让我们组了解到了更多的知识，加深了对数字逻辑课程的理解。本次的上板实验，使我们组的各个成员将知识与实践相结合，制作出了一个慢跑计时器的实物。本次的计时装置，使我们接触到了更新的知识，不仅仅是Verilog语言变成技术，而且还有上板、按键去抖、设置管脚等更新的知识。

在本次实验中，我们组遇到了如下的问题：

（1）管脚配置的电平异常导致无法生成比特流文件(bitstream)

（2）蜂鸣器写好之后没有响起

（3）数码管显示在暂停时出现异常

（4）开始按钮按下时不开始计时

我们组解决以上问题的办法：

（1）打开vivado软件中的Windows的I/O parts 界面，将配置电平由LVCMOS18改为LVCMOS33；

（2）这是由于蜂鸣器在clk变化的每一次都在持续响起，导致频率过高，成为超声波，超越人耳听力的范畴，导致蜂鸣器在人耳中听起来看似未响起。解决问题的措施是加上一个时间，使得蜂鸣器的频率调整至人耳听力的范畴内，调整声音的频率，解决了蜂鸣器声音超过人耳范畴的问题；

（3）暂停出现异常，是由于设置暂停时，数码管的显示的触发信号仍然不能暂停，否则会出现显示的异常。解决方法为多添加一个时钟信号，对数目管的显示进行单独的调整，令其不随计时的毫秒信号的暂停的影响。

（4）按钮按下后不计时，是因为在代码中如果超过一分钟，会一直将start信号设置成0，导致计时无法继续进行，将代码改正即可。

在实验的过程中，我们要培养自己的独立分析问题和解决问题的能力。同时，团队的配合能力也在本次大实验的过程中得到了极大的锻炼。数字逻辑的实验，有利于培养我们的创新钻研的能力，有利于书本知识技能的巩固与迁移。

在此过程中，我们通过查找大量的资料，学习书籍，请教老师以及不懈的努力，在各种能力上都有了提高。在实验课上，我们学习了很多地方法，真的是受益匪浅。我们在学习知识的路上，只有不断的学习、实践，才能巩固我们学到的知识。