一、实验内容

蓝牙控制移动接物游戏数字系统

1. 概况

基于 FPGA、蓝牙从模块、声音传感器、vga 显示屏的接随机掉落物体计分小游戏。

2. 所用器件

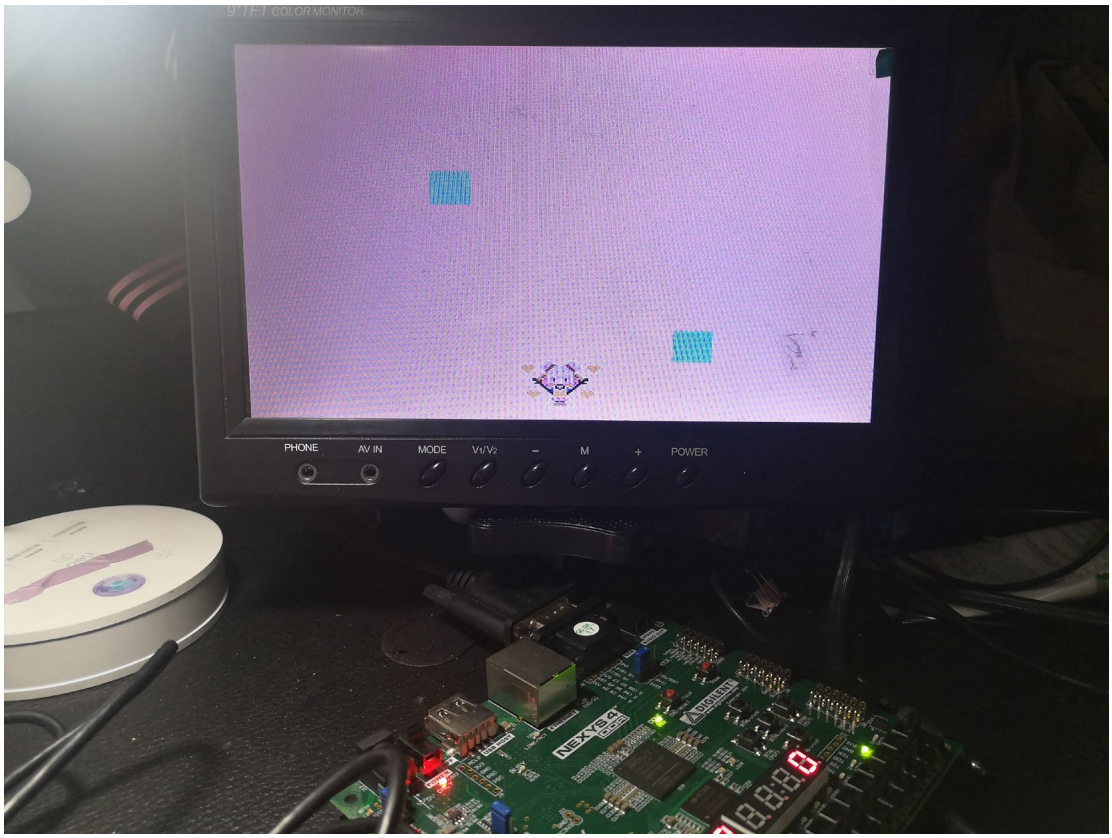
Nexys 4 DDR Artix-7——由 Xilinx 公司开发出的一款现场可编程门阵列（FPGA）开发板。

VGA——VGA(Video Graphics Array)是 IBM 在 1987 年随 PS/2 机一起推出的一种 视频传输标准，具有分辨率高、显示速率快、颜色丰富等优点；

Bluetooth Slave UART Board——基于 URART 接口的蓝牙从模块，应用于无线蓝牙数据传输。

LM386——LM386 是一种音频集成功率放大器，具有自身功耗低、更新内链增益可调整、电源电压范围大、外接元件少和总谐波失真小等优点。主要应用于 低电压消费类产品。

3. 界面



4. 操作说明

打开板子上的 J15 开关开机，进入开始界面，此时七段数码管显示作者学号。点击

手机上的蓝牙按键“开始”进入游戏。

游戏过程中通过手机控制画面小人左右移动接住下落物体，若接中则板子上的三色led灯会闪同物体的颜色，未接中则会闪红灯；每轮游戏共10个物体，板子七段数码管左侧显示错过的物体数，右侧显示接中物体数；每局基础生命值为3，若错过三个以上的物体则游戏失败，进入充能界面通过声音传感器恢复生命值，只有将生命值充至3才能开启下一局游戏。

若游戏成功结束则可按“开始”开启下一局游戏，或按“退出”回到开机界面。

二、蓝牙控制移动游戏数字系统总框图

各模块共用一个顶层控制器：

（1） 图片存储调用了IP核用于存储游戏素材。

（2） 时钟分频模块块调用了 IP 核，用于将板内时钟分频；但似乎clocking wizard有分频范围的限制，我又根据需要手写了分频器。

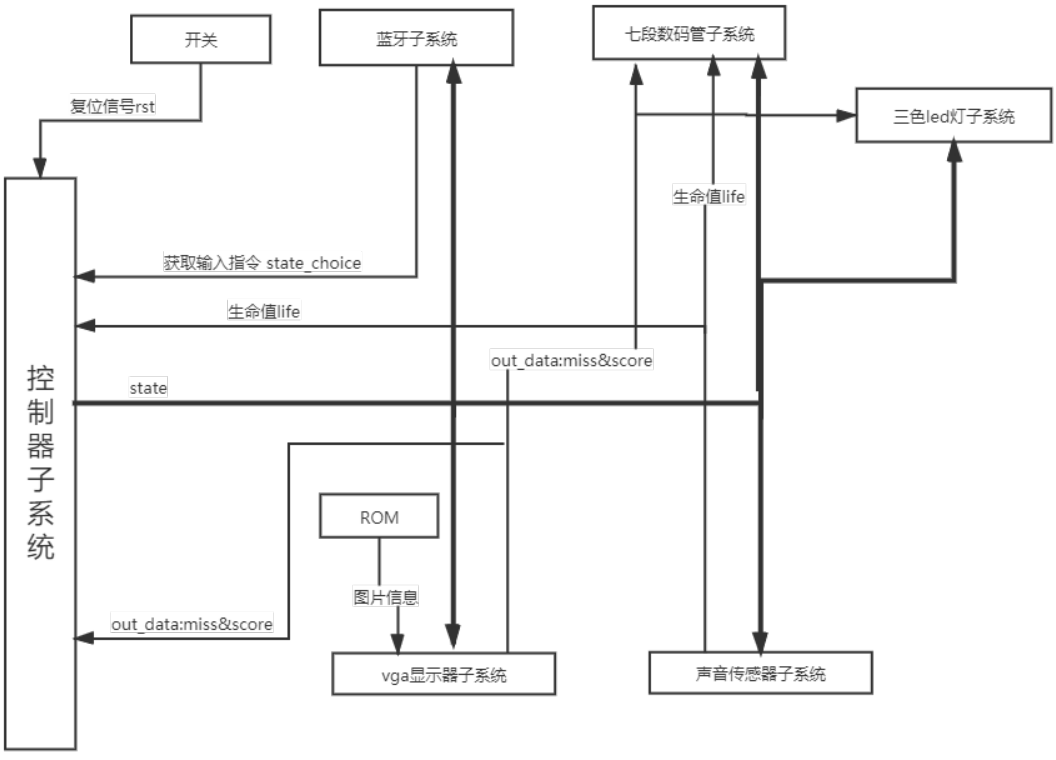
（3） 蓝牙模块实现与手机的交互，获取手机端发送的指令，从而达到游戏操作的效果。

（4） VGA显示模块与显示屏交互，控制图片的显示和不同页面的切换。

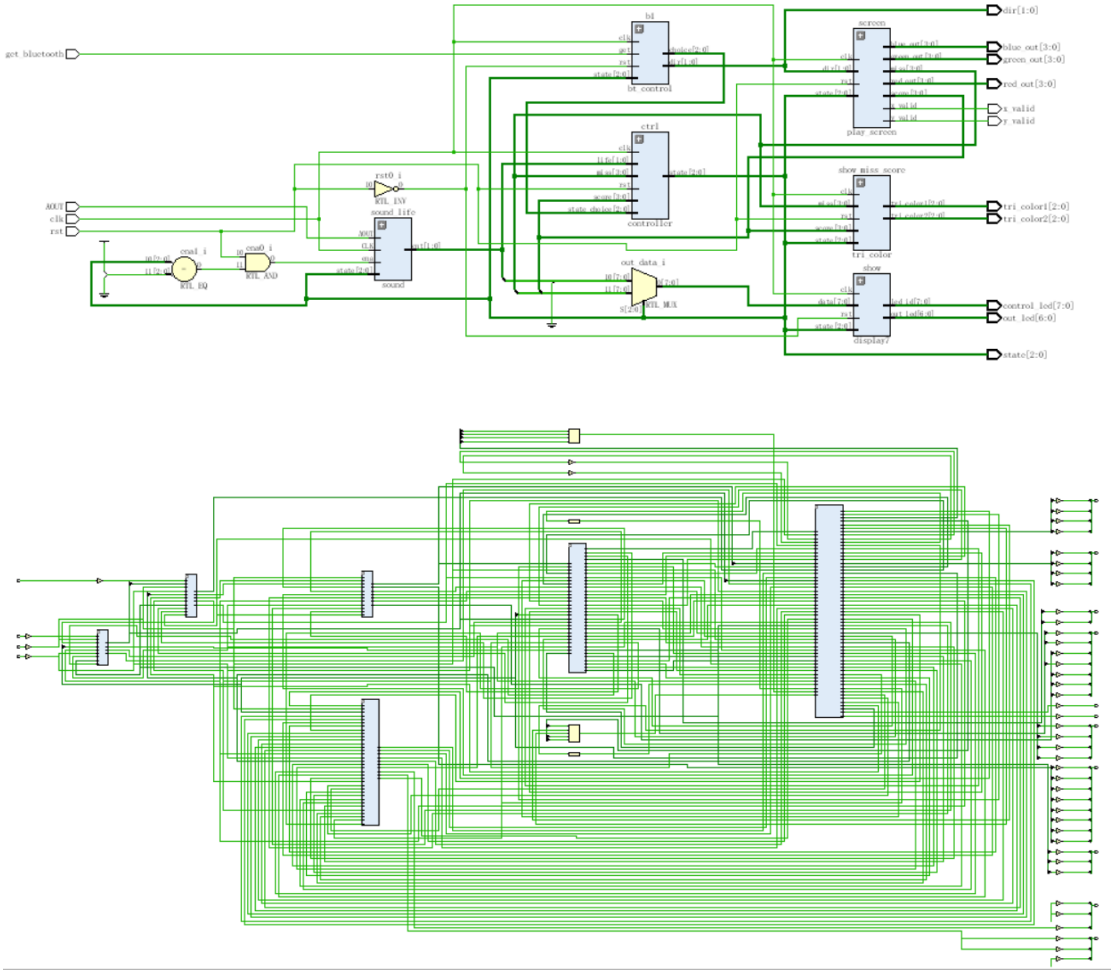
（5） 七段数码管：初始状态下显示作者学号，在游戏过程中根据游戏进程显示错失值、得分、以及生命值的十进制值。

（6） 三色led灯：当游戏过程中小人接住下落物体时LD16闪物体同色的蓝绿光，而错失物体时闪红光。

（7） 声音传感器：当错失第4个物体时，声音传感器使能信号有效，感知周围的声音达到一定强度且持续一段时间后，增加一生命值，计数生命值达到3时停止。

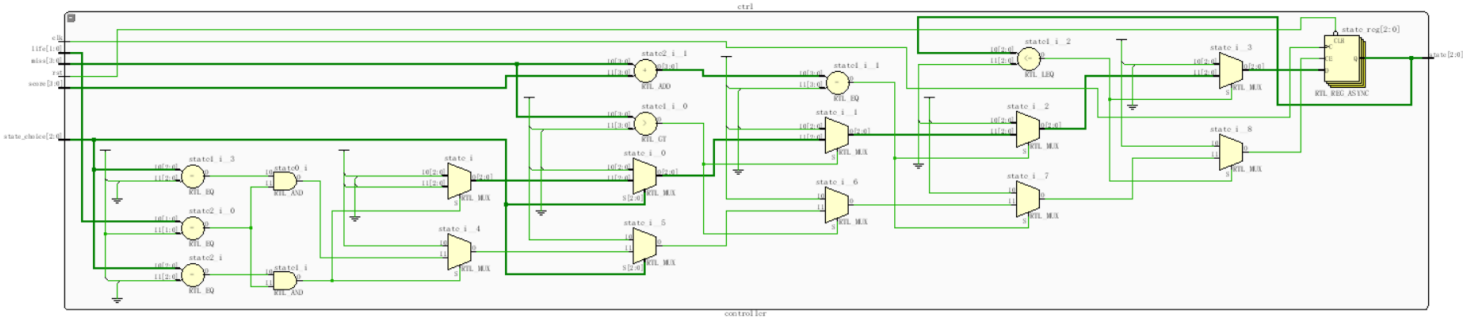


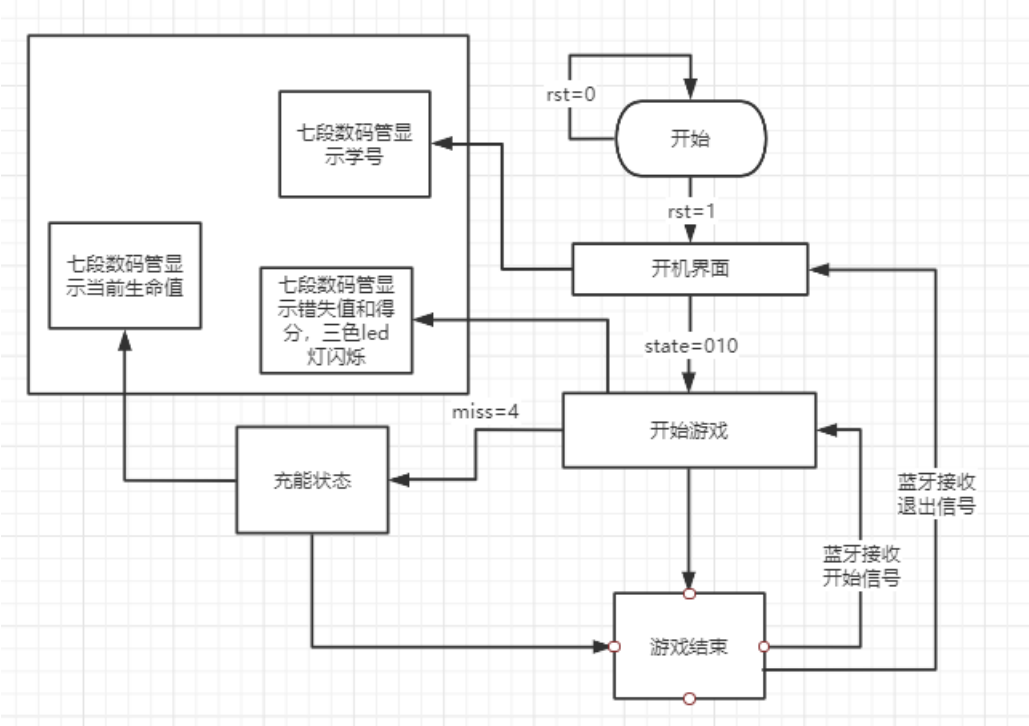
RTL 原理图



三、系统控制器设计

RTL 级设计图：





现态	转移条件	次态
000（开始状态，复位状态）	rst=1	001
	rst=0	000
001(开机状态，显示初始界面)	蓝牙读入开始指令	010
	蓝牙读入退出指令	101
010（游戏状态）	错失值达到 4	110
	掉落物体达到 10	100
100（游戏结束状态）	蓝牙读入开始指令	010
	蓝牙读入退出指令	101
101（游戏退出状态，回到初始界面）	蓝牙读入开始指令	010
110（生命值充能状态）	生命值达到 3	100

四、子系统模块建模

（1） 顶层模块 top

用来统一协调各子系统，提供与外设间的接口，方便各模块之间的数据传输和与外设的交互。

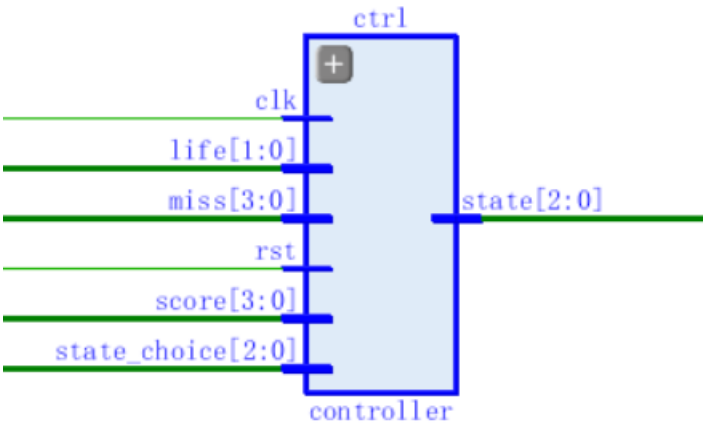
```
module top(
    input clk,
    input rst,
    input get_bluetooth,//蓝牙接口
    input AOUT,//声传
    output x_valid,//行时序信号
    output y_valid,//场时序信号
    output [3:0]  red_out,green_out,blue_out,//rgb 像素信息
```

```
    output [1:0]dir,    //位移
    output [7:0] control_led,//七段数码管控位
    output [6:0] out_led, //led 显示
    output [2:0] tri_color1,//三色灯 1
    output [2:0] tri_color2,//三色灯 2
    output [2:0]state
    );
    //wire [2:0]state;
    wire [3:0]score,miss;
    wire [7:0]out_data;
    wire [1:0]life;
    wire [2:0]state_choice;
    assign out_data=((state==3'b110)?{6'd0,life[1:0]}:{miss[3:0],score[
3:0]});

    bt_control b1(clk,!rst,get_bluetooth,state,dir,state_choice);

    controller ctrl(clk,rst,miss,score,state_choice,life,state);

    play_screen screen(clk,rst,state,dir,x_valid,y_valid,red_out,green_
out,blue_out,score,miss);

    display7 show(clk,!rst,state,out_data,control_led,out_led);

    tri_color show_miss_score(clk,rst,state,score,miss,tri_color1,tri_c
olor2);

    sound sound_life(clk,state,(rst&&(state==3'b110)),AOUT,life);
endmodule
```

（2） 顶层控制器 controller：

根据子系统反馈信号完成状态转移。

功能框图：



接口定义：

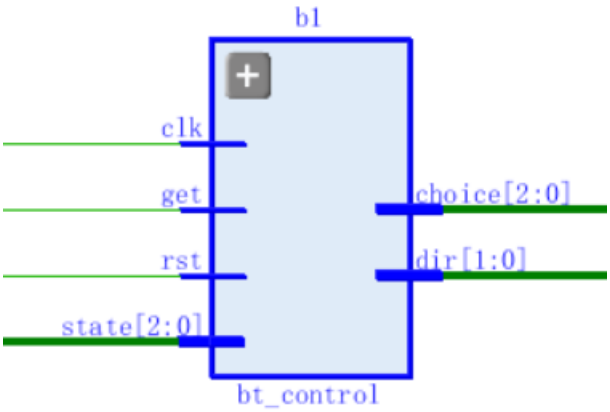
接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz
rst	input	复位信号,低电平有效
miss	input	游戏错失值
score	input	游戏得分
state_choice	input	蓝牙指令
life	input	生命值
state	output	状态

```
module controller(
    input clk,
    input rst,
    input [3:0]miss,
    input [3:0]score,
    input [2:0]state_choice,//蓝牙
    input [1:0]life,
    output reg [2:0]state
);
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        state<=3'b000;
    end
    else
    begin
        if(state<=3'b000)
            state<=3'b001;//显示开启画面
        else
        begin
            if(state_choice==3'b010&&life==2'b11)
                state<=3'b010;//是否开始 or 退出
            else if(state_choice==3'b110&&life==2'b11)
                state<=3'b100;//是否开始 or 退出
            else if(state_choice==3'b101)
                state<=3'b101;//是否开始 or 退出
            else if(miss>4'd3)
                state<=3'b110;  //充能
            else if(miss+score==4'd10)
                state<=3'b100;  //结束游戏
        end
    end
end
```

```
endmodule
```

（3） 蓝牙模块 bt_control:

功能框图：



接口定义：

接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz
rst	input	复位信号，高电平有效
get	input	串口接收端
state	input	状态
choice	output	手机端选择指令
dir	output	选择移动方向

```
module bt_control(
    input clk,
    input rst,
    input get,
    input [2:0]state,
    output [1:0]dir,
    output [2:0]choice
    );
    parameter bps=10417;//对应 9600 波特率
    reg [14:0] count_1;
    reg [3:0] count_2;
    reg buffer_0,buffer_1,buffer_2;//除去滤波
    wire buffer_en;//检测到边沿
    reg add_en;//计数使能信号
    reg [7:0]out;

    assign dir=(state==3'b010)?{out[3],out[0]}:2'b00;
    assign choice[2:0]=out[6:4];

    always @ (posedge clk)
    begin
```

```
    if(rst)
    begin
        buffer_0<=1;
        buffer_1<=1;
        buffer_2<=1;
    end
    else
    begin
        buffer_0<=get;
        buffer_1<=buffer_0;
        buffer_2<=buffer_1;
    end
end

assign buffer_en=buffer_2&~buffer_1;

always @ (posedge clk)
begin
    if(rst)
    begin
        count_1<=0;
    end
    else if(add_en)
    begin
        if(count_1==bps-1)
        begin
            count_1<=0;
        end
        else
        begin
            count_1<=count_1+1;
        end
    end
end

always @ (posedge clk)
begin
    if(rst)
    begin
        count_2<=0;
    end
    else if(add_en&&count_1==bps-1)//如果每一位加
    begin
        if(count_2==8)
```

```
            begin
                count_2<=0;
            end
            else
            begin
                count_2<=count_2+1;
            end
        end
    end

    always @ (posedge clk)
    begin
        if(rst)
        begin
            add_en<=0;
        end
        else if(buffer_en)
        begin
            add_en<=1;
        end
        else if(add_en&&count_2==8&&count_1==bps-1)
        begin
            add_en<=0;
        end
    end

    always @ (posedge clk)
    begin
        if(rst)
        begin
            out<=8'd0;
        end
        else if(add_en&&count_1==bps/2-1&&count_2!=0)
        begin
            out[count_2-1]<=get;
        end
    end
endmodule
```

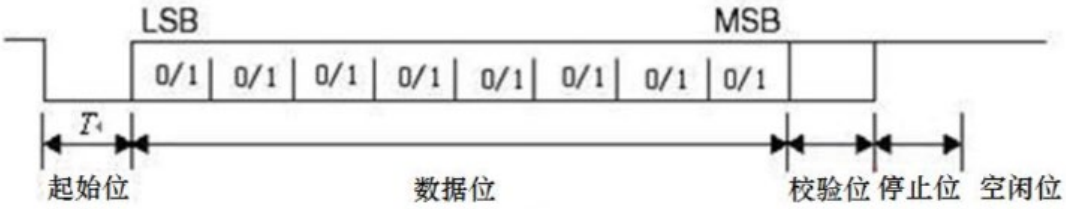
蓝牙模块利用的是 UART 协议进行传输，此蓝牙模块支持的是 8 位数据位，一位停止位，无奇偶校验位，波特率为默认的 9600，配对密码 1234 同时为了保证传输数据的有效，加了滤波处理模块，传入的时钟为 100Mhz 既可。

串口通信协议 UART 介绍：UART 作为异步串口通信协议的一种，工作原理是将传输数据的每个字符一位接一位地传输。

其中每一位(Bit)的意义如下：起始位：先发出一个逻辑”0”的信号，表示传输字符的开始。 数据位：紧接着起始位之后。数据位的个数可以是 4、5、6、7、8 等，构成一个 字符。通常采用 ASCII 码。从最低位开始传送，靠时钟定位。 奇偶校验位：数据位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇 校验)，以此来校验数据传送的正确性。 停止位：它是一个字符数据的结束标志。可以是 1 位、1.5 位、2 位的高电平。

UART 首先将接收到的并行数据转换成串行数据来传输。消息帧从一个低位起始位开始，后面是 7 个或 8 个数据位，一个可用的奇偶位和一个或几个高位停止位。接收器发现开始位时它就知道数据准备发送，并尝试与发送器时钟频率同步。如果选择了奇偶校验，UART 就在数据位后面加上奇偶位。奇偶位可用来帮助错误校验。在接收过程中，UART 从消 息帧中去掉起始位和结束位，对进来的字节进行奇偶校验，并将数据字节从串行转换成并行。UART 传输时序如下图所示 ：

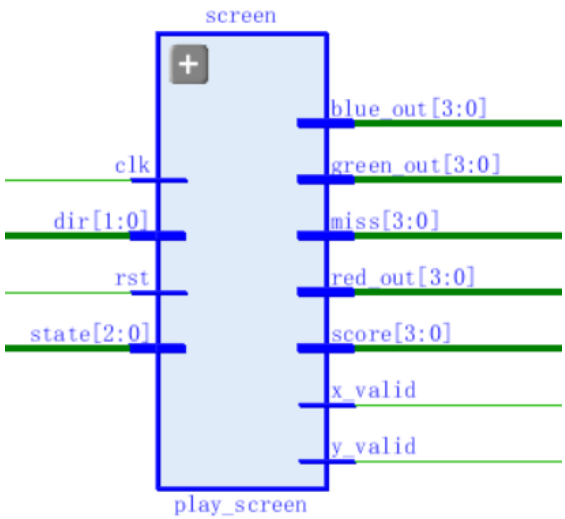
=



由于 UART 是异步传输，没有传输同步时钟。为了能保证数据传输的正确性，UART 采用 16 倍数据波特率的时钟进行采样。每个数据有 16 个时钟采样，取中间的采样值，以保证采样不会滑码或误码。一般 UART 一帧的数据位数为 8，这样即使每个数据有一个时钟的误差，接收端也能正确地采样到数据。

（4） vga 显示屏模块

用于显示图像信息，下有子模块 vga_control 和 vga_display，vga_control 用于控制 vga 时序，vga_display 处理像素输出。



接口定义：

接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz

rst	input	复位信号，低电平有效
state	input	状态
dir	input	选择移动方向
x_valid	output	行时序信号
y_valid	output	场时序信号
red_out,green_out,blue_out	output	rgb 像素信息
score	output	得分
miss	output	错失值

```
module play_screen(
    input clk,
    input rst,//低电平有效
    input [2:0]state,
    input [1:0]dir,
    output x_valid,//行时序信号
    output y_valid,//场时序信号
    output [3:0]red_out,green_out,blue_out,//rgb 像素信息
    output [3:0]score,
    output [3:0]miss
    );
wire [15:0]data_color;
reg [13:0]addr;
wire clk_vga;
wire addr_ena;
wire [11:0]x_begin;
wire [11:0]y_begin;

wire [11:0]obj1_x_begin;
wire [11:0]obj1_y_begin;
wire [11:0]obj2_x_begin;
wire [11:0]obj2_y_begin;
wire obj1_ena;
wire end_show1;
wire obj2_ena;
wire end_show2;
wire [3:0]miss1;
wire [3:0]score1;
wire [3:0]miss2;
wire [3:0]score2;

reg [20:0]cnt;
reg [7:0]wait_cnt;
wire rst1;
reg rst2;
```

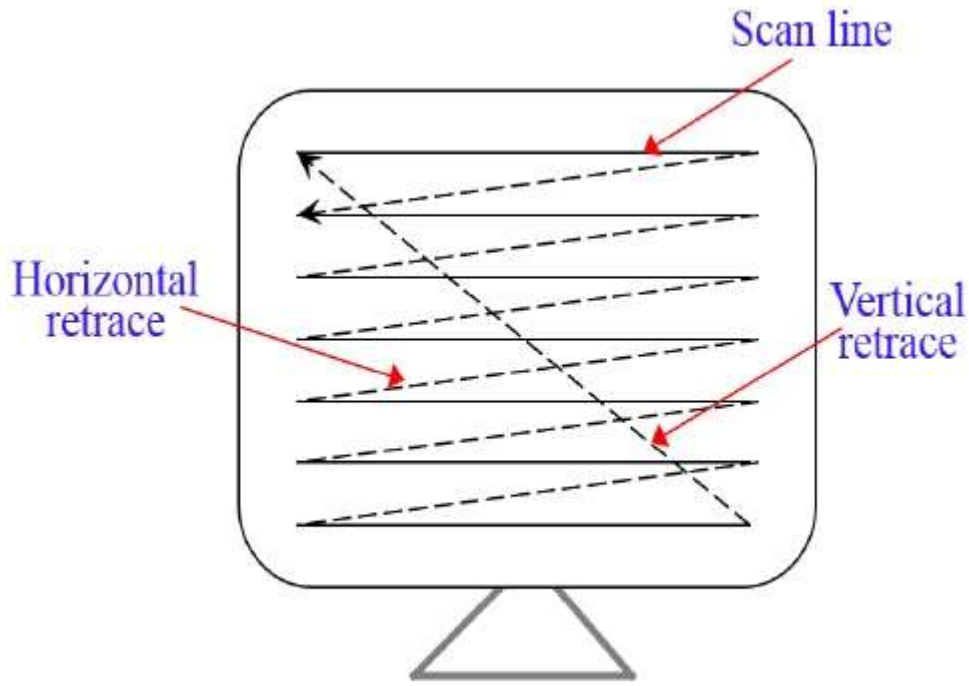
```
assign score=score1+score2;
assign miss=miss1+miss2;

always@(posedge clk_vga or negedge rst)
begin
    if(!rst)
    begin
        addr<=0;
    end
    else
    begin
    if(addr_ena)
        begin
            if(addr<14'd9999)
                addr<=addr+1;
            else
                addr<=0;
        end
    end
end

clk_wiz_0 div(.clk_in1(clk),.clk_out1(clk_vga));
deal_p x1(clk,(!rst||(state!=3'b010&&state!=3'b100&&state!=3'b110)),dir
,addr_ena,x_begin,y_begin); //处理人物位置
blk_mem_gen_0 p1(.clka(clk_vga),.addra(addr),.douta(data_color));

assign obj1_ena=~end_show1;
assign obj2_ena=~end_show2;

assign rst1=(!rst||(state!=3'b010)||(y_begin!=11'd379));
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        cnt<=21'd0;
    end
    else
    begin
        cnt<=cnt+1;
    end
end

always@(posedge cnt[20] or posedge rst1)
```

```
begin
    if(rst1)
    begin
        wait_cnt<=8'd0;
        rst2<=0;
    end
    else
    begin
        if(wait_cnt!=8'd200)
            wait_cnt<=wait_cnt+1;
        else
            rst2<=1;
    end
end

dealXY y1(clk,rst1,(21'b111101111011110111010),obj1_ena,state,x_begin,ob
j1_x_begin,obj1_y_begin,end_show1,score1,miss1),//处理物体位置,miss 也可以
放进去
       y2(clk,(!rst2||(state!=3'b010)||(y_begin!=11'd379)),(21'b1001011
0101010110110),obj2_ena,state,x_begin,obj2_x_begin,obj2_y_begin,end_sho
w2,score2,miss2);//处理物体位置,miss 也可以放进去

vga_display #(11'd100,11'd100) d1(clk_vga,!rst,end_show1,end_show2,x_be
gin,y_begin,obj1_x_begin,obj1_y_begin,obj2_x_begin,obj2_y_begin,data_co
lor,x_valid, y_valid,red_out,green_out,blue_out,addr_ena);
endmodule
```

VGA 全称是 Video Graphics Array，即视频图形阵列，是模拟信号的一种视频传输标准，VGA 接口是一种 D 型接口，上面共有 15 针孔，分成三排，每排五个，其中比较重要的是 3 根 RGB 彩色分量信号和 2 根扫描同步信号 HSYNC 和 VSYNC 针。

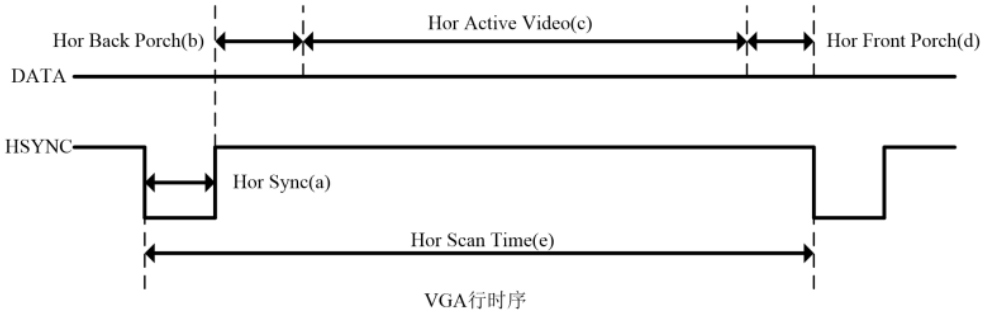
我们一般使用的屏幕分辨率大小是 640*480，640*480 的规格就是显示屏幕上每行有 640 个像素点，总共有 480 行。注意，一件很重要的事情是，虽然你看到的屏幕大小是 640*480 的，但是它的实际大小并不只有那么点，形象一点就是说，VGA 扫描的范围是包含了你能够看到的 640*480 这一块区域的更大区域，他会在周围一圈你看不到的区域部分进行扫描，因此，我们在处理扫描信号的时候一定要注意只有扫描到有效区域的时候才能把像素点数据传给 VGA 显示

VGA 显示器扫描方式从屏幕左上角一点开始，从左向右逐点扫描，每扫描完一行，电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，开始下一帧。完成一行扫描的时间称为水平扫描时间，其倒数称为行频率；完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率，即屏幕的刷新频率。其扫描示意图如下图所示

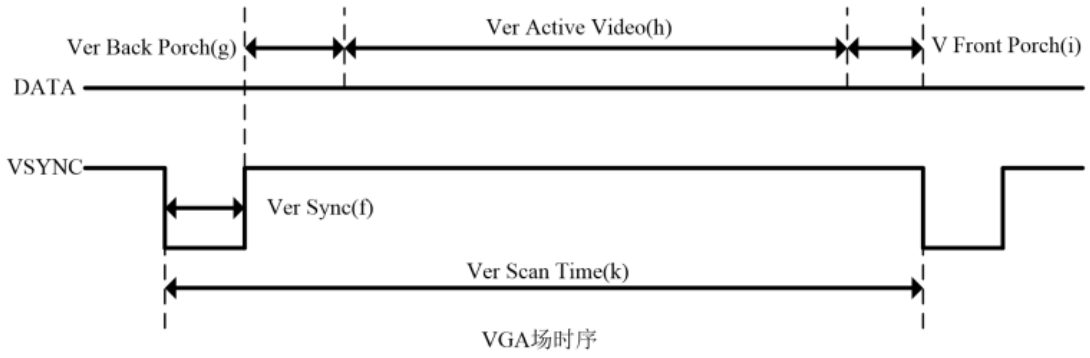


Raster scanning terminology for a single output frame.

行扫描：Hor Sync 、Hor Back Porch 、Hor Active Video 和 Hor Front Porch
Hor Scan Time 是一个扫描周期，它会先扫描到 Hor Sync、再扫描 Hor Back Porch，然后才进入有效显示区 Hor Active Video，最后是一段 Hor Front Porch；可以看出来，四段区间只有 Hor Active Video 这一段是能够正常显示图像信息的，也就是屏幕上显示的那一块区间



列扫描也同理

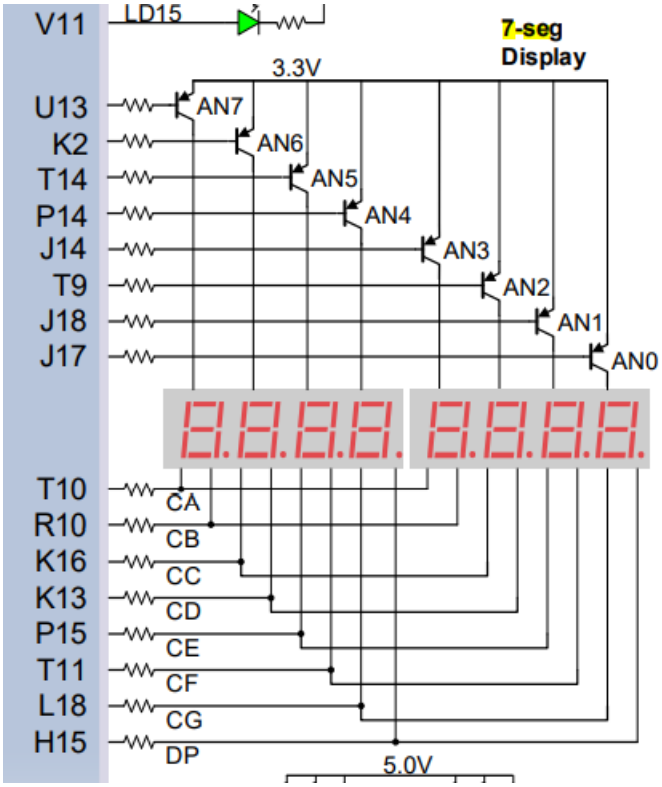


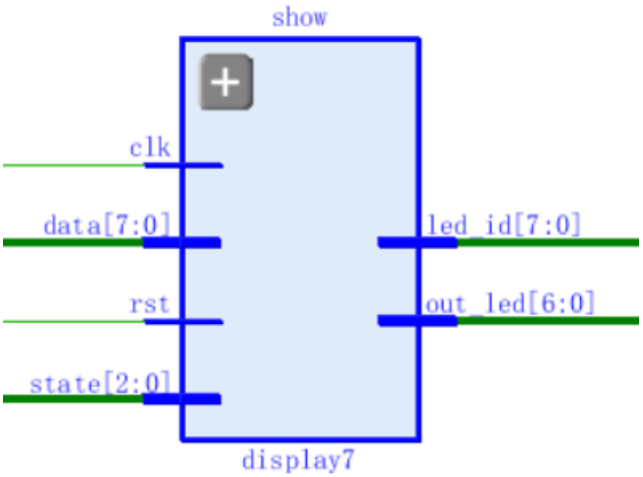
无论是行扫描还是列扫描还是给像素点赋值，我们都需要用一个时钟，必须要先将 100MHZ

的系统时钟分频为 25MHZ，然后拿去作为扫描信号的时钟，这里我们用到了 Clocking Wizard IP 核，以及存储图片用到了 block Memory IP 核

（5） 七段数码管

选用板子上的控制信号来达到视觉上每一位显示不同数字的目的。





接口定义

接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz
rst	input	复位信号，低电平有效
state	input	状态
data	input	需要显示的数据

led_bit	output	七段数码管位控
out_led	output	七段数码管显示

```
module display7(
    input clk,
    input rst,
    input [2:0]state,
    input [7:0]data,
    output reg [7:0] led_bit,   //位控
    output reg [6:0] out_led
    );

    reg [3:0] temp_data;

    reg trans_en=1;
    wire trans_done;
    reg [2:0] id;
    reg [2:0] data_id;
    wire pclk;
    divider m (clk,rst,pclk);
    always @(posedge pclk or posedge rst)
    begin
    if(rst)
    begin
        id<=0;
        data_id<=0;
        led_bit <= 8'b11111111;
    end
    else
    begin
        case(id)
            0 : begin led_bit <= 8'b11111110; id<=1;data_id<=0; end
            1 : begin led_bit <= 8'b11111101; id<=2;data_id<=1; end
            2 : begin led_bit <= 8'b11111011; id<=3;data_id<=2; end
            3 : begin led_bit <= 8'b11110111; id<=4;data_id<=3; end
            4 : begin led_bit <= 8'b11101111; id<=5;data_id<=4; end
            5 : begin led_bit <= 8'b11011111; id<=6;data_id<=5; end
            6 : begin led_bit <= 8'b10111111; id<=7;data_id<=6; end
            7 : begin led_bit <= 8'b01111111; id<=0;data_id<=7; end
            default: id<=1;
        endcase
    end
    end
    always @(*)
```

```
    begin
         case(data_id)
                3'b000: temp_data <=((state==3'b001)? 4'h9:((data[3:0]==4'd10) ? 4'h0:data[3:0]));
                3'b001: temp_data <=((state==3'b001)? 4'h1:((data[3:0]==4'd10) ? 4'h1:4'hF));
                3'b010: temp_data <=((state==3'b001)? 4'h2:4'hF);
                3'b011: temp_data <=((state==3'b001)? 4'h2:4'hF);
                3'b100: temp_data <=((state==3'b001)? 4'h5:((state==3'b110)?4'hf:data[7:4]));
                3'b101: temp_data <=((state==3'b001)? 4'h9:4'hF);
                3'b110: temp_data <=((state==3'b001)? 4'h1:4'hF);
                3'b111: temp_data <= 4'hF;
                default:
                   temp_data <= 4'b1111;
           endcase
    end
    always @(*)
    case(temp_data)
        4'b0000: out_led <=7'b1000000;    //0
        4'b0001: out_led <=7'b1111001;
        4'b0010: out_led <=7'b0100100;
        4'b0011: out_led <=7'b0110000;
        4'b0100: out_led <=7'b0011001;
        4'b0101: out_led <=7'b0010010;
        4'b0110: out_led <=7'b0000010;
        4'b0111: out_led <=7'b1111000;
        4'b1000: out_led <=7'b0000000;
        4'b1001: out_led <=7'b0010000;
        default: out_led <=7'b1111111;
    endcase
endmodule
```

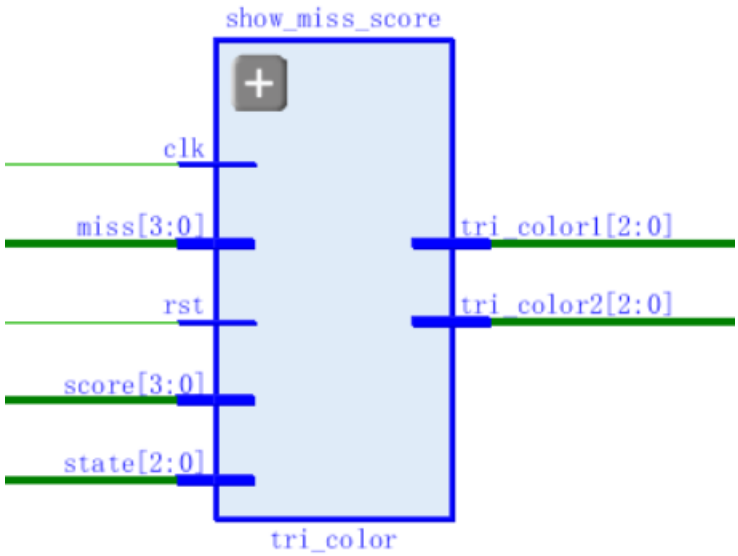
扫描频率太低数码管会出现闪烁的现象，频率太高则亮度不够甚至无法看清，所以一般扫描间隔多为几毫秒。根据多次测试选择了以下分频参数，间隔 1ms 左右，效果较好。

```
module divider #(parameter div=16)
(
    input clk100M,
    input rst,
    output clk1
);
 reg [div:0] count;
assign clk1 = count[div];
always @ (posedge clk100M or posedge rst)
```

```
begin
    if (rst)
    count <= 0;
    else
    count <= count+1;
end
endmodule
```

（6） 三色 led 灯模块

根据 score 和 miss 的变化情况，当 score 变化时闪蓝绿光和 miss 变化时闪红光。



接口定义

接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz
rst	input	复位信号，低电平有效
state	input	状态
score	input	得分
miss	input	错失值
tri_color1	output	LD17rgb 灯输出信号
tri_color2	output	LD16rgb 灯输出信号

```
module tri_color(
    input clk,
    input rst,
    input [2:0]state,
    input [3:0]score,
    input [3:0]miss,
    output reg[2:0]tri_color1,
    output reg[2:0]tri_color2
```

```
);
reg [3:0]score_change;
reg [3:0]miss_change;
wire mid_clk;
Divider divider2(.I_CLK(clk),.O_CLK(mid_clk));

always @ (posedge mid_clk or negedge rst)
begin
    if (!rst||state!=3'b010)
    begin
        score_change<= score;
        miss_change<= miss;
    end
    else
    begin
        if(miss_change!=miss)
        begin
            tri_color1<=3'b100;
            miss_change<=miss;
        end
        else
            tri_color1<=3'b000;
        if(score_change!=score)
        begin
            tri_color2<=3'b011;
            score_change<=score;
        end
        else
            tri_color2<=3'b000;
    end
end
endmodule
```

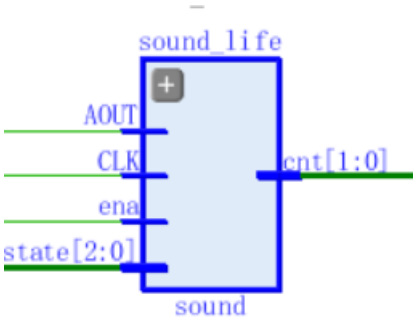
建议使用 PWM,但在本实验中对 led 颜色及亮度的要求不高，且技术较为复杂，未应用。实际效果可能 led 灯略微刺眼，但显色效果很好。

> Note: Digilent strongly recommends the use of pulse-width modulation (PWM) when driving the tri-color LEDs (for information on PWM, see section 15.1 Pulse Density Modulation (PDM)). Driving any of the inputs to a steady logic '1' will result in the LED being illuminated at an uncomfortably bright level. You can avoid this by ensuring that none of the tri-color signals are driven with more than a 50% duty cycle. Using PWM also greatly expands the potential color palette of the tri-color led. Individually adjusting the duty cycle of each color between 50% and 0% causes the different colors to be illuminated at different intensities, allowing virtually any color to be displayed.

（7） 声音传感器模块

声音传感器有效时，检测周围达到一定音量强度后持续一定时间后（基准 0.05s）计

数加一。



接口定义

接口名称	接口属性	接口描述
clk	input	开发板系统时钟，100MHz
ena	input	使能信号，低电平有效
state	input	状态
AOUT	input	模拟量输出
cnt	output	生命值计数

```
module sound(
    input CLK,
    input [2:0]state,
    input ena,
    input AOUT,
    output reg [1:0]cnt
    );
reg pause;
wire mid_clk;
Divider divider1(.I_CLK(CLK),.O_CLK(mid_clk));

always @(posedge mid_clk or negedge ena)
begin
    if(!ena)
    begin
        if(state==3'b001||state==3'b100||state==3'b101)
            cnt<=3;
        else
            cnt<=0;
        pause<=1;
    end
    else if(!AOUT)
        pause<=1;
    else if(pause)
    begin
        pause<=0;
        if(cnt!=2'b11)
```

```
        begin
            cnt<=cnt+1;
        end
    end
end
endmodule
```

五、测试模块建模

仅部分模块使用 testbench 进行测试。

七段数码管：

```
module display7_tb;
    reg [3:0] iData;     //4 位输入 D,～D。
    wire [6:0] oData;    //7 位译码输出 g～a
display7 uut(
.iData(iData),
.oData(oData)
);
initial
begin
    iData=4'h0;
    #100;
    iData=4'h1;
    #100;
    iData=4'h2;
    #100;
    iData=4'h3;
    #100;
    iData=4'h4;
    #100;
    iData=4'h5;
    #100;
    iData=4'h6;
    #100;
    iData=4'h7;
    #100;
    iData=4'h8;
    #100;
    iData=4'h9;
end
endmodule
```

VGA 显示屏：

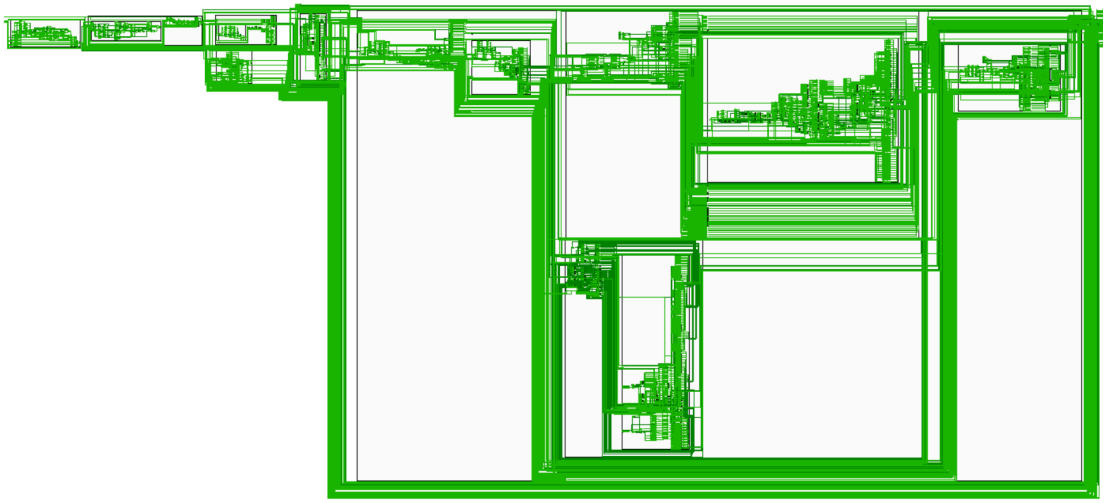
```
module screen_tb;
    reg CLK;
    reg get;
    wire x_valid;//行时序信号
    wire y_valid;//场时序信号
    wire [3:0]  red_out,green_out,blue_out;//rgb 像素信息
    wire [7:0]out;
    screen uut(
    .clk(CLK),
    .get_bluetooth(get),
    .x_valid(x_valid),
    .y_valid(y_valid),
    .red_out(red_out),
    .green_out(green_out),
    .blue_out(blue_out),
    .bt(out)
    );
    initial
    begin
        get=0;
        CLK=0;
    end
    always #1 CLK=~CLK;
    always #3 get=~get;
endmodule
```

声音传感器：

```
module sound;
    reg AOUT;
    reg DOUT;
    reg ena;
    wire test;
    sound uut(AOUT,DOUT,ena,test) ;
endmodule
```

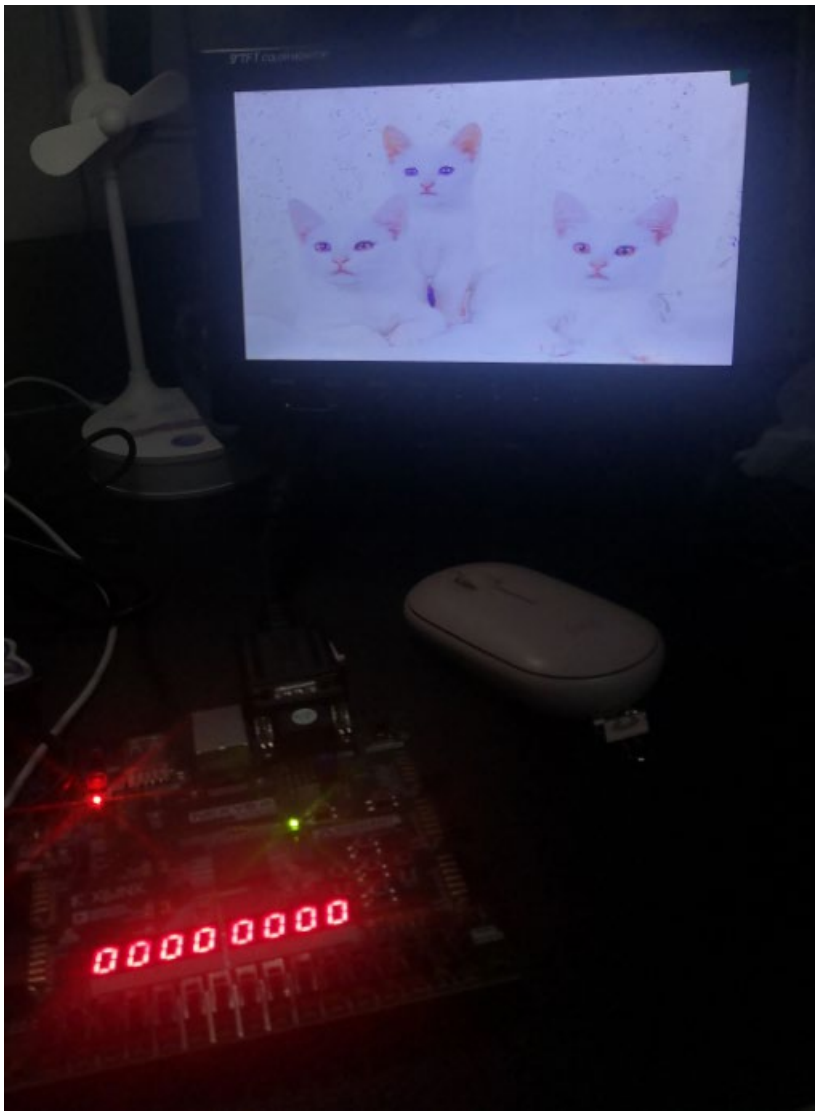
六、实验结果

设计原理图：

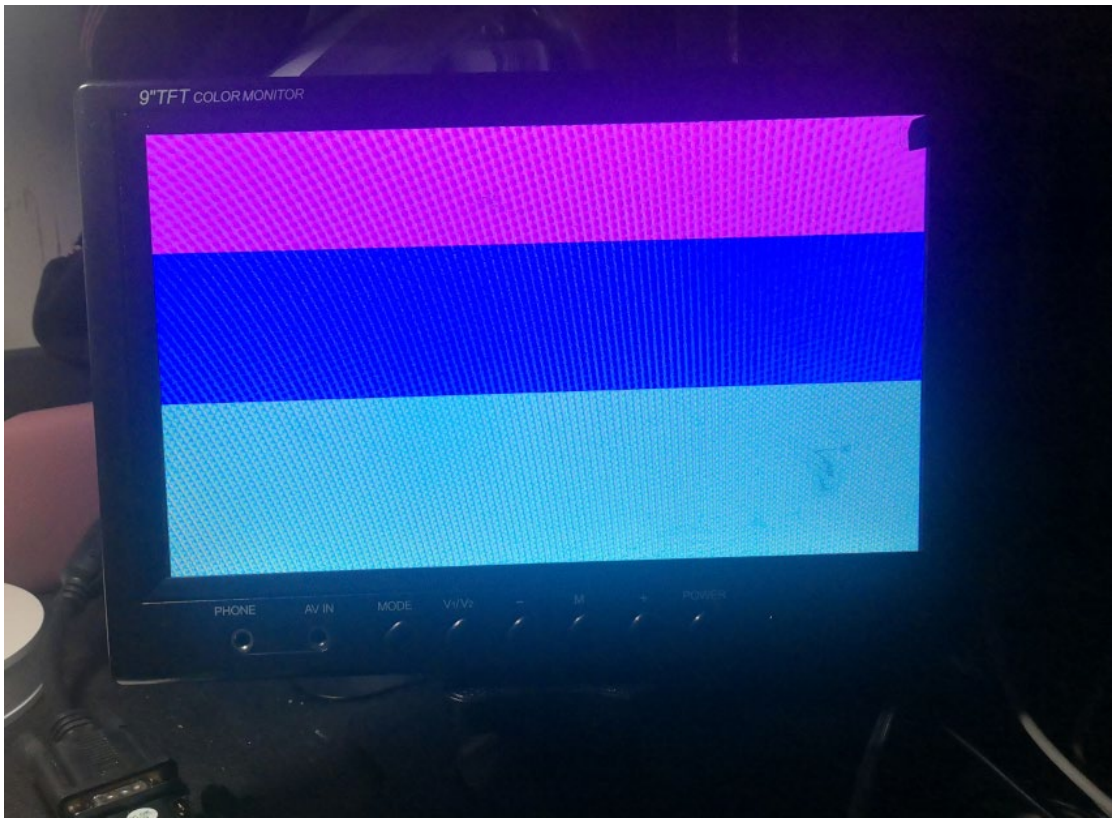


VGA 显示测试：

单图片测试（附带数码管显示）：

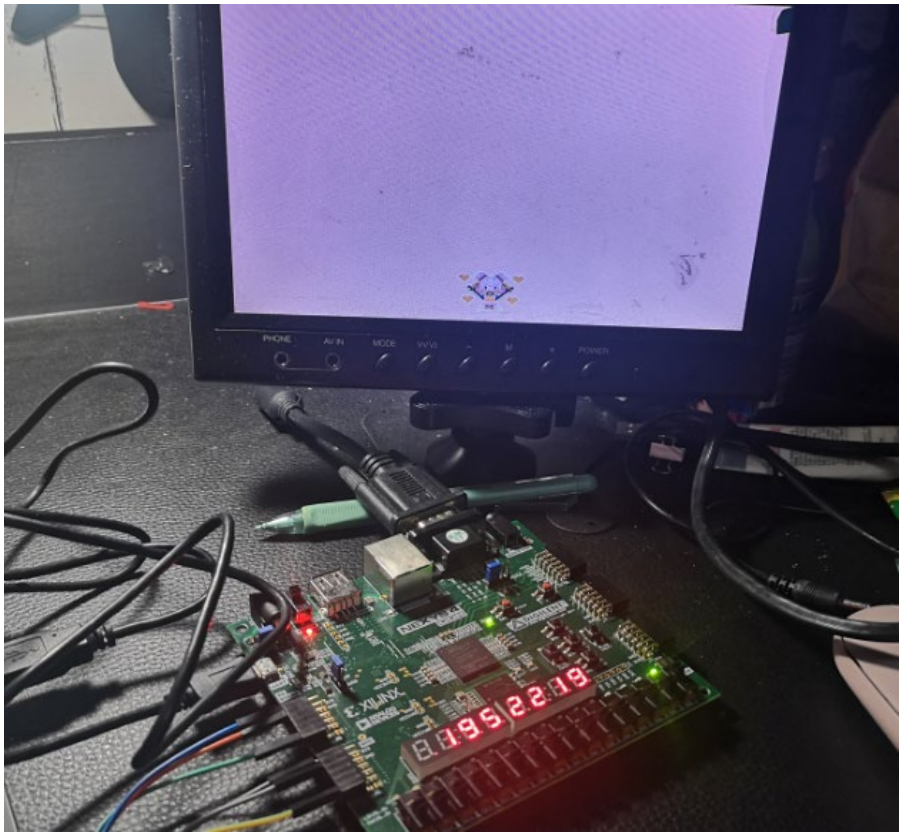


彩条测试：



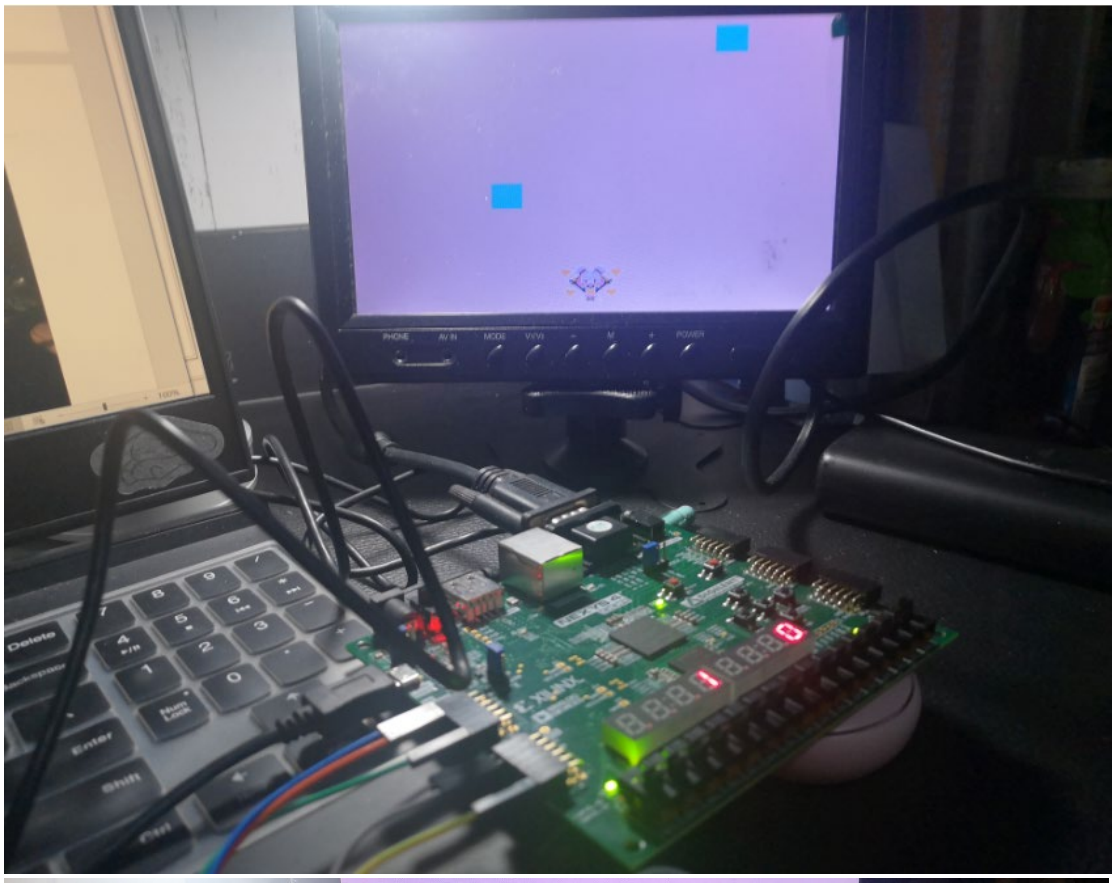
开机：

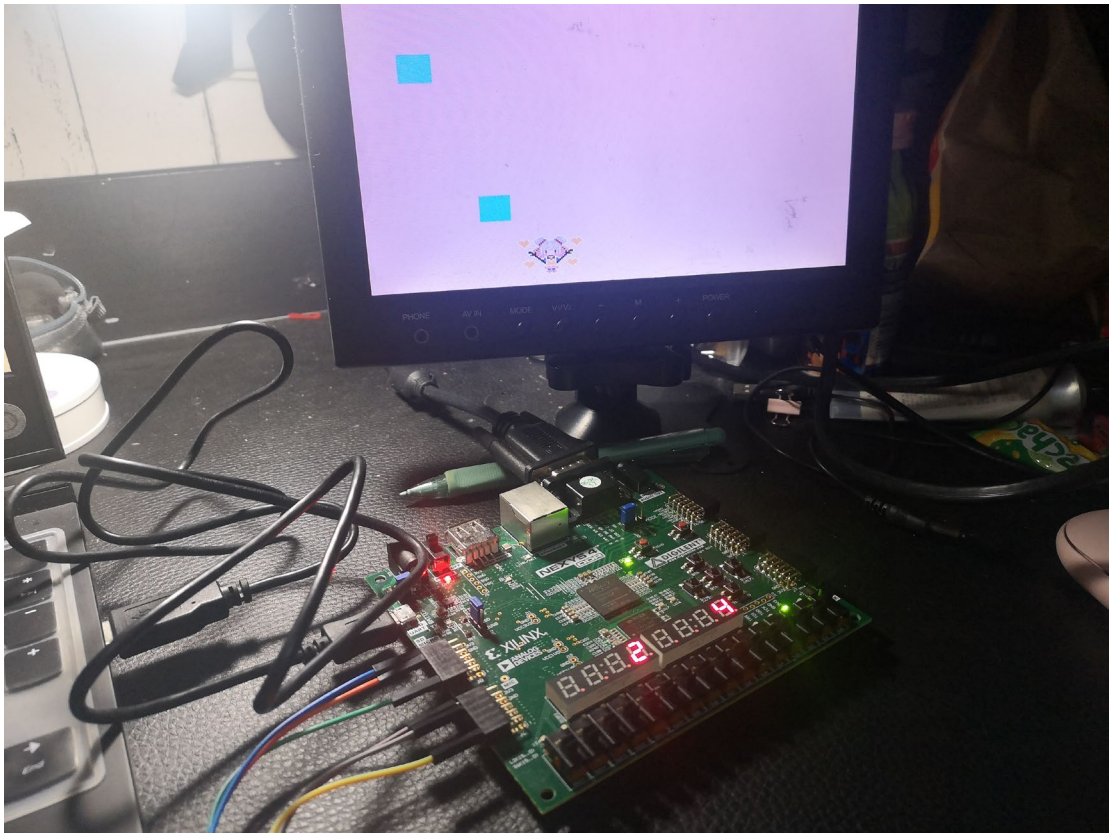
初始界面小人处于正中央，无法移动 除非按“开始”键开始游戏。七段数码管显示学号。



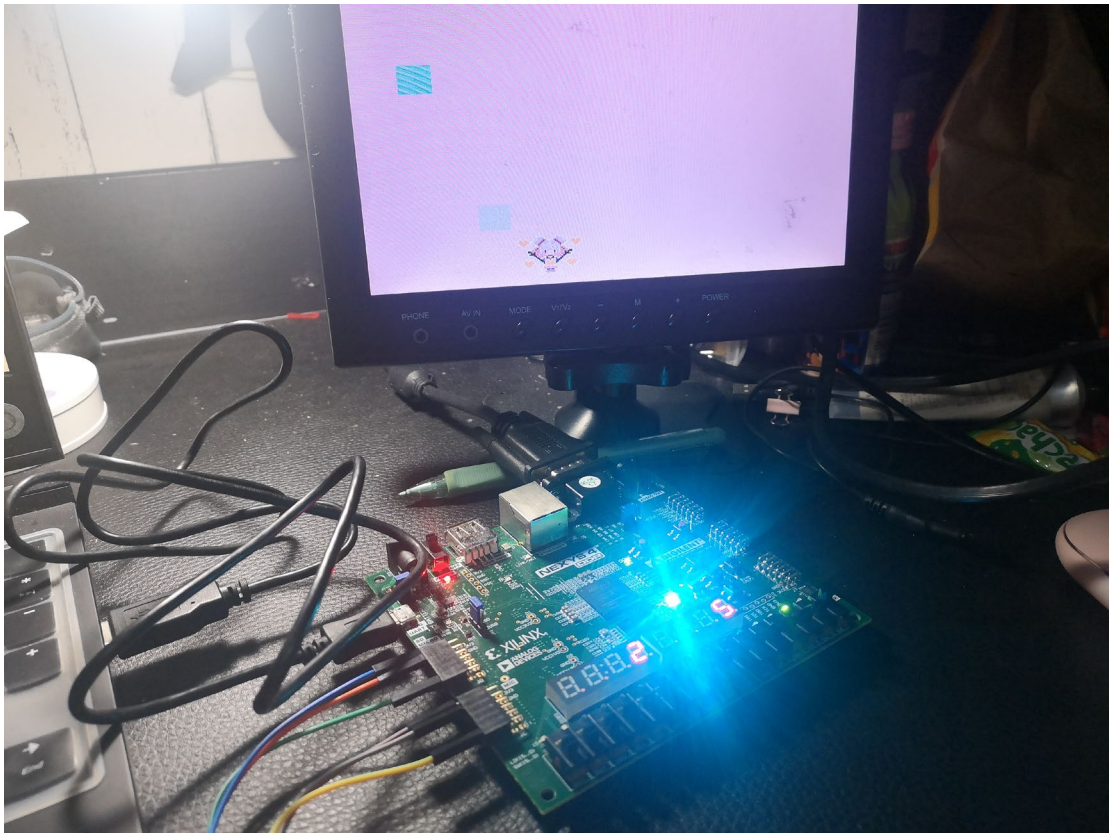
手机端按键：

游戏中：
七段数码管左半部分显示错过的物体数量，右半部分显示得分。
得分时闪蓝绿光，错过时闪红光。

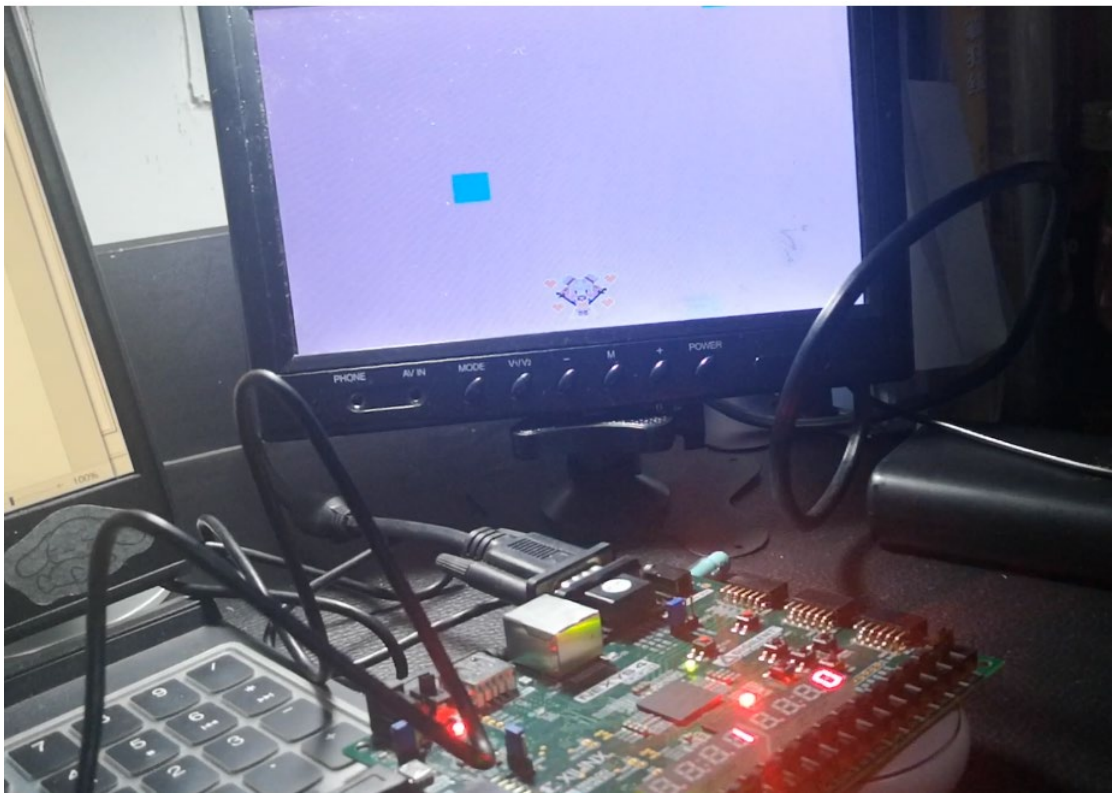




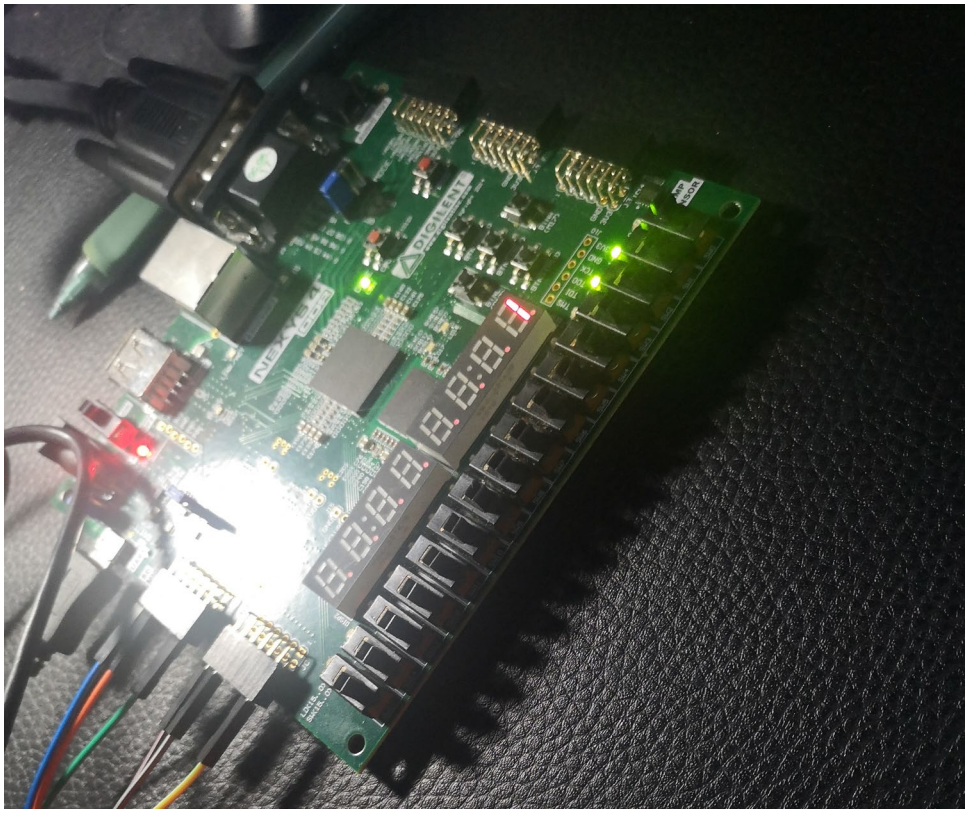
接中左侧方块

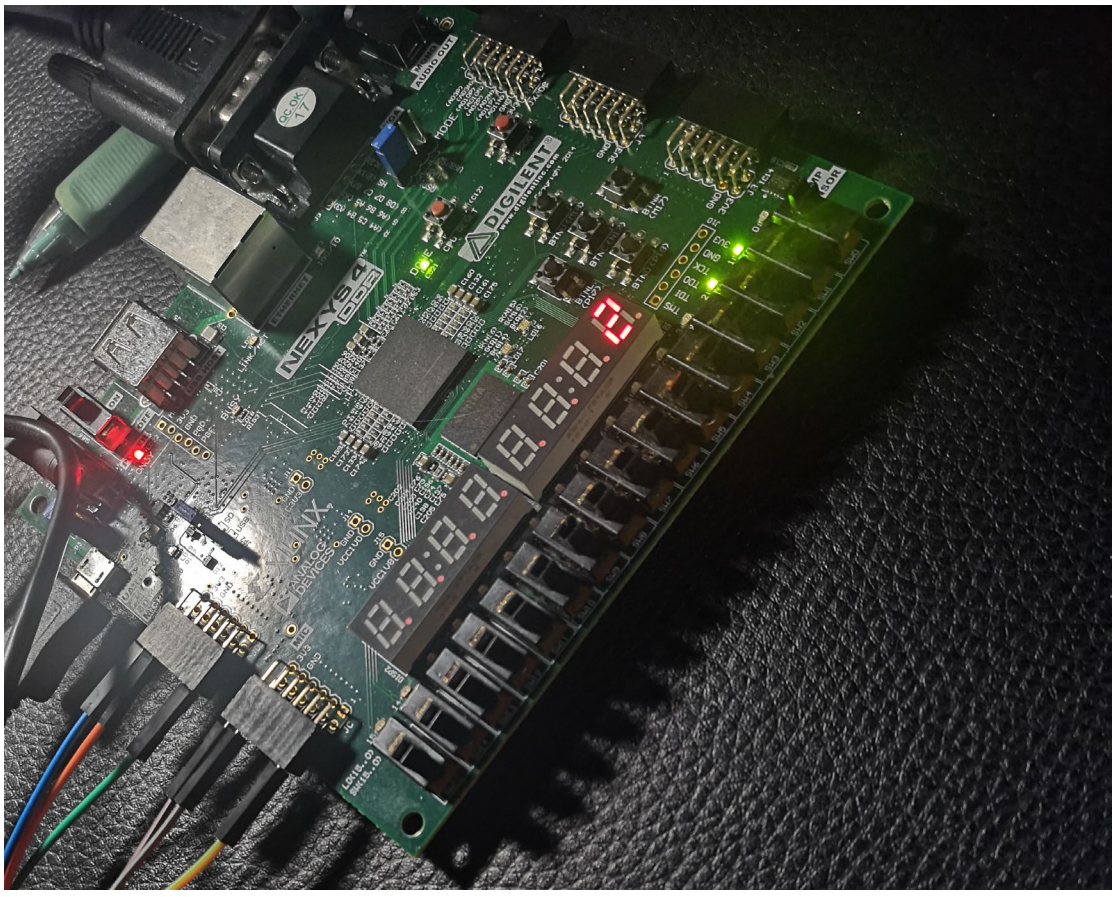


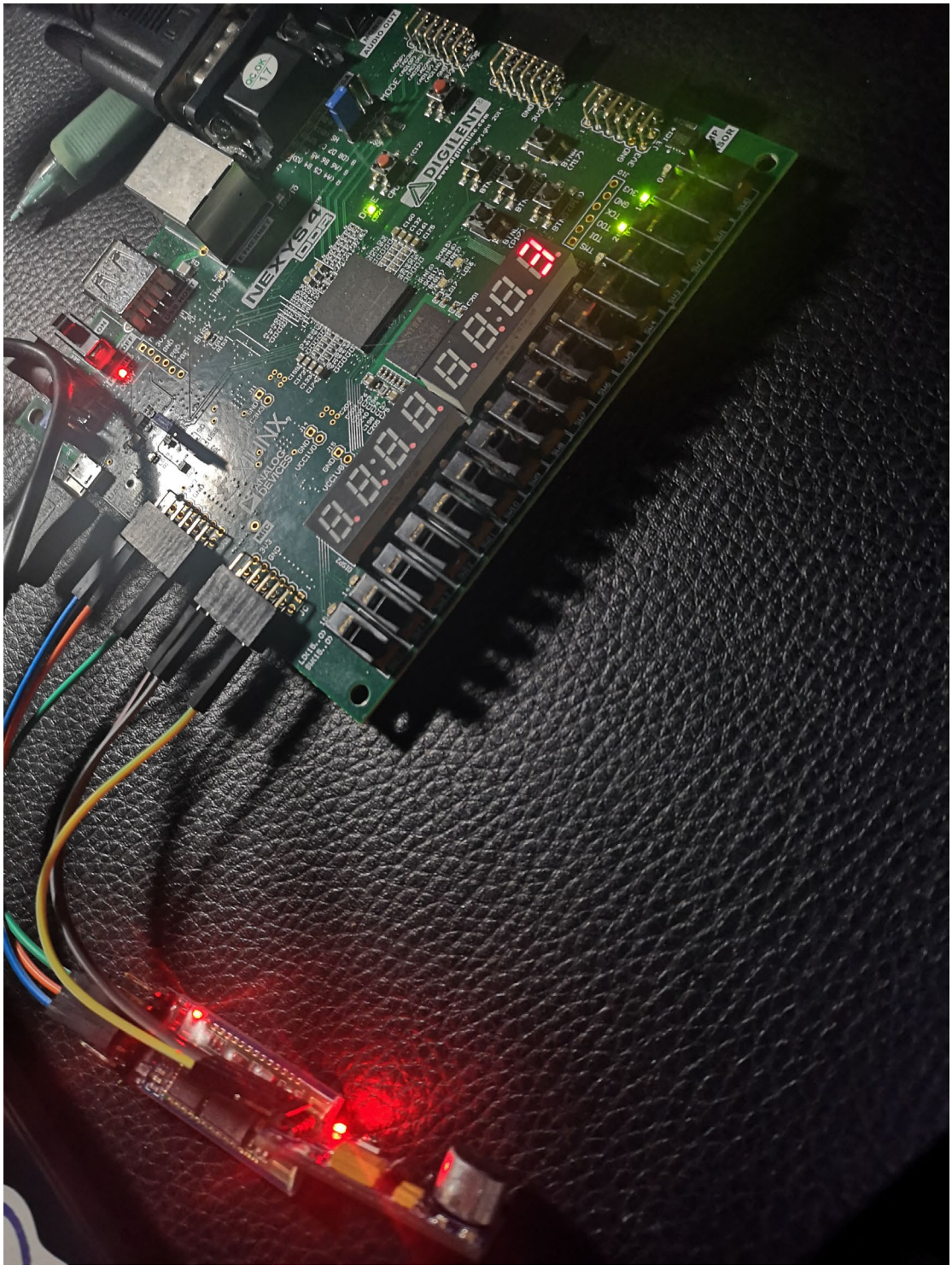
错过右侧方块：



声音传感器充能：







充满了，可以重新开始游戏了！（ps:在恢复体力之前是不可以继续游戏的，除非退出重开）。

七、结论

各个子模块均按预期运行，该数字系统工作正常。
对于三色 led 灯的显示，可以研究一下 PWM 技术，使其亮度和显色达到更好的效果。
声音传感器的数字量输出和模拟量输出可以得到更好的应用。

对进行过拉伸、缩放的图片存入 IP 核后，读出时会出现图片扭曲，变成黑白色的情况，本来以为是个例，但其他同学也有遇到过此情况。查找资料未果，原因不明。此情况导致我耗费大量精力去寻找素材及处理图片，但绝大部分都无法使用，故最终界面简化。

八、心得体会及建议

（1）本课程收获：

本课程及此次大作业都让我收获颇丰。老师讲课清楚有条理，并有合理适当的课堂互动。网站的学习资料内容丰富详实，对课程学习有很大帮助。本课程上机实验占很重要的一部分，学校配发的硬件及提供的环境很好，总体来说学习环境、上课效果、上机实验条件和学习资料都很不错。通过本学期的学习，我接触了解了硬件及 Verilog 语言，应用数字电路完成各个实验及进行数字系统逻辑设计。

本次大作业的完成过程中，无论是知识的学习积累、资料的搜集筛选、设计编写并修正一个大作业的过程，都让我的相应能力有所提高。但我仍有很多不足，比如在数字系统自上而下再自下而上的设计方法上，我在自下而上这方面做得还行，两天时间就使各个子模块可以单独使用。但因为我没有规划好顶层设计，导致我在将各个模块连接起来时出现了许多问题，比如各模块时钟差异、复位信号和使能信号差异等，尤其在图片的显示方面因显示器扫描和取 rgb 数据递增不同步，导致图片出现多种旋转、偏移效果，虽然意外的看起来是不错的动画效果，但不是我想要的效果。

（2）课程建议：

完成此次大作业的过程非常具有挑战力，同时也非常有趣。美中不足的是这学期学业任务较重，需要兼顾多门课程的期末项目及考核等，时间和精力上还是比较紧张，如果大作业能单独作为一门课设，作为假期项目，可以有更大的发挥空间。比如现在我就对我的外设还没实现和利用的部分充满兴趣，但期末实在无法兼顾，以后可能也很难有机会再研究。

在高性能计算互联领域，FPGA 长期是互联方案的有力军，在各种各样的细分领域，例如神经形态计算与脑模拟，FPGA 已经成为了学科的主要工具，因而当下的 FPGA 教学，尤其是计算机系的教学应更加注重信号处理及与已有的计算场景的结合，而不仅仅将其作为体系结构实验的一个工具，发挥其更大的价值。

（3）对国内外数字芯片设计的看法

在芯片设计制造上，国内仍很难实现自给自足，对外国技术及产品的依赖很大，虽然我国近年在该行业上投入很多，但仍没有达到很理想的效果。

近年来，我国现在正在鼓励快速发展 IC 行业，人才空缺也很大。2019 年中国半导体设计企业有 1780 家，其中 88.54%的企业不足 100 人。2020 年上半年中国就新增了 2.6 万家集成电路相关企业，绝大部分只是完成工商注册，少数属于真正进入芯片行业创业，招兵买马。芯片是烧钱的行业，对于处在初创阶段的芯片公司首要任务是融资。因此大部分国内企业去竞争热门产品方向，聚焦主流市场，市场单一化、产品同质化。因此更加渴求能把芯片产品做到有市场竞争力的研发人才就。芯片产品的聚焦竞争，使国内研发竞争变得更焦灼了。

作为计科学子，我们在培养软件能力的同时，也应该更加重视硬件的设计与发展，积极国家政策与时代潮流，努力为祖国建设添砖加瓦。