同济大学计算机系

数字逻辑课程综合实验报告

学　　号 2052333

姓　　名 黄紫盈

专　　业 计算机科学与技术

授课老师 张冬冬

一、实验内容

fpga 制作 flappy bird 游戏（声控版）。

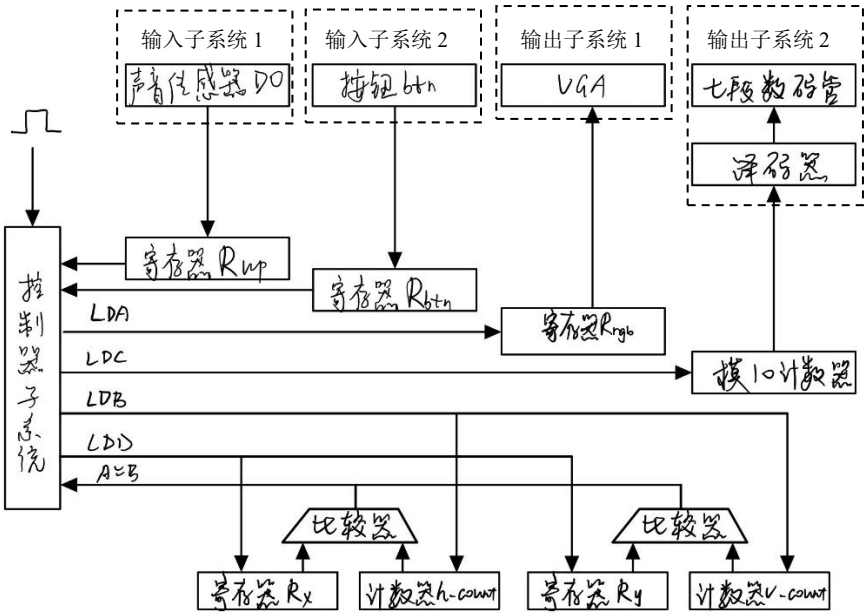
游戏规则：发声——小鸟上移/开始游戏

小鸟碰到地面或上下水管——游戏失败

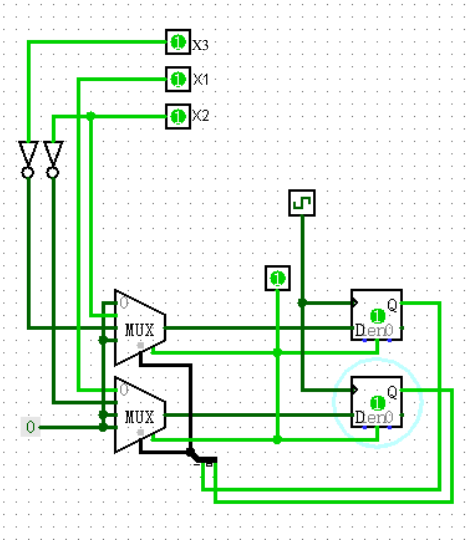
小鸟安全经过一组水管（上下两根）——得一分

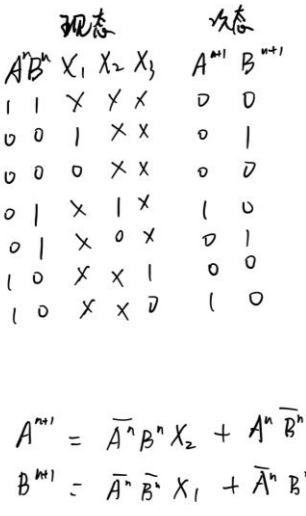
游戏失败后按指定按钮——回到准备界面

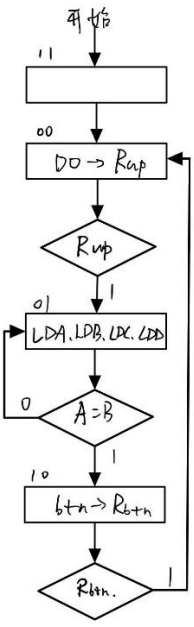
二、xxx 数字系统总框图



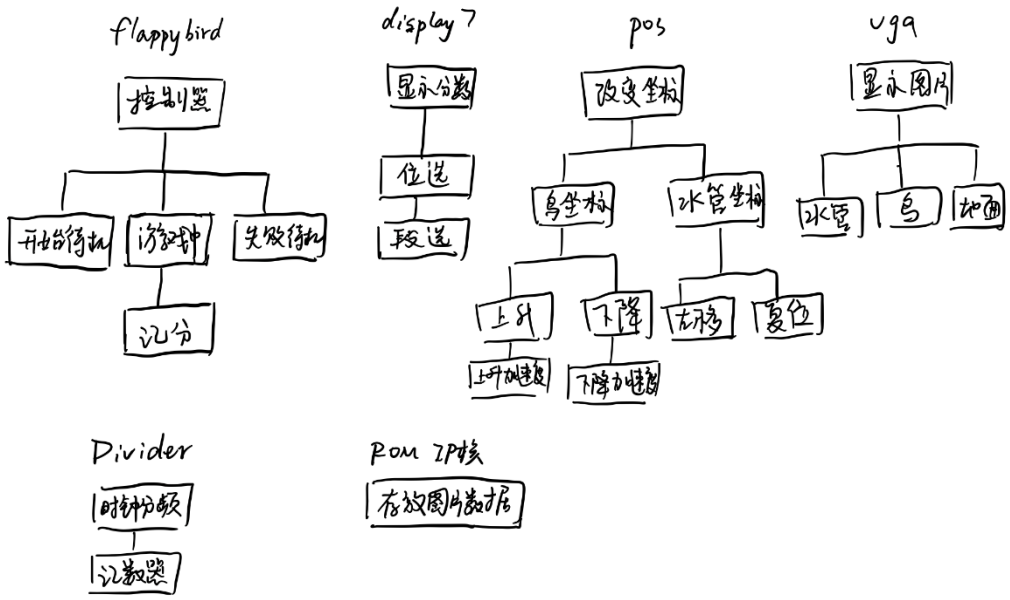
三、系统控制器设计







四、子系统模块建模



1.flappybird(控制器+模 10 计数)

```
module flappybird(
    input oriclk,//100M
    input rst,//复位信号，低电平有效
    input btn,//游戏结束->等待开始状态转变的信号
    input dn,//声音输入信号
    output hsync,//行扫描信号
    output [3:0] blue,
    output [3:0] red,
    output [3:0] green,
    output vsync,//场扫描信号
    output [7:0] seg_select,//位选信号
    output [6:0] seg_LED//段选信号
    );
    wire up;
    assign up=~dn;
    wire clk;//25M
    wire vneg;//完成一帧扫描的信号
    reg start=1'b1;
    reg fail=1'b1;
    wire [9:0] birdy;//鸟纵坐标
    wire [9:0] upx1;//上方水管 1 横坐标
    wire [9:0] upx2;//上方水管 2 横坐标
    wire [9:0] upy1;//上方水管 1 纵坐标
    wire [9:0] upy2;//上方水管 2 纵坐标
```

```
wire [9:0] dnx1;//下方水管
wire [9:0] dnx2;
wire [9:0] dny1;
wire [9:0] dny2;
reg [9:0]birdx=10'd445;//鸟横坐标，不变
reg [3:0]ge=4'd0;//得分个位
reg [3:0]shi=4'd0;//得分十位
reg [3:0]bai=4'd0;//得分百位
always@(posedge vneg or posedge btn or negedge rst )
    begin
    if(~rst)
        begin
        ge<=4'd0;
        shi<=4'd0;
        bai<=4'd0;
        end
    else if(btn)
        begin
        ge<=4'd0;
        shi<=4'd0;
        bai<=4'd0;
        end
    else if(vneg &&(birdx==upx1+10'd52||birdx==upx2+10'd52))
        begin
        if(ge==4'd9)
        begin
            if(shi==4'd9)
                begin
                bai<=bai+4'd1;
                shi<=4'd0;
                end
            else
                shi<=shi+4'd1;
            ge<=4'd0;
        end
        else
            ge<=ge+4'd1;
        end
    end
always@(posedge clk or negedge rst)
begin
if(~rst)
begin
    fail<=1'b0;
```

```
            start<=1'b0;
        end
        else
        begin
        if((birdx<=upx1+10'd51&&birdx>=upx1-10'd33&&(birdy<=upy1+10'd149||birdy>=dny1-
10'd23))||(birdx<=upx2+10'd51&&birdx>=upx2-10'd33&&(birdy<=upy2+10'd149||birdy>=dny2-
10'd23))||birdy>=10'd391)
            begin
            fail<=1'b1;
            start<=1'b0;
            end
        if(start&&fail)
            begin
            start<=1'b0;
            fail<=1'b0;
            end
        else if(fail&&~start&&btn)
            fail<=1'b0;
        else if(~start&&~fail&&up)
            start<=1'b1;
        end
        end
        Divider#(4) d1(.I_CLK(oriclk),.rst(rst),.O_CLK(clk));
        display7 d7(.clk(clk),.rst(rst),.ge(ge),.shi(shi),.bai(bai),.an(seg_select),.seg(seg_LED));
        pos p1(.up(up),.clk(vneg),.start(start),.fail(fail),.birdy(birdy),
                .upx1(upx1),.upx2(upx2),.upy1(upy1),.upy2(upy2),
                .dnx1(dnx1),.dnx2(dnx2),.dny1(dny1),.dny2(dny2));
        vga v1(.clk(clk),.rst(rst),.hsync(hsync),.vsync(vsync),.red(red),.blue(blue),.green(green),
                .birdy(birdy),.upx1(upx1),.upx2(upx2),.upy1(upy1),.upy2(upy2),
                .dnx1(dnx1),.dnx2(dnx2),.dny1(dny1),.dny2(dny2),.vneg(vneg));
endmodule
```

2.Divider（分频器）

```
module Divider(
        input I_CLK,//原时钟
        input rst,//复位信号，低电平有效
        output reg O_CLK//分配后的时钟
        );
        parameter t=4;
        reg[32:0] count;
        initial
        begin
        count<=0;
        O_CLK<=0;
        end
```

```
    always@(posedge I_CLK)
    begin
    if(~rst)
        begin
        count<=0;
        O_CLK<=0;
        end
    else
    if(count==t/2-1)
        begin
        count<=0;
        O_CLK<=~O_CLK;
        end
    else
        count<=count+1;
    end
endmodule
```

3.display7(七段数码管显示)

```
module display7(
    input clk,
    input rst,
    input [3:0] ge,//个位数
    input [3:0] shi,//十位数
    input [3:0] bai,//百位数
    output reg [7:0] an,//位选信号
    output reg [6:0] seg//段选信号
    );
    reg [2:0]select=3'b000;//当前选中位
    reg [3:0]din;//该位数值
    wire dclk;
    Divider#(50000) d4(.I_CLK(clk),.rst(rst),.O_CLK(dclk));//500Hz
    always@(posedge dclk)
    begin
        if(~rst)
            select<=3'b000;
        else
            select<=select+3'b1;
    end
    always@(select)
    begin
    case(select)
        3'b000:begin
        an <= 8'b11111110; //选中第 1 个数码管
        din <= ge; //数码管显示的数字由 din 控制
```

```
        end
        3'b001:begin
        an <= 8'b11111101; //选中第二个数码管
        din <= shi;
        end
        3'b010:begin
        an <= 8'b11111011;
        din <= bai;
        end
        3'b011:begin
        an<=8'b11110111;
        din<=4'ha;
        end
        3'b100:begin
        an<=8'b11101111;
        din<=4'ha;
        end
        3'b101:begin
        an<=8'b11011111;
        din<=4'ha;
        end
        3'b110:begin
        an<=8'b10111111;
        din<=4'ha;
        end
        3'b111:begin
        an<=8'b01111111;
        din<=4'ha;
        end
endcase
case(din)
        4'h0: seg<= 7'b0000001; //共阳极数码管
        4'h1: seg<= 7'b1001111;
        4'h2: seg<= 7'b0010010;
        4'h3: seg<= 7'b0000110;
        4'h4: seg<= 7'b1001100;
        4'h5: seg<= 7'b0100100;
        4'h6: seg<= 7'b0100000;
        4'h7: seg<= 7'b0001111;
        4'h8: seg<= 7'b0000000;
        4'h9: seg<= 7'b0000100;
        4'ha: seg<= 7'b1111111;
endcase
end
```

```
endmodule
```
4. pos（坐标变化）
```
module pos(
    input up,//小鸟跳起信号
    input clk,
    input start,//1-游戏开始
    input fail,//游戏失败
    output reg [9:0] birdy,//同 flappybird 定义
    output reg [9:0] upx1,
    output reg [9:0] upx2,
    output reg [9:0] upy1,
    output reg [9:0] upy2,
    output [9:0] dnx1,
    output [9:0] dnx2,
    output [9:0] dny1,
    output [9:0] dny2
    );
    reg [3:0]counthigh=4'd5;//每帧移动高度
    reg [2:0]countstep=3'd0;//移动步数（帧数）
    reg [9:0]zzy[3:0]={10'd65,10'd15,10'd40,10'd85}; //水管纵坐标备选
    reg [1:0]choose=2'b00;//水管纵坐标选项
    reg [9:0]oriy=10'd263;//小鸟初始纵坐标（屏幕中央）
    reg upreg1;
    reg upreg2;
    wire upsign;//为防止连续上跳信号干扰，使用寄存器制造类似上升沿效果
                //不能直接使用 up 上升沿异步触发，因为位置改变一定是按帧频的
    reg upflag;//向上或向下的标志
    always@(posedge clk)
    begin
    if(~start)
        begin
        upreg1<=1'b0;
        upreg2<=1'b0;
        end
    else
        begin
        upreg1<=up;
        upreg2<=upreg1;
        end
    end
    assign upsign=upreg1&~upreg2;
    always@(posedge clk)
    begin
        if(~start&&~fail)
```

```
        birdy<=oriy;
    else if(~start&&fail)
        birdy<=birdy;
    else if(upsign)
        begin
        counthigh<=4'd4;
        countstep<=3'd0;
        birdy<=birdy-counthigh;
        countstep<=countstep+3'd1;
        upflag<=1'b1;
        if(birdy<10'd85)
            birdy<=10'd85;
        end
    else
        begin
        if(counthigh>0&&counthigh<=4'd8&&upflag)
            begin
            if(countstep==3'd2)
                begin
                counthigh<=counthigh-4'd2;
                countstep<=3'd0;
                end
            countstep<=countstep+3'd1;
            birdy<=birdy-counthigh;
            end
        else if(counthigh==4'd0&&upflag)
            begin
                countstep<=3'd1;
                upflag<=1'b0;
                counthigh<=4'd1;
                birdy<=birdy+counthigh;
            end
        else if(~upflag)
            begin
            if(countstep==3'd2)
                begin
                if(counthigh<4'd3)
                    counthigh<=counthigh+4'd2;
                countstep<=3'd0;
            end
            countstep<=countstep+3'd1;
            birdy<=birdy+counthigh;
            end
        if(birdy<10'd85)
```

```
                birdy<=10'd85;
            end
    end
    assign dny1=upy1+10'd250;
    assign dny2=upy2+10'd250;
    assign dnx1=upx1;
    assign dnx2=upx2;
    always@(posedge clk)
    begin
        if(~start&&~fail)
            begin
            upx1<=10'd617;
            upx2<=10'd799;
            choose=2'd0;
            upy1<=zzy[0];
            upy2<=zzy[1];
            end
        else if(~start&&fail)
            ;
        else
            begin
            upx1<=upx1-10'd1;
            upx2<=upx2-10'd1;
            if(upx1+10'd52==10'd320)
                begin
                upx1<=upx1+10'd364;
                upy1<=zzy[choose];
                choose<=choose+2'd1;
                end
            if(upx2==10'd268)
                begin
                upx2<=upx2+10'd364;
                upy2<=zzy[choose];
                choose<=choose+2'd1;
                end
            end
    end
endmodule
```

5.vga（屏幕显示输出）

```
module vga(
    input clk,//同 flappybird 模块说明
    input rst,
    output hsync,
    output vsync,
```

```
output [3:0] red,
output [3:0] green,
output [3:0] blue,
input [9:0]upx1,
input [9:0]upy1,
input [9:0]upx2,
input [9:0]upy2,
input [9:0]dnx1,
input [9:0]dny1,
input [9:0]dnx2,
input [9:0]dny2,
input [9:0]birdy,
output vneg//完成一帧扫描信号
);
reg[9:0] h_count=10'h0;//横坐标
reg[9:0] v_count=10'h0;//纵坐标
//h_count
always@(posedge clk or negedge rst)
begin
    if(~rst)
        h_count<=10'h0;
    else if(h_count==10'd799)
        h_count<=10'h0;
    else
        h_count<=h_count+10'h1;
end
//v_vount
always@(posedge clk or negedge rst)
begin
    if(~rst)
        v_count<=10'h0;
    else if(h_count==10'd799)
        begin
        if(v_count==10'd524)
            v_count<=10'h0;
        else
            v_count<=v_count+10'h1;
        end
end
assign hsync=(h_count>10'd95);
assign vsync=(v_count>10'd1);
reg[13:0]addrbt=14'h0;
wire [11:0]doutbt;
reg[9:0]addrbd=10'h0;
```

```
    wire [11:0]doutbd;
    reg[12:0]addrup=13'h0;
    wire [11:0]doutup;
    reg[12:0]addrdn=13'h0;
    wire [11:0]doutdn;
    reg[12:0]addrup2=13'h0;
    wire [11:0]doutup2;
    reg[12:0]addrdn2=13'h0;
    wire [11:0]doutdn2;
    BOTTOM bt(.addra(addrbt),.clka(clk),.douta(doutbt));
    BIRD bd(.addra(addrbd),.clka(clk),.douta(doutbd));
    ZZUP zup(.addra(addrup),.clka(clk),.douta(doutup));
    ZZDOWN zdn(.addra(addrdn),.clka(clk),.douta(doutdn));
    ZZUP2 zup2(.addra(addrup2),.clka(clk),.douta(doutup2));
    ZZDOWN2 zdn2(.addra(addrdn2),.clka(clk),.douta(doutdn2));
    //display
    reg [3:0]datar;
    reg [3:0]datag;
    reg [3:0]datab;
    reg [9:0]birdx=10'd445;//不变
    wire rdn;
    assign
rdn=~(h_count>=10'd140&&h_count<10'd780&&v_count>=10'd35&&v_count<10'd515);//640*
480
    assign red=~rdn?datar:4'h0;
    assign blue=~rdn?datab:4'h0;
    assign green=~rdn?datag:4'h0;
    always@(posedge clk or negedge rst)
    begin
    if(~rst)
    begin
        addrbt<=14'b0;
        addrbd<=10'b0;
        addrup<=13'b0;
        addrdn<=13'b0;
        addrup2<=13'b0;
        addrdn2<=13'b0;
    end
    else
    begin
        if(addrbt==14'd14250||~vsync)
            addrbt<=14'b0;
        if(addrbd==10'd816||~vsync)
            addrbd<=10'b0;
```

```
            if(addrup==13'd7800||~vsync)
                addrup<=13'b0;
            if(addrdn==13'd7800||~vsync)
                addrdn<=13'b0;
            if(addrup2==13'd7800||~vsync)
                addrup2<=13'b0;
            if(addrdn2==13'd7800||~vsync)
                addrdn2<=13'b0;
            datar<=4'h0;
            datag<=4'h0;
            datab<=4'h0;

if(h_count>=upx1&&h_count<=upx1+10'd51&&v_count>=upy1&&v_count<=upy1+10'd149)//
柱子（上）
                begin
                datar<=doutup[11:8];
                datag<=doutup[7:4];
                datab<=doutup[3:0];
                addrup<=addrup+13'b1;
                end

if(h_count>=upx2&&h_count<=upx2+10'd51&&v_count>=upy2&&v_count<=upy2+10'd149)//
柱子（上 2）
                begin
                datar<=doutup2[11:8];
                datag<=doutup2[7:4];
                datab<=doutup2[3:0];
                addrup2<=addrup2+13'b1;
                end

if(h_count>=dnx1&&h_count<=dnx1+10'd51&&v_count>=dny1&&v_count<=dny1+10'd149)//
柱子（下）
                begin
                datar<=doutdn[11:8];
                datag<=doutdn[7:4];
                datab<=doutdn[3:0];
                addrdn<=addrdn+13'b1;
                end

if(h_count>=dnx2&&h_count<=dnx2+10'd51&&v_count>=dny2&&v_count<=dny2+10'd149)//
柱子（下 2）
                begin
                datar<=doutdn2[11:8];
                datag<=doutdn2[7:4];
```

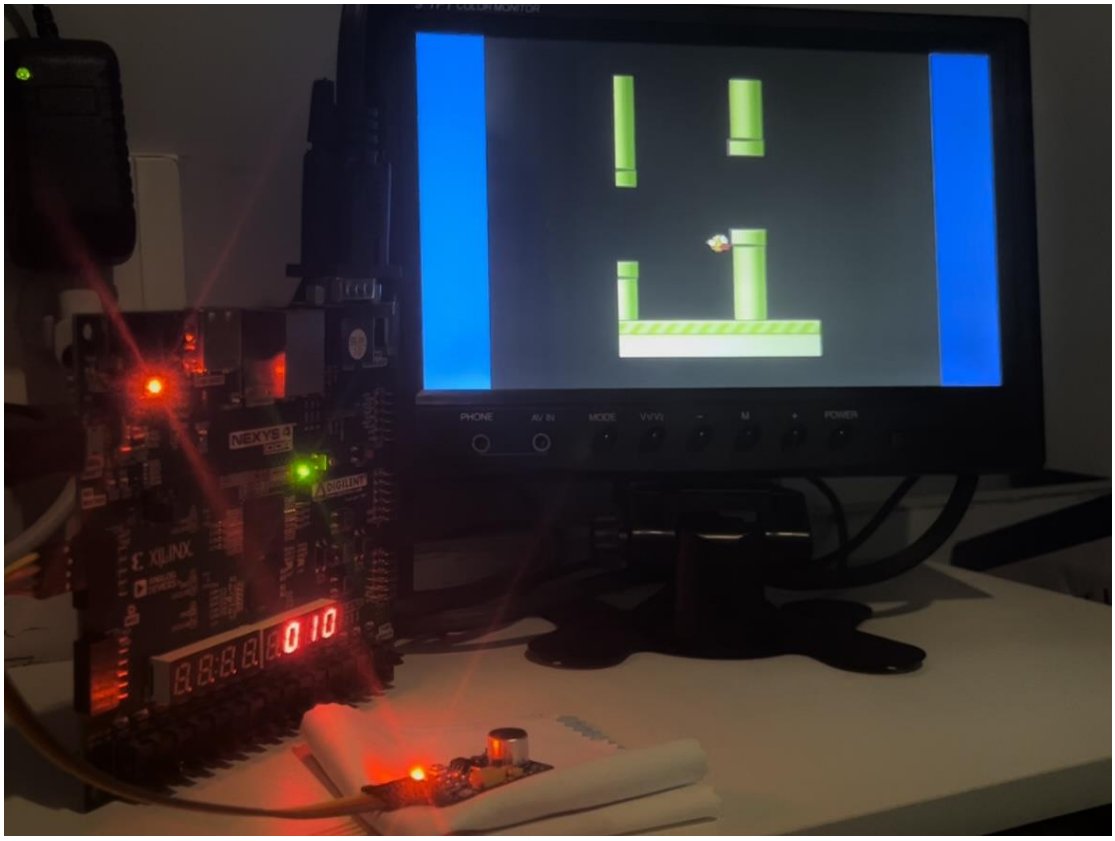
```
                datab<=doutdn2[3:0];
                addrdn2<=addrdn2+13'b1;
                end

if(h_count>=10'd320&&h_count<=10'd320+10'd284&&v_count>=10'd85+10'd330&&v_count<=10'd85+10'd379)//地面
                begin
                datar<=doutbt[11:8];
                datag<=doutbt[7:4];
                datab<=doutbt[3:0];
                addrbt<=addrbt+14'b1;
                end

if(h_count>=birdx&&h_count<=birdx+10'd33&&v_count>=birdy&&v_count<=birdy+10'd23)//鸟
                begin
                datar<=doutbd[11:8];
                datag<=doutbd[7:4];
                datab<=doutbd[3:0];
                addrbd<=addrbd+10'b1;
                end

if(h_count>=10'd320&&h_count<=10'd320+10'd284&&v_count>=10'd85&&v_count<=10'd85+10'd379)//游戏区域
                ;
        else
                begin
                datar<=4'h0;
                datag<=4'h0;
                datab<=4'h0;
                end
    end
    end
    reg vreg1;
    reg vreg2;
    always@(posedge clk or negedge rst)
    begin
    if(~rst)
        begin
        vreg1<=1'b0;
        vreg2<=1'b0;
        end
    else
        begin
```

```
            vreg1<=vsync;
            vreg2<=vreg1;
            end
        end
        assign vneg=~vreg1&vreg2;
endmodule
```

五、测试模块建模

```
module test();
        reg oriclk;
        reg rst;
        reg up;
        reg btn;
        wire hsync;
        wire [3:0] blue;
        wire [3:0] red;
        wire [3:0] green;
        wire vsync;
        wire [3:0] seg_select;
        wire [6:0] seg_LED;
        initial
        begin
            oriclk=0;
            rst=1;
            btn=0;
        forever
        begin
            #5 oriclk=~oriclk;
        end
        end
        initial
        begin
                up=0;
                #20000000 up=1;
                //#5 up=0;
end
        flappybird
uut(.oriclk(oriclk),.rst(rst),.up(up),.btn(btn),.hsync(hsync),.blue(blue),.red(red),.green(green),.vsyn
c(vsync),.seg_select(seg_select),.seg_LED(seg_LED));
endmodule
```

六、实验结果



七、结论

基本还原了 flappy bird 的功能和玩法，声控较之按键/触控移动，多了一些不稳定性，增加了一些游戏难度，因此在小鸟移动的加速度和初速度上做了减法，以此平衡整体难度。

制作过程中遇到一些问题，现总结如下：

第一，从 ROM 中读取图片是地址指针混乱导致图片无法稳定显示，出现波动现象，图片像素地址不易确定，因此我认为最佳的解决方法是在期待有所显示的区域使指针稳定加 1，无论是否会被更上方的图片遮挡。

第二，图片无法正确移动，由于显示的画面总是按帧频改变，如果改变图片坐标的模块所用时钟无法与帧频较好匹配，则图片移动时会出现问题，位置跳变异常等，因此建议直接将帧频作为时钟输入。

第三，七段数码管的显示异常问题，主要是由于时钟过快或过慢，过快导致数码管无法正确显示，过慢则会出现一闪一闪的现象，七段数码管的时钟以 60-1kHz 为宜。

八、心得体会及建议

不同于 c++等高级语言，硬件语言编写的数字系统设计中，很多问题都是时序错误导致，尊重硬件特性是学习硬件时非常重要的一点。

本次实验由于时间紧迫，没有实现上升/下降时小鸟形态的变化，处于美观考虑（避免边角黑色部分过于明显），也没有还原原蓝天背景，但是目前的成品已是本人在综合考虑下制作出的最优选。