同济大学计算机系

数字逻辑课程综合实验报告

学　　号 1952650

姓　　名 陈子翔

专　　业 信息安全

授课老师 张冬冬

目录

一、实验内容

1. 项目内容概括

基于 FPGA 开发板设计的一款类似 flappy bird 的小游戏：flappy mario，并在 VGA 上显示；同时可通过蓝牙连接手机操控。

2. 界面样式

如图所示，柱子源源不断从左往右移动，玩家需要通过控制上下左右移动马里奥使得马里奥顺利通过每根柱子的通道，得到分数；如果撞到柱子则游戏结束。

3. 游戏规则

游戏开始后，可选择两种控制马里奥位置的方式：一种通过按下设置的上下左右四个管脚以及加速管脚控制马里奥速度和位置；一种通过蓝牙连接手机，用手机控制马里奥运动方向。Vga 屏幕上会有源源不断的带有通道的柱子从左向右移动，玩家需控制马里奥顺利通过

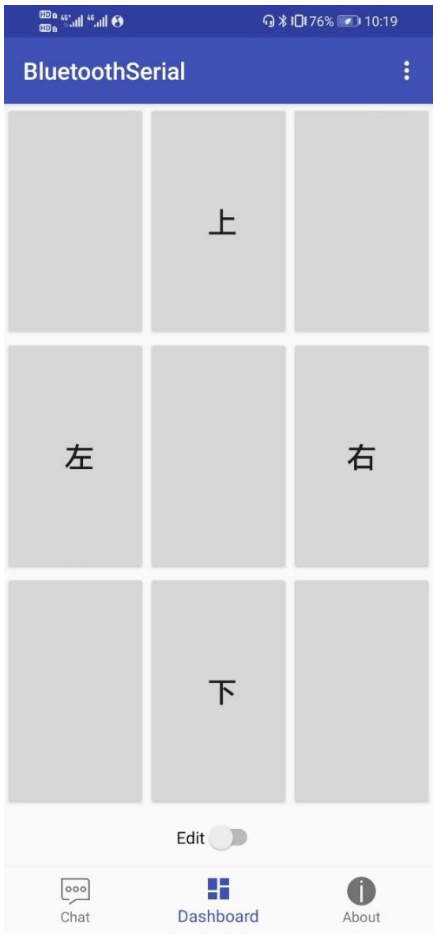
每根柱子的通道即可得分，如若撞到柱子其他部分，则游戏结束。

4. 按键设置

① 管脚

开关编号	用途说明
(rst) J15	游戏复位键，为 1 时重置游戏
(up) M18	向上移动，按下后马里奥上移
(down) P18	向下移动，按下后马里奥下移
(left) P17	向左移动，按下后马里奥左移
(right) M17	向右移动，按下后马里奥右移
(start) V10	开始键，按下后进入游戏界面
(speed_up) N17	加速键，按下后马里奥速度加快
(level2) U11	普通模式
(level3) U12	困难模式

② 蓝牙

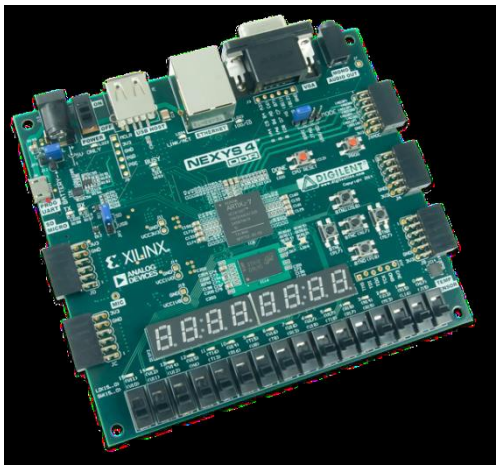


通过手机屏幕显示的上下左右按键控制马里奥运动。

5. 所用器件

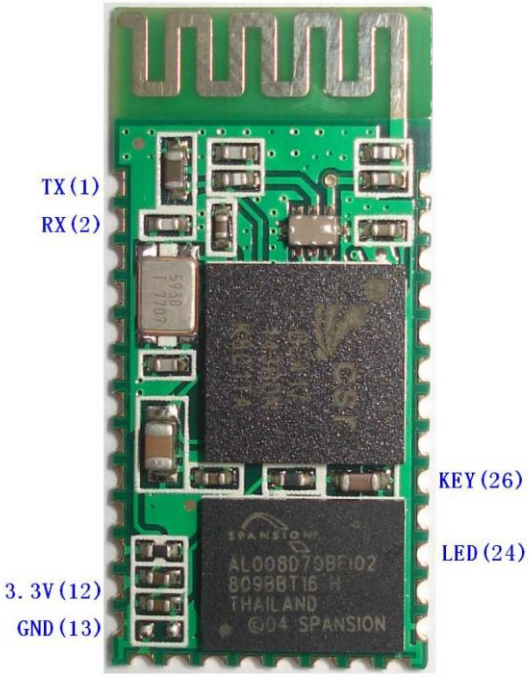
① FPGA 开发板：Nexys4 DDR™ FPGA Board

由 Xilinx 公司开发出的一款可编程门阵列（FPGA）开发板。



② VGA PS2 Board：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。

③ 蓝牙：基于 UART 接口的蓝牙从模块



二、Flappy mario 游戏数字系统总框图

（按由顶向下方法进行子系统的划分，给出包含各子系统相互关系及控制信号的总框图，并对各子系统功能及实现进行概述。具体可参考教材 183 页的相关描述方法。）

① 控制器顶层系统：由此顶层系统统一控制各子系统，定义所需所有管脚及相应变量，传入调用各子系统：调用蓝牙子模块连接手机传输系统；调用七段数码管显示模块将倒计时及得分显示在七段数码管上；调用倒计时模块，传出此时倒计时剩余时间；调用 vga 显示模块，将游戏各界面通过状态机显示。

② 输入子系统 1：由游戏开始按键 START 组成，按下此按键后 VGA 由游戏准备界面跳转至游戏开始界面。与此同时，倒计时重置 100s，计分清零重置。

③ 输入子系统 2：由游戏复位键 rst 组成，按下此键后游戏结束，vga 停止扫描，七段数码管停止显示，直至按回此键游戏方可继续。

④ 输入子系统 3：由游戏控制键（管脚）组成，玩家可通过操作 FPGA 开发板上的相应管脚对马里奥进行上下左右操作以及加速操作。

⑤ 输入子系统 4：由游戏控制键（手机蓝牙）组成，玩家可在手机连接蓝牙模块后通过蓝牙协议将手机的输入转换为 ASCII 传回进行分析，通过手机设置的上下左右按键对马里奥进行操作，更具有游戏性。

⑥ 输入子系统 5：选择游戏难度，可选择简单、普通以及困难模式。

⑦ 输出子系统 1：VGA 显示器输出，通过控制器顶层系统对 VGA 时序扫描控制；通过 RGB 计算系统在屏幕上不同区域显示相应颜色内容，同时相应游戏控制键对马里奥位置实时控制。

⑧ 输出子系统 2：七段数码管显示输出，在顶层模块向此子模块传入当前倒

计时以及当前得分，通过数码管位控决定显示哪一位数码管的数据，通过七段译码管显示具体数值。由此实现时间与得分的实时可视化。

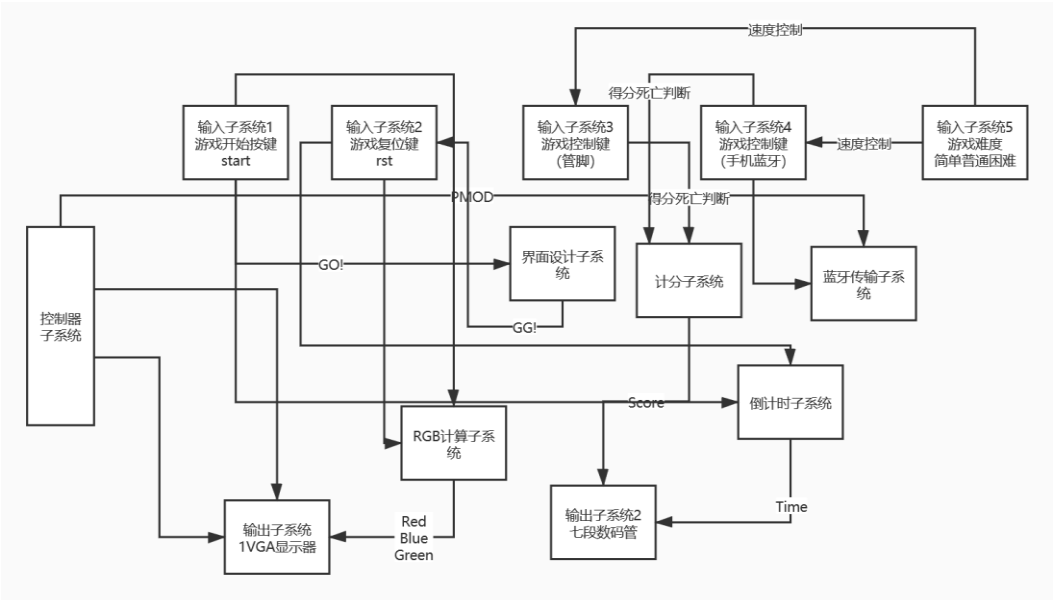
⑨ 计分子系统：此子系统由加法器组成，通过对马里奥位置实时判断，如若顺利通过柱子通道则得分增加，若游戏结束则得分清零。

⑩ 界面设计子系统：根据游戏逻辑，分为三个不同状态——开始准备界面、正式游戏界面以及游戏结束提示界面。通过一个统一的 flag 进行控制：flag 为 0 时 vga 显示开始准备界面，flag 为 1 时 vga 显示正式游戏界面，flag 为 2 时显示游戏结束提示界面。

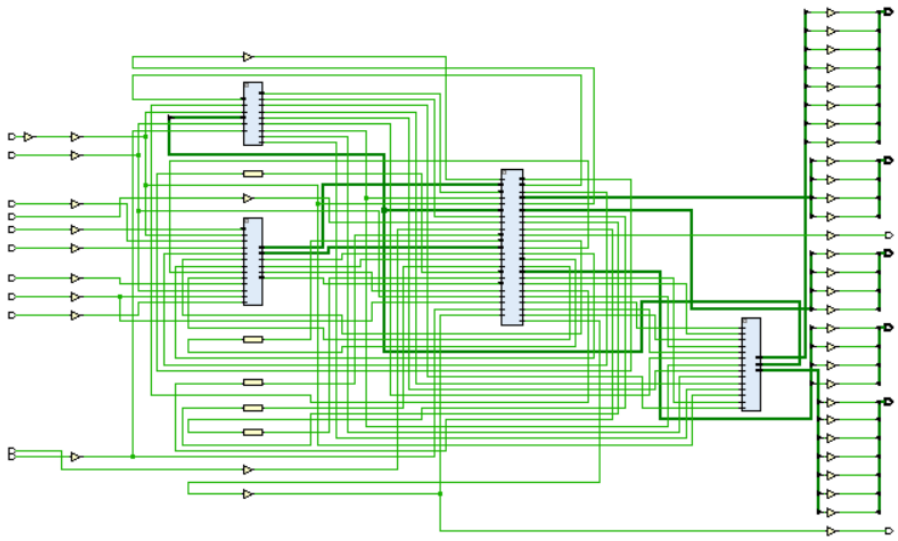
⑪ 倒计时子系统：此子系统用到了分频器将时钟分频，实现倒计时。

⑫ 蓝牙传输子系统：此子系统通过对蓝牙协议运用，对波特率进行设置实现手机传输数据功能，通过手机输入数据，将数据 ASCII 值传回程序进行分析。

总程序总框图如下：



总程序 RTL 图片如下图：



三、系统控制器设计

（要求画出所设计数字系统的 ASM 流程图，列出状态转移真值表。由状态转移真值表，求出系统控制器的次态激励函数表达式和控制命令逻辑表达式，并用 Logisim 画出系统控制器逻辑方案图。）

① 马里奥运动逻辑

PS	NS	转换条件
开始(0000)	马里奥自动下移(0001)	/
马里奥自动下移(0001)	马里奥自动下移(0001)	没有进行按键操作($\neg X$)
马里奥自动下移(0001)	马里奥移动(0010)	进行了手机/管脚按键操作(X)
马里奥移动(0010)	马里奥反复撞墙静止(0100)	马里奥到达边界(Y)
马里奥移动(0010)	马里奥移动(0010)	未到达边界且未撞到柱子
马里奥移动(0010)	马里奥死亡(1000)	马里奥撞到柱子(Z)
马里奥死亡(1000)	游戏结束(1111)	/
游戏结束(1111)	开始(0000)	按下按键 START(M)

令 X：有按键操作时为 1，无按键操作时为 0

令 Y：马里奥到达边界时为 1，未到达边界时为 0

令 Z：马里奥撞上柱子时为 1，未撞上柱子时为 0

令 M：按下按键 START 时为 1，未按下按键 START 时为 0

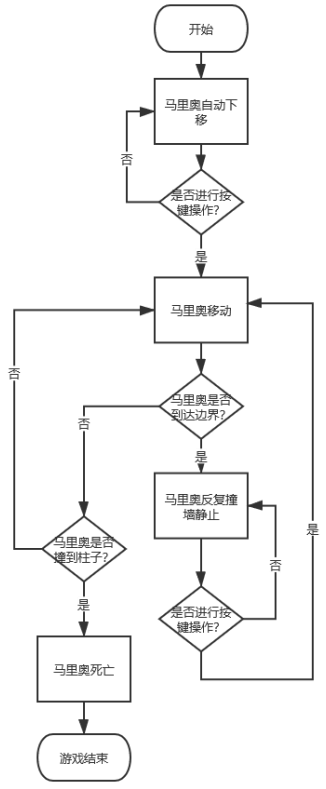
用四个触发器 DCBA 控制状态：

$$D^{n+1} = ZB^n + D^n$$

$$C^{n+1} = YB^n + D^n\neg C^n$$

$$B^{n+1} = XA^n + B^n\neg Y\neg Z$$

$$A^{n+1} = \neg D^n\neg C^n\neg B^n\neg A^n + \neg XA^n + D^n$$



② 游戏逻辑

PS	NS	转换条件
开始(000000)	初始化(000001)	/
初始化(000001)	马里奥运动(000010)	/
马里奥运动(000010)	游戏结束(100000)	马里奥撞到柱子(X)
马里奥运动(000010)	检测倒计时(000100)	马里奥未撞到柱子($\neg X$)
检测倒计时(000100)	游戏结束(100000)	倒计时结束(Y)
检测倒计时(000100)	检测计分模块(001000)	倒计时未结束($\neg Y$)
检测计分模块(001000)	加分(010000)	马里奥通过柱子(Z)
检测计分模块(001000)	马里奥运动(000010)	马里奥未通过柱子($\neg Z$)
加分(010000)	马里奥运动(000010)	/
游戏结束(100000)	游戏结束(100000)	未按下 START 开始键($\neg M$)
游戏结束(100000)	开始(000000)	按下 START 开始键(M)

令 X：马里奥撞到柱子时为 1，未撞到柱子时为 0

令 Y：倒计时结束时为 1，倒计时未结束时为 0

令 Z：马里奥通过柱子时为 1，未通过柱子时为 0

令 M：按下按键 START 时为 1，未按下按键 START 时为 0

用六个触发器 FEDCBA 控制状态：

$$F^{n+1} = B^n X + C^n Y + F^n \neg M$$

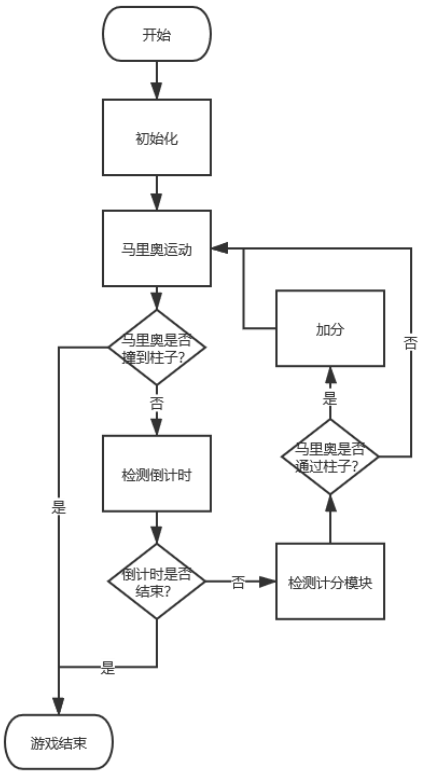
$$E^{n+1} = D^n Z$$

$$D^{n+1} = C^n \neg Y$$

$$C^{n+1} = B^n \neg X$$

$$B^{n+1} = A^n + D^n \neg Z + E^n$$

$$A^{n+1} = \neg F^n \neg E^n \neg D^n \neg C^n \neg B^n \neg A^n$$



③ VGA

PS	NS	转换条件
开始(0001)	行扫描(0010)	/
行扫描(0010)	行扫描(0010)	行扫描未结束
行扫描(0010)	列扫描(0100)	行扫描结束
列扫描(0100)	行扫描(0010)	扫描未达到最后一行
列扫描(0100)	扫描结束(1000)	扫描达到最后一行
扫描结束(1000)	开始(0001)	/

令 X：行扫描结束时为 1，未结束时为 0

令 Y：扫描达到最后一行时为 1，扫描未达到最后一行时为 0

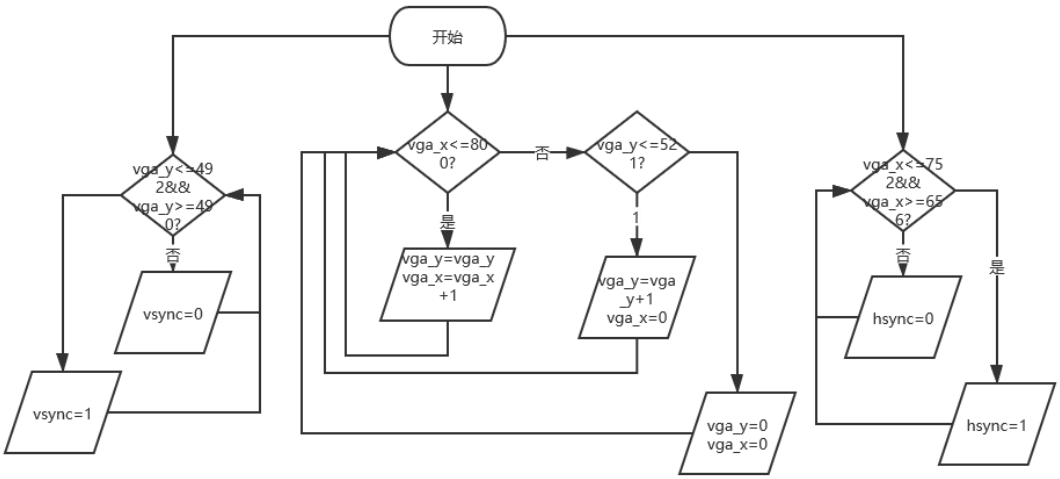
采用一对一法对状态进行编码，有四个不同状态，故采用四个触发器 DCBA 控制状态：

$$D^{n+1} = YC^n$$

$$C^{n+1} = XB^n$$

$$B^{n+1} = A^n + \neg XB^n + \neg YC^n$$

$$A^{n+1} = D^n$$



④ 界面设计

PS	NS	转换条件
开始(0001)	开始界面 GO!(0010)	/
开始界面 GO! (0010)	进入游戏界面(0100)	按下 START 键
开始界面 GO! (0010)	开始界面 GO! (0010)	未按下 START 键
进入游戏界面(0100)	游戏结束界面 GG!(1000)	达到游戏结束条件
进入游戏界面(0100)	进入游戏界面(0100)	未达到游戏结束条件
游戏结束界面 GG! (1000)	进入游戏界面(0100)	按下 START 键
游戏结束界面 GG! (1000)	游戏结束界面 GG! (1000)	未按下 START 键

令 X：按下按键 START 时为 1，未按下按键 START 时为 0

令 Y：达到游戏结束条件时为 1，未达到游戏结束条件时为 0

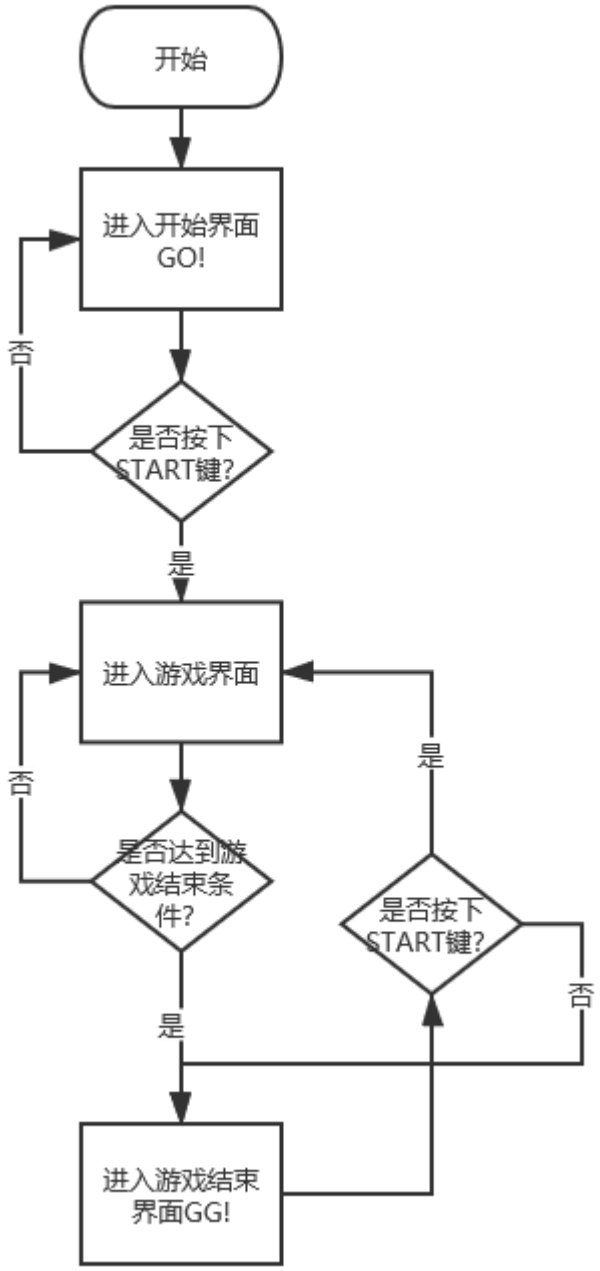
采用一对一法对状态进行编码，有四个不同状态，故采用四个触发器 DCBA 控制状态：

$$D^{n+1} = YC^n + \neg XD^n$$

$$C^{n+1} = XB^n + \neg YC^n + XD^n$$

$$B^{n+1} = A^n + \neg XB^n + \neg YC^n$$

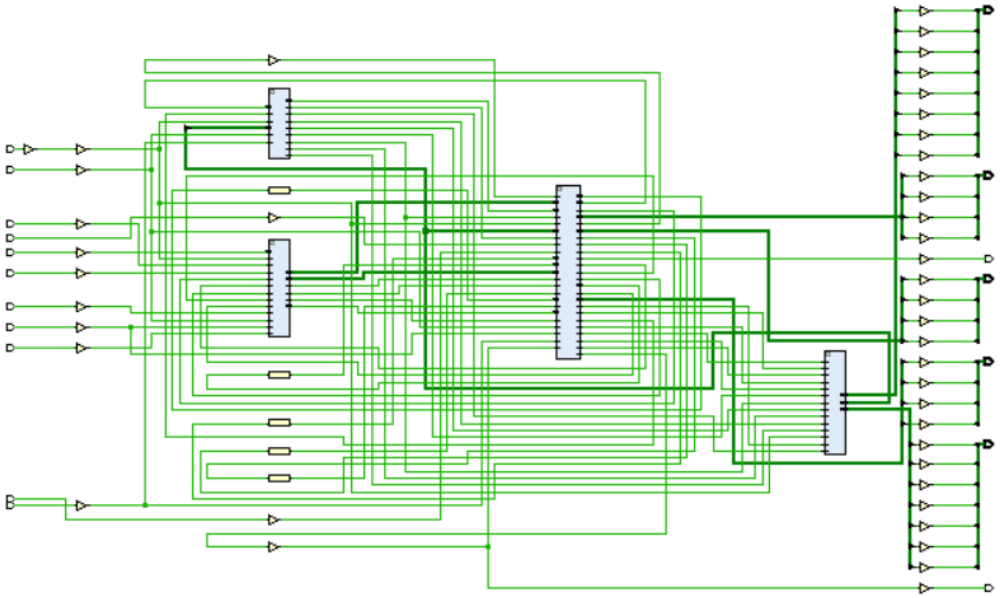
$$A^{n+1} = \neg D^n \neg C^n \neg B^n \neg A^n$$



四、子系统模块建模

（该部分要求对实验中的所有子系统模块进行描述，给出各子系统的功能框图及接口信号定义，并列出各模块建模的 verilog 代码。

1. *Top* 顶层模块

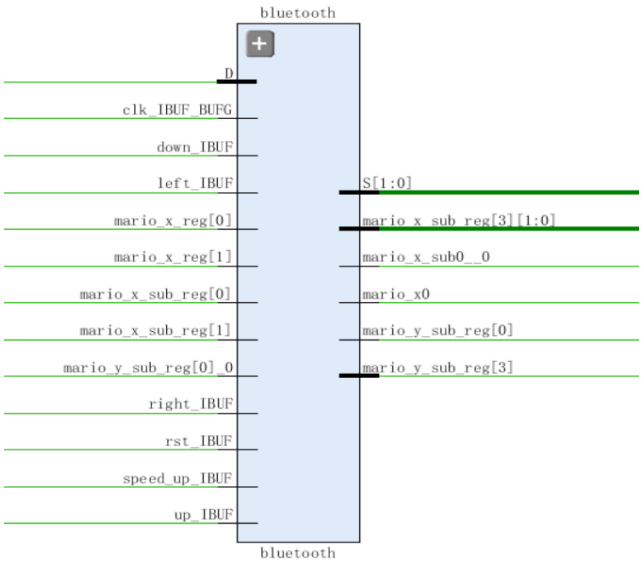


顶层模块，总体操控游戏，连接各个子模块，设置所有需要用到的管脚。传入调用各子系统：调用蓝牙子模块连接手机传输系统；调用七段数码管显示模块将倒计时及得分显示在七段数码管上；调用倒计时模块，传出此时倒计时剩余时间；调用 vga 显示模块，将游戏各界面通过状态机显示。

```
module Top(
    input rst,                  //游戏复位键
    input clk,                  //系统时钟
    output hsync,vsync,         //行时序、场时序
    output [3:0] red,           //VGA 红色分量
    output [3:0] blue,          //VGA 蓝色分量
    output [3:0] green,         //VGA 绿色分量
    output [6:0] seg_led,       //七段数码管显示数据
    output [7:0] AN,            //七段数码管选择
    input get,                  //连接蓝牙 PMOD
    input up,                   //控制马里奥上移
    input down,                 //控制马里奥下移
    input left,                 //控制马里奥左移
    input right,                //控制马里奥右移
    input start,                //游戏开始按钮
    input speed_up,             //加速按钮
    input level2,               //普通模式
    input level3                //困难模式
);

wire [7:0] out;                 //蓝牙输出

//数码管
reg [15:0] count_clk;
reg CLK_1000HZ=1'b0;
```

```
reg [3:0]data[0:7];
wire [6:0] Score;
wire [6:0] Time;
reg [6:0] Data;
reg [2:0] display_count;

wire live;

//蓝牙模块
bluetooth bluetooth(.clk(clk),.rst(rst),.get(get),.out(out));

//七段数码管显示模块
display7 Display(.clk(clk),.rst(rst),.Score(Score),.Time(Time),.seg_led
(seg_led),.AN(AN));

//倒计时模块
timecount Timecount(.clk(clk),.rst(rst),.start(start),.live(live),.Time
(Time));

//vga 显示模块
vga_display vga_display(.rst(rst),.clk(clk),.hsync(hsync),.vsync(vsync)
,.red(red),.blue(blue),.green(green),.up(up),.down(down),.left(left),.r
ight(right),.start(start),.speed_up(speed_up),.level2(level2),.level3(l
evel3),.Score(Score),.out(out));

endmodule
```

2. *Bluetooth* 子模块



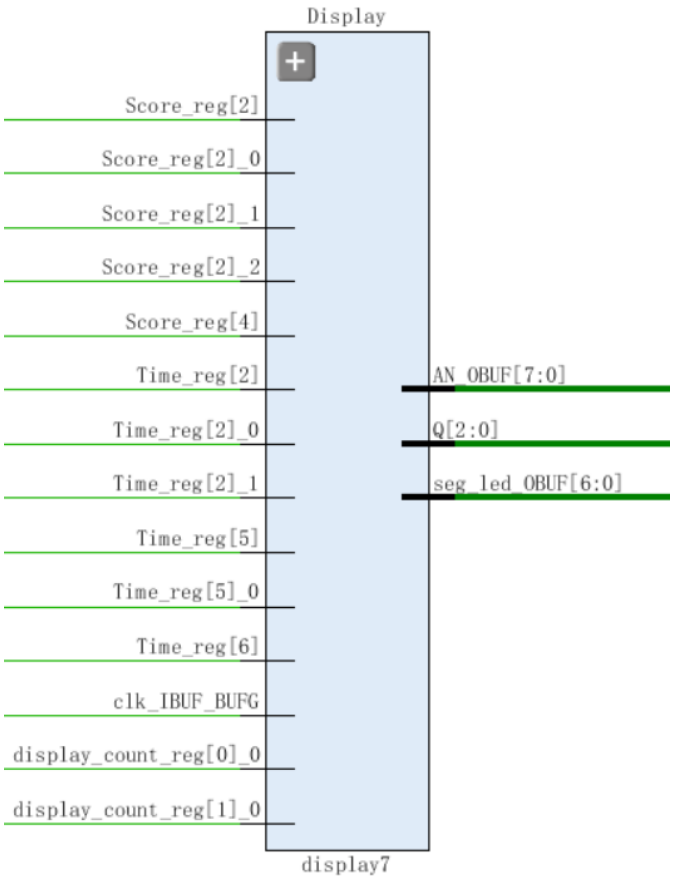
蓝牙子模块：此子系统通过对蓝牙协议运用，对波特率进行设置实现手机传输数据功能，通过手机输入数据，将数据 ASCII 值传回程序进行分析。

```
module bluetooth(
    input clk,
    input rst,
    input get,
    output reg [7:0] out
);
    parameter bps=10417;//对应 9600 波特率
    reg [14:0] count_1;//每一位中的计数器
    reg [3:0] count_2;//每一组数据的计数器
    reg buffer_0,buffer_1,buffer_2;//除去滤波
    wire buffer_en;//检测到边沿
    reg add_en;//加法使能信号

    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<=0;
        end
        else if(add_en&&count_1==bps/2-1&&count_2!=0)
        begin
            out[count_2-1]<=get;
        end
    end

endmodule
```

3. *Display7*子模块



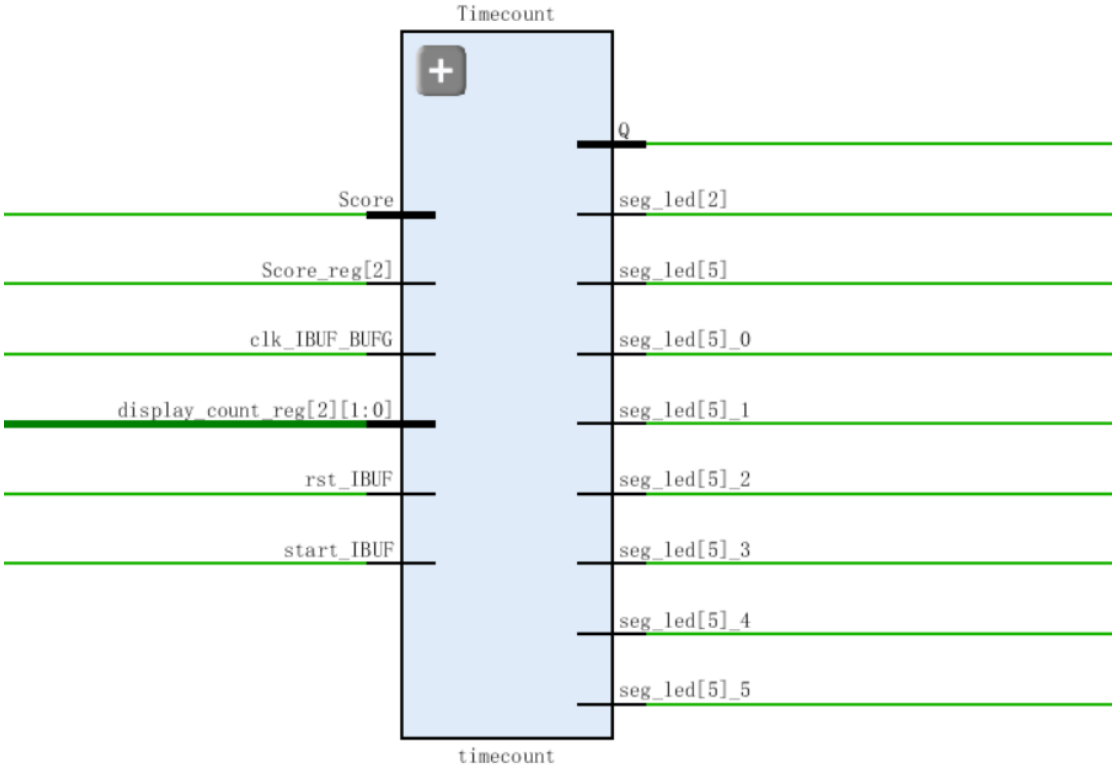
七段数码管子模块：七段数码管显示输出，在顶层模块向此子模块传入当前倒计时以及当前得分，通过数码管位控决定显示哪一位数码管的数据，通过七段译码管显示具体数值。由此实现时间与得分的实时可视化。

```
module display7(
    input clk,
    input rst,
    input [6:0] Score,          //得分
```

```
    input [6:0] Time,            //倒计时
    output reg [6:0] seg_led,
    output reg [7:0] AN          //位控
);

reg [3:0] data[0:7];
reg [6:0] Data;

reg CLK_1000HZ=1'b0;
reg [15:0] count_clk;

reg [2:0] display_count;

//1000HZ 分频
always@(posedge clk)
    begin
        if(count_clk==16'd49999)
        begin
        count_clk<=16'd0;
        CLK_1000HZ<=~CLK_1000HZ;
        end

        else
        count_clk<=count_clk+1;
    end
//得到各个数码管应显示的数字
always@(*)
begin
Data = Score;
data[0] = Data % 10;
Data = Data / 10;
data[1] = Data % 10;
Data = Data / 10;
data[2] = Data % 10;
Data = Data / 10;
data[3] = Data % 10;

Data = Time;
data[4] = Data % 10;
Data = Data / 10;
data[5] = Data % 10;
Data = Data / 10;
data[6] = Data % 10;
Data = Data / 10;
```

```
data[7] = Data % 10;
Data = Data / 10;
end

//位控
always@ (posedge CLK_1000HZ)
begin
if(display_count == 3'd7)
display_count = 0;
else
display_count = display_count + 1;
case(display_count)
3'd0: AN = 8'b11111110;
3'd1: AN = 8'b11111101;
3'd2: AN = 8'b11111011;
3'd3: AN = 8'b11110111;
3'd4: AN = 8'b11101111;
3'd5: AN = 8'b11011111;
3'd6: AN = 8'b10111111;
3'd7: AN = 8'b01111111;
default: AN = 8'b11111111;
endcase
end

//译码
always@(*) begin
case(data[display_count])
0: seg_led = 7'b1000000;         //0
1: seg_led = 7'b1111001;         //1
2: seg_led = 7'b0100100;         //2
3: seg_led = 7'b0110000;         //3
4: seg_led = 7'b0011001;         //4
5: seg_led = 7'b0010010;         //5
6: seg_led = 7'b0000010;         //6
7: seg_led = 7'b1111000;         //7
8: seg_led = 7'b0000000;         //8
9: seg_led = 7'b0010000;         //9
default seg_led=7'b0;
endcase
end


endmodule
```

4. *timecount*子模块

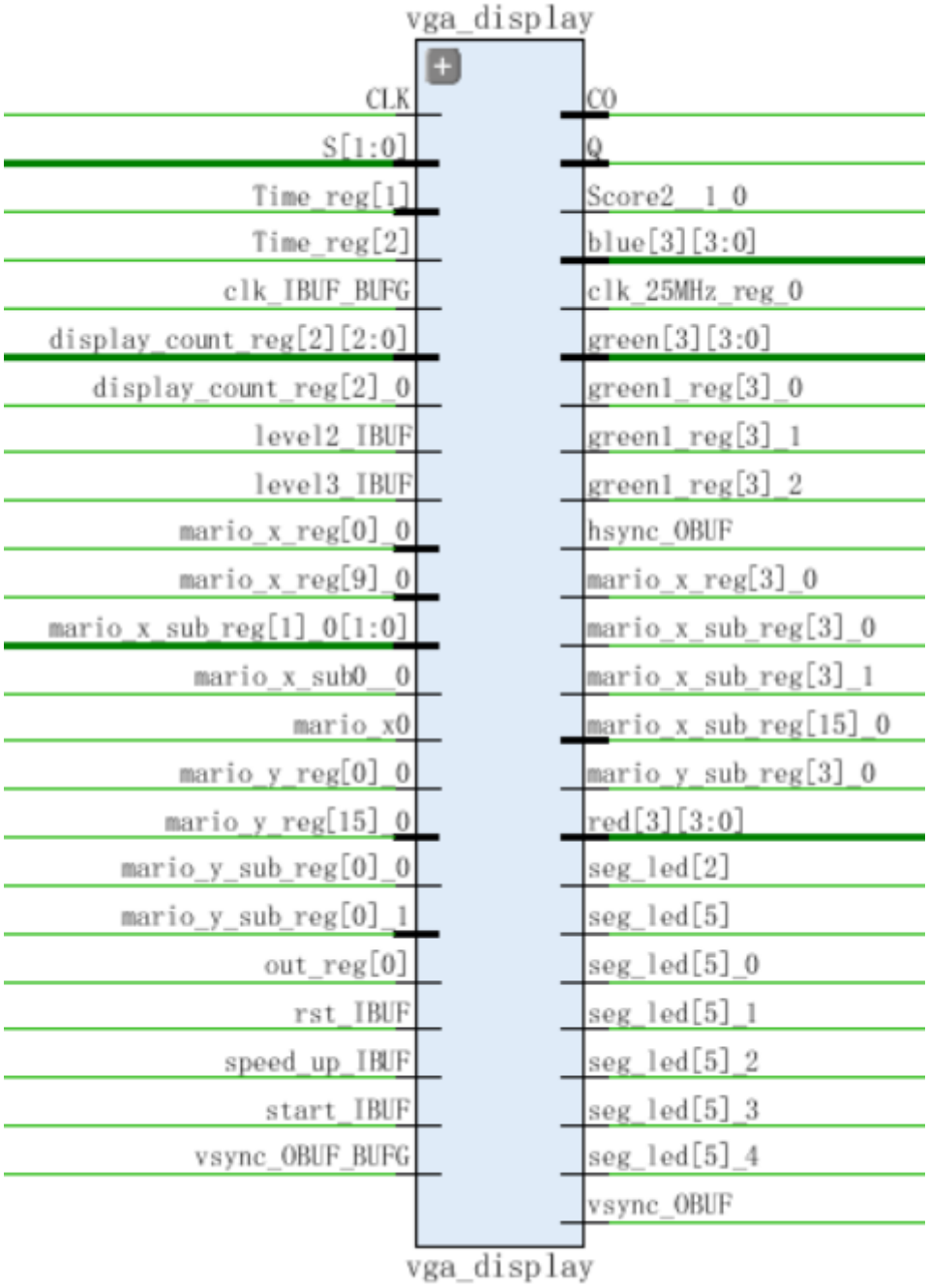


倒计时子模块：此子系统用到了分频器将时钟分频，实现倒计时。

```
module timecount(
    input clk,
    input rst,
    input start,
    output reg live,              //传出是否存活
    output reg [6:0] Time         //传出剩余时间
);

integer counter = 0;
reg out = 0;
always@(posedge clk)
begin
    if (rst == 1'b1||start)       //初始化
    begin
        counter = 0;
        out <= 0;
    end
    else
    begin
        if (counter >= 50000000)
```

```
        begin
            counter = 0;
            out <= ~out;
        end
        else
        begin
            counter = counter + 1;
            out <= out;
        end
    end
end

always@(posedge out or negedge rst)

begin
    if (rst == 1'b1||start)          //初始化
    begin
        Time <= 100;
        live <= 1;
    end
    else
        begin
        if (Time > 0)
        begin
            Time <= Time - 1;
            live <= 1;
        end

        else
        begin
            Time <= 0;
            live <= 0;
        end
    end

end
endmodule
```

5. *vga_display* 子模块



Vga_display 子模块：VGA 显示器输出，通过控制器顶层系统对 VGA 时序扫描控制；通过 RGB 计算系统在屏幕上不同区域显示相应颜色内容，同时相应游戏控制键对马里奥位置实时控制。

```
module vga_display(
    input rst,
    input clk,
    output reg hsync,vsync,
    output reg [3:0] red,
    output reg [3:0] blue,
    output reg [3:0] green,
    input up,
```

```
    input down,
    input left,
    input right,
    input start,
    input speed_up,
    input level2,
    input level3,
    output reg [6:0] Score,
    input [7:0] out
);

reg [3:0] red0;
reg [3:0] green0;
reg [3:0] blue0;
reg [3:0] red1;
reg [3:0] green1;
reg [3:0] blue1;
reg [3:0] red2;
reg [3:0] green2;
reg [3:0] blue2;
reg [1:0] flag;

reg [9:0] vga_x = 0; // 800 TS , x
reg [9:0] vga_y = 0; // 521 Ts , y
reg clk_25MHz = 0;
reg [1:0] count = 0;

//马里奥：坐标：上：170+mario_y-mario_y_sub 下：202+mario_y-mario_y_sub 左：
558+mario_x-mario_x_sub 右：590+mario_x-mario_x_sub
reg [15:0] mario_x = 0;
reg [15:0] mario_y = 0;
reg [15:0] mario_x_sub = 0;
reg [15:0] mario_y_sub = 0;

reg [9:0] path_count=0;
reg [9:0] column_path=0;
reg [9:0] column_path1=0;
reg [9:0] column_path2=0;
reg [9:0] column_path3=0;

//数码管
reg [15:0] count_clk;
reg CLK_1000HZ=1'b0;
reg [3:0]data[0:7];
```

```
wire [6:0] Time;
reg [6:0] Data;
reg [2:0] display_count;

wire live;


//得分模块
always@(posedge vsync, negedge rst)
if(rst==1||flag==2'd0||flag==2'd2)
begin
    Score<=0;
    //Time<=0;
end
else
begin
    if(((574+mario_x-mario_x_sub-column_path-15)*(574+mario_x-mario_x_s
ub-column_path-15)<2)&&((170+mario_y-mario_y_sub>50)&&(202+mario_y-mari
o_y_sub<120)))
        begin
        Score<=Score+1;
        //Time<=Time+1;
        end
    else if(((574+mario_x-mario_x_sub-column_path2-15)*(574+mario_x-mar
io_x_sub-column_path2-15)<2)&&((170+mario_y-mario_y_sub>180)&&(202+mari
o_y-mario_y_sub<250)))
        begin
        Score<=Score+1;
        //Time<=Time+1;
        end
    else
        begin
        Score<=Score;
        //Time<=Time;
        end
end

//马里奥：坐标：上：170+mario_y-mario_y_sub 下：202+mario_y-mario_y_sub 左：
558+mario_x-mario_x_sub 右：590+mario_x-mario_x_sub
always @(posedge vsync)
begin
    if(rst)
        flag<=2'd0;
    else if(start)
        flag<=2'd1;
```

```
    else if(((574+mario_x-mario_x_sub-column_path3-15)*(574+mario_x-mar
io_x_sub-column_path3-15)<36)&&((170+mario_y-mario_y_sub<100)||(170+mar
io_y-mario_y_sub>170)))
        flag<=2'd2;
    else
        flag<=flag;
end

//分频 25M Hz
always@(posedge clk)begin
        if(rst)begin
            count<=0;
        end
        else begin
            if(count < 1)begin
            count <= count + 1;
            end
        else
        begin
            clk_25MHz <=~clk_25MHz;
            count<= 0;
        end
    end
end

always @ (posedge clk_25MHz or posedge rst) begin
  if(rst) begin
    vga_x <= 0;
    vga_y <= 0;
  end
  else begin
  //根据 basic VGA controller 的电路图，在水平方向扫描一次后，使得垂直方向开始
扫描
  // 因为水平方向是时钟计数的，垂直方向是根据水平方向的脉冲计数的
    if(vga_x >= 800)
    begin
        vga_x <= 0;
        if(vga_y>=521)
            vga_y<=0;
        else
            vga_y<=vga_y+1'b1;
        vga_y <= (vga_y >= 521? 0 : vga_y + 1'b1);
    end
    else
```

```
        vga_x <= vga_x + 1'b1;
  end
end

//设置行选，列选信号有效。 由于有建立的 Tpw 时间，所以要把 Tpw(脉冲宽度）时间段内的坐标视为无效
always @(posedge clk_25MHz or posedge rst) begin
if(rst)
begin
    hsync <= 0;
    vsync <= 0;
end
else
begin
    if(vga_x < 752 &&vga_x  >= 656)// 脉冲内为0 (800 - Tbp -Tpw) ~ (800 - Tbp)
        hsync <= 0;
    else
        hsync <= 1;

    if(vga_y < 492 && vga_y >= 490 )// 脉冲内为0 (521 - Tbp -Tpw) ~ (521 - Tbp)
        vsync <= 0;
    else
        vsync <= 1;
end
end

//步数移动模块
always @(posedge vsync)
begin
    if(rst)
    begin
    mario_x<=0;
    mario_y<=0;
    mario_x_sub<=0;
    mario_y_sub<=0;
    end

    else if(flag==2'd1)
    begin
        if(up||out==49)
        begin
            mario_x<=mario_x;
```

```
    mario_y<=mario_y;
    mario_x_sub<=mario_x_sub;
    if(speed_up)
        mario_y_sub<=mario_y_sub+10'd5;
    else if(out==49)
        mario_y_sub<=mario_y_sub+10'd2;
    else
        mario_y_sub<=mario_y_sub+10'd3;
end

if(down||out==50)
begin
    mario_x<=mario_x;
    if(speed_up)
        mario_y<=mario_y+10'd5;
    else if(out==50)
        mario_y<=mario_y+10'd1;
    else
        mario_y<=mario_y+10'd3;
    mario_x_sub<=mario_x_sub;
    mario_y_sub<=mario_y_sub;
end

if(left||out==51)
    begin
    mario_x<=mario_x;
    mario_y<=mario_y;
    if(speed_up)
        mario_x_sub<=mario_x_sub+10'd4;
    else if(out==51)
        mario_x_sub<=mario_x_sub+10'd1;
    else
        mario_x_sub<=mario_x_sub+10'd2;
    mario_y_sub<=mario_y_sub;
    end

if(right||out==52)
    begin
    if(speed_up)
        mario_x<=mario_x+10'd4;
    else if(out==52)
        mario_x<=mario_x+10'd1;
    else
        mario_x<=mario_x+10'd2;
```

```
            mario_y<=mario_y;
            mario_x_sub<=mario_x_sub;
            mario_y_sub<=mario_y_sub;

        end
        mario_y<=mario_y+1;
        //马里奥：坐标：上：170+mario_y-mario_y_sub 下：
202+mario_y-mario_y_sub 左：558+mario_x-mario_x_sub 右：
590+mario_x-mario_x_sub
        if(170+mario_y-mario_y_sub<5)
            mario_y<=mario_y+10;
        else if(202+mario_y-mario_y_sub>460)
            mario_y_sub<=mario_y_sub+10;
        else if(558+mario_x-mario_x_sub<50)
            mario_x<=mario_x+10;
        else if(590+mario_x-mario_x_sub>700)
            mario_x_sub<=mario_x_sub+10;
    end
    else
    begin
    mario_x<=0;
    mario_y<=0;
    mario_x_sub<=0;
    mario_y_sub<=0;
    end

end

always @ (posedge vsync)
begin
if(rst)
    path_count<=0;
else if(flag==2'd0||flag==2'd2)
    path_count<=0;
else
begin
    if(path_count<700)
        begin
        if(level2)
            path_count<=path_count+2;
        else if(level3)
            path_count<=path_count+3;
        else
            path_count<=path_count+1;
```

```
        end
    else
        path_count<=path_count;
end
end

always @ (posedge vsync)
begin
if(rst)
    column_path<=0;
else if(flag==0||flag==2)
begin
    column_path<=0;
end
else
begin
    if(column_path>640)
        column_path<=0;
    else
        begin
        if(level2)
            column_path<=column_path+2;
        else if(level3)
            column_path<=column_path+3;
        else
            column_path<=column_path+1;
        end
end
end



//vga 显示模块
always @ (posedge clk_25MHz)
begin
    if(rst==0&&flag==2'd1)
    begin
    if(vga_x >= 0  && vga_x <= 700  && vga_y >= 0 && vga_y <= 480)
    begin
        if((vga_x>=leftline&&vga_x<=rightline&&vga_y>=topline&&vga_y<=d
ownline)||(vga_x>=column_path&&vga_x<=column_path+30)||(vga_x>=column_p
ath1&&vga_x<=column_path1+30)||(vga_x>=column_path2&&vga_x<=column_path
2+30)||(vga_x>=column_path3&&vga_x<=column_path3+30)||(vga_x>=558+mario
_x-mario_x_sub&&vga_x<=580+mario_x-mario_x_sub&&vga_y>=170+mario_y-mari
o_y_sub&&vga_y<=202+mario_y-mario_y_sub))
```

```
begin
    if(vga_x>=column_path&&vga_x<=column_path+30)
    begin
    if((vga_y>=50&&vga_y<=120)||(vga_y>=170&&vga_y<=240))
    begin
    red1<=4'b0000;
    green1<=4'b0000;
    blue1<=4'b0000;
    end
    else
    begin
    red1<=4'b1111;
    green1<=4'b0000;
    blue1<=4'b1111;
    end
    end

    else if(vga_x>=column_path1&&vga_x<=column_path1+30)
    begin
    if(vga_y>=250&&vga_y<=320)
    begin
    red1<=4'b0000;
    green1<=4'b0000;
    blue1<=4'b0000;
    end
    else
    begin
    red1<=4'b0000;
    green1<=4'b1111;
    blue1<=4'b1111;
    end
    end

    else if(vga_x>=column_path2&&vga_x<=column_path2+30)
    begin
    if((vga_y>=180&&vga_y<=250)||(vga_y>=300&&vga_y<=370))
    begin
    red1<=4'b0000;
    green1<=4'b0000;
    blue1<=4'b0000;
    end
    else
    begin
    red1<=4'b1110;
```

```
                green1<=4'b0101;
                blue1<=4'b1111;
                end
                end


                else
                begin
                if(vga_y>=100&&vga_y<=170)
                begin
                red1<=4'b0000;
                green1<=4'b0000;
                blue1<=4'b0000;
                end

                else
                begin
                red1<=4'b0101;
                green1<=4'b1101;
                blue1<=4'b0000;
                end
                end
            end
            else
            begin
            red1<=4'b0000;
            green1<=4'b0000;
            blue1<=4'b0000;
            end
        end
        end
end

always @ (posedge clk_25MHz)
begin
    case(flag)
    2'd0:begin red=red0;green=green0;blue=blue0; end
    2'd1:begin red=red1;green=green1;blue=blue1; end
    2'd2:begin red=red2;green=green2;blue=blue2; end

    endcase
end

//绘制游戏开始界面 GO
```

```
always @ (posedge clk_25MHz)
begin
    if(vga_x >= 0  && vga_x <= 700  && vga_y >= 0 && vga_y <= 480)
    begin

    if(vga_x >= 174  && vga_x <= 194  && vga_y >= 150 && vga_y <= 300)
    begin
        red0<=4'b1111;
        green0<=4'b1111;
        blue0<=4'b1111;
    end
    else if(vga_x >= 454  && vga_x <= 474  && vga_y >= 150 && vga_y <=
300)
    begin
        red0<=4'b1111;
        green0<=4'b1111;
        blue0<=4'b1111;
    end
    else if(vga_x >= 534  && vga_x <= 554  && vga_y >= 150 && vga_y <=
260)
    begin
        red0<=4'b1111;
        green0<=4'b1111;
        blue0<=4'b1111;
    end
    else if(vga_x >= 534  && vga_x <= 554  && vga_y >= 280 && vga_y <=
300)
    begin
        red0<=4'b1111;
        green0<=4'b1111;
        blue0<=4'b1111;
    end
    else
    begin
        red0<=4'b0000;
        green0<=4'b0000;
        blue0<=4'b0000;
    end

    end
end


//绘制游戏结束界面 GG
always @ (posedge clk_25MHz)
```

```
begin
    if(vga_x >= 0  && vga_x <= 700  && vga_y >= 0 && vga_y <= 480)
    begin

    if(vga_x >= 174  && vga_x <= 194  && vga_y >= 150 && vga_y <= 300)
    begin
        red2<=4'b1111;
        green2<=4'b1111;
        blue2<=4'b1111;
    end
    else if(vga_x >= 534  && vga_x <= 554  && vga_y >= 280 && vga_y <=3
00)
    begin
        red2<=4'b1111;
        green2<=4'b1111;
        blue2<=4'b1111;
    end
    else
    begin
        red2<=4'b0000;
        green2<=4'b0000;
        blue2<=4'b0000;
    end

    end
end

endmodule
```

五、测试模块建模

1. vga 模块

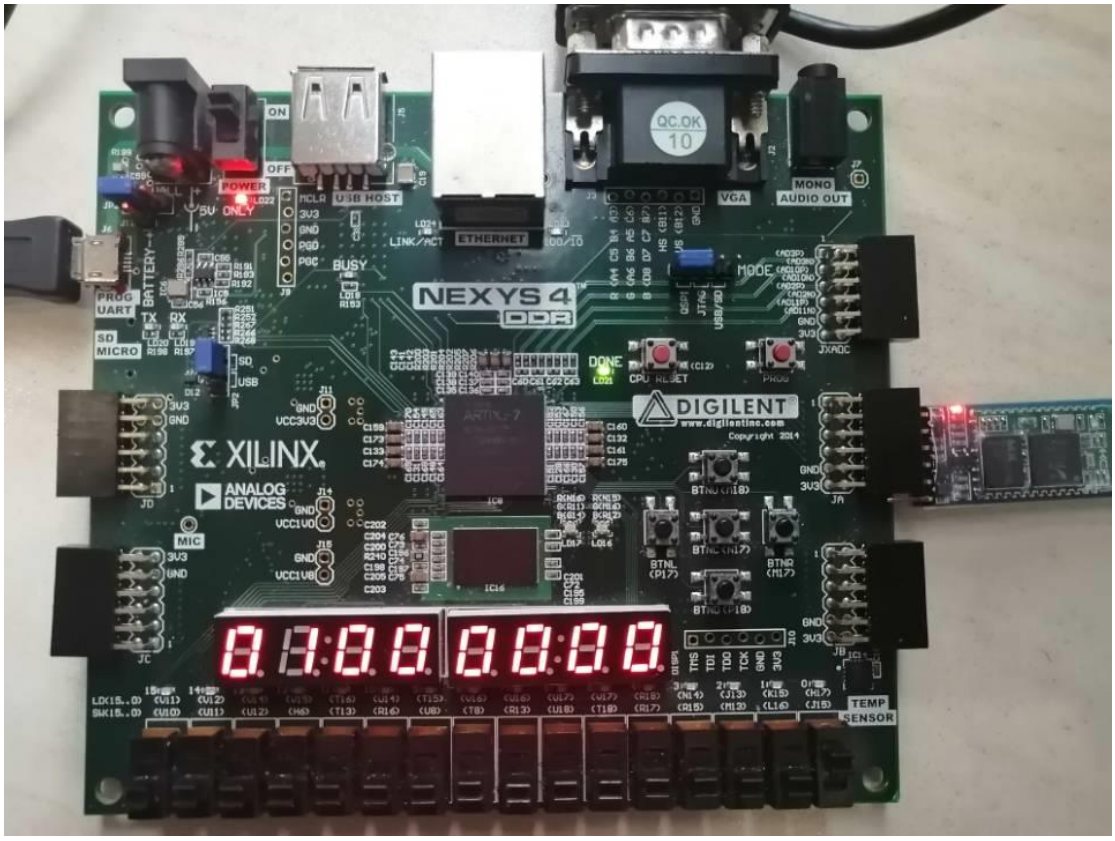
对于 vga 采用直接下板方式，采用对屏幕均匀划分不同颜色彩条方式进行检验。

六、实验结果

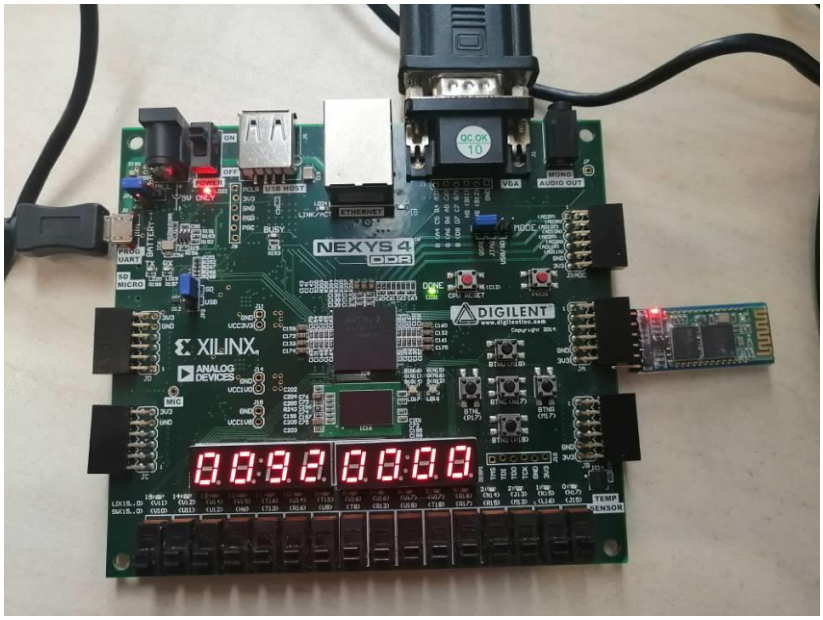
（该部分可截图说明，可包含 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图）

1. timecount 模块

下板结果：



游戏一开始有 100 秒的倒计时

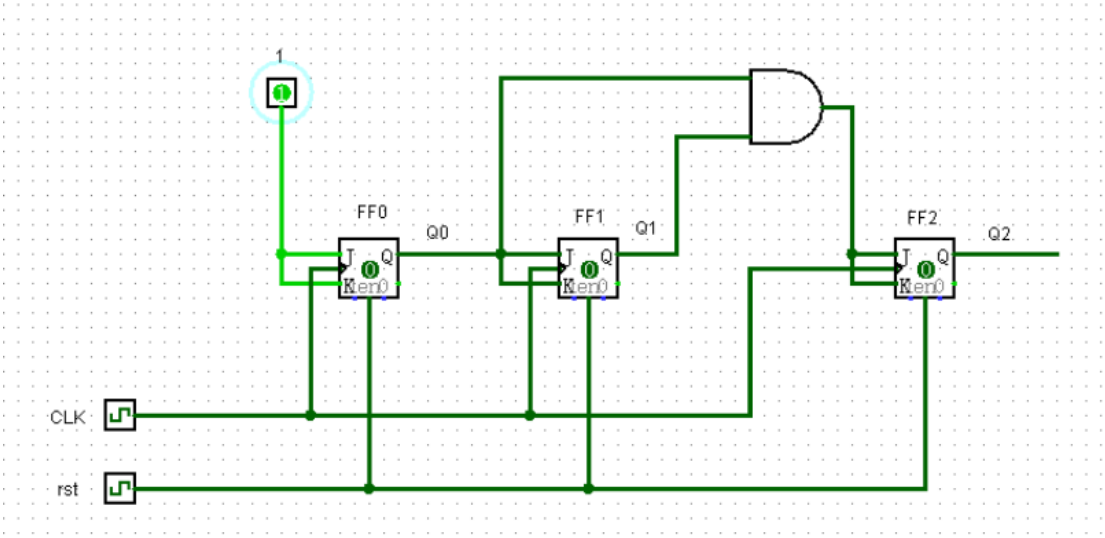


倒计时正常进行，timecount 模块运行正常

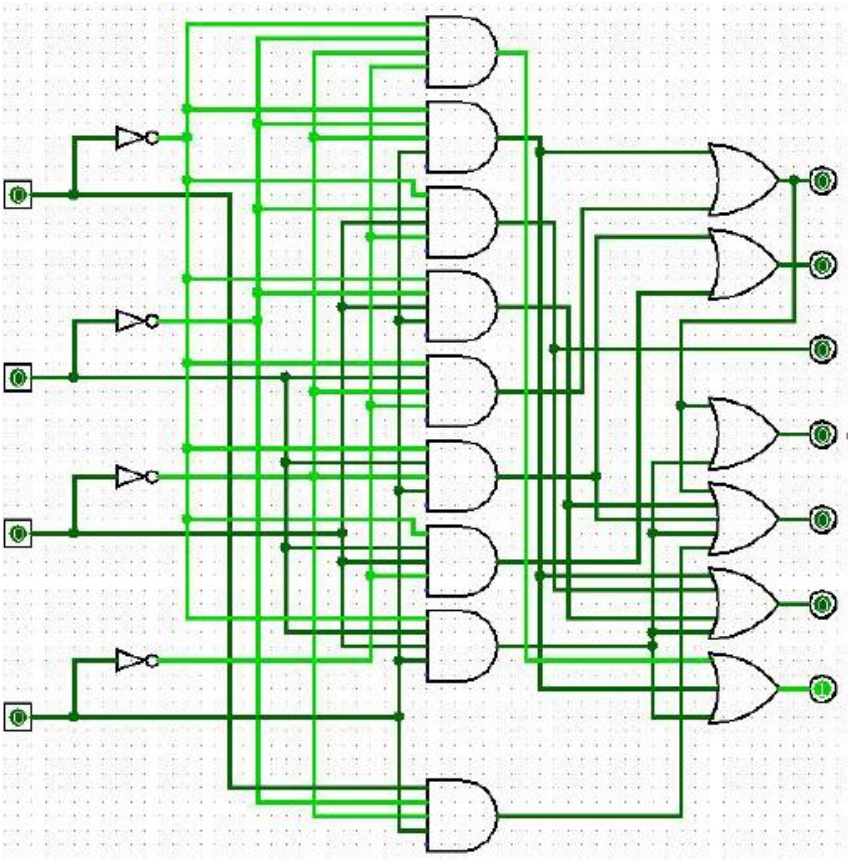
2. display7 模块

加法器与计数器逻辑正确，可接下去进行数码管显示操作。

计数器 logisim 原理图：



七段数码管 logisim 原理图：



3. vga 模块

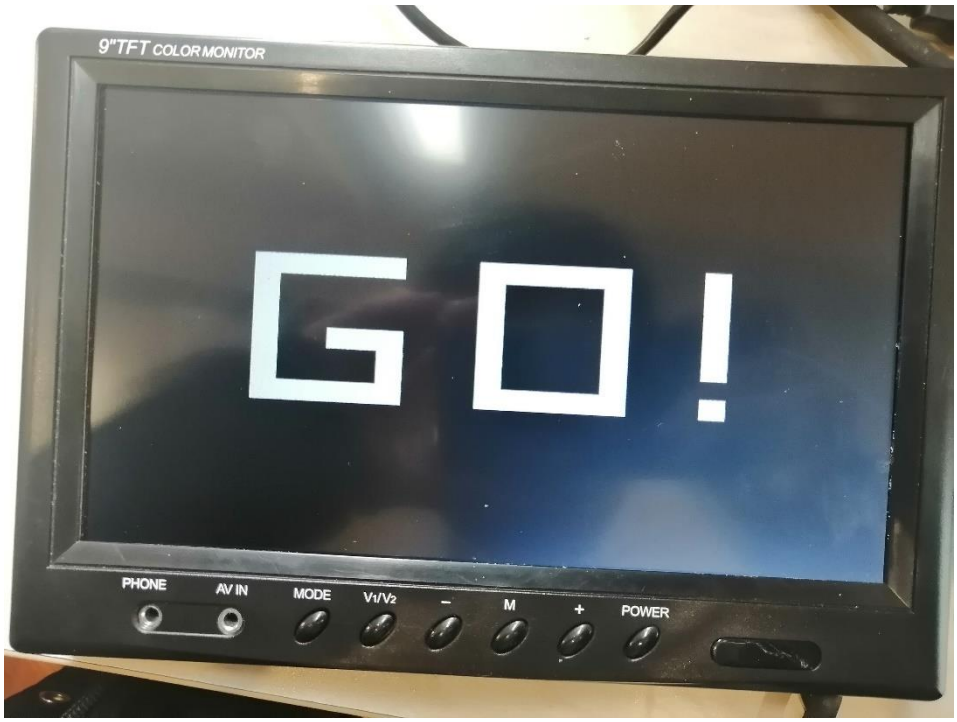
由于 vga 模块的特殊性，直接下板实验效果更佳，采用对屏幕均匀划分不同颜色彩条方式进行检验。

开始阶段对 vga 行序和列序处理有些小毛病，导致颜色不均匀。

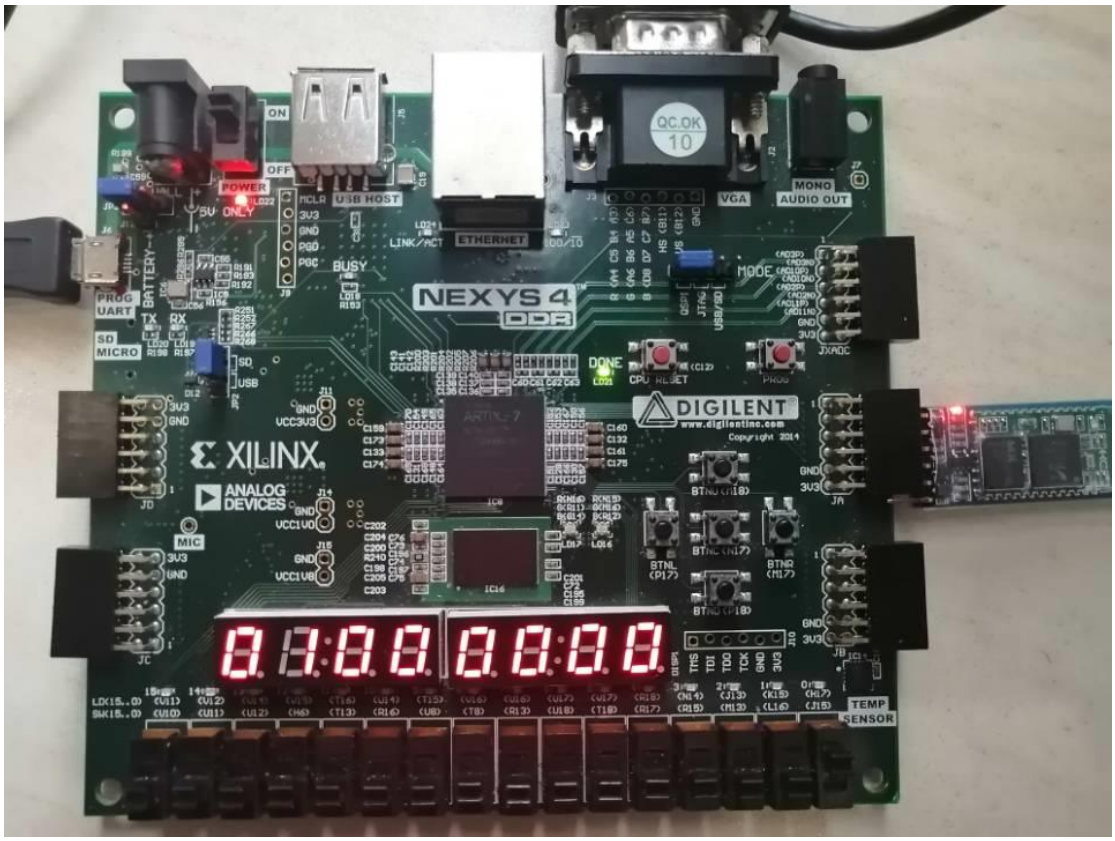
调整后，得到了理想的结果，vga 模块运行正常。

总实验结果：

① 程序运行下板后 vga 显示如下界面，指示游戏就绪界面

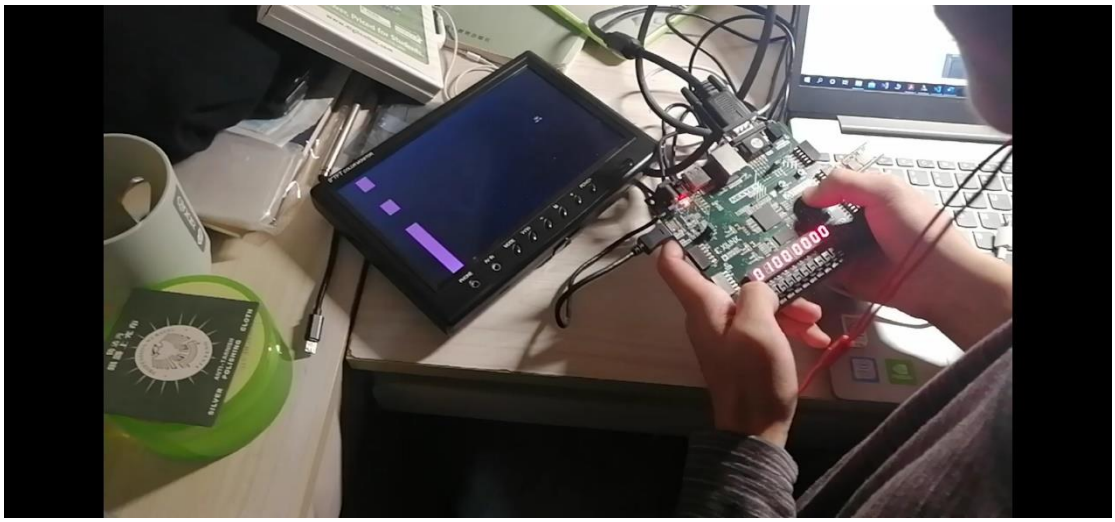


此时七段数码管中左侧倒计时模块时间固定在 100 秒，由于游戏未开始，右侧计分模块为 0 分。

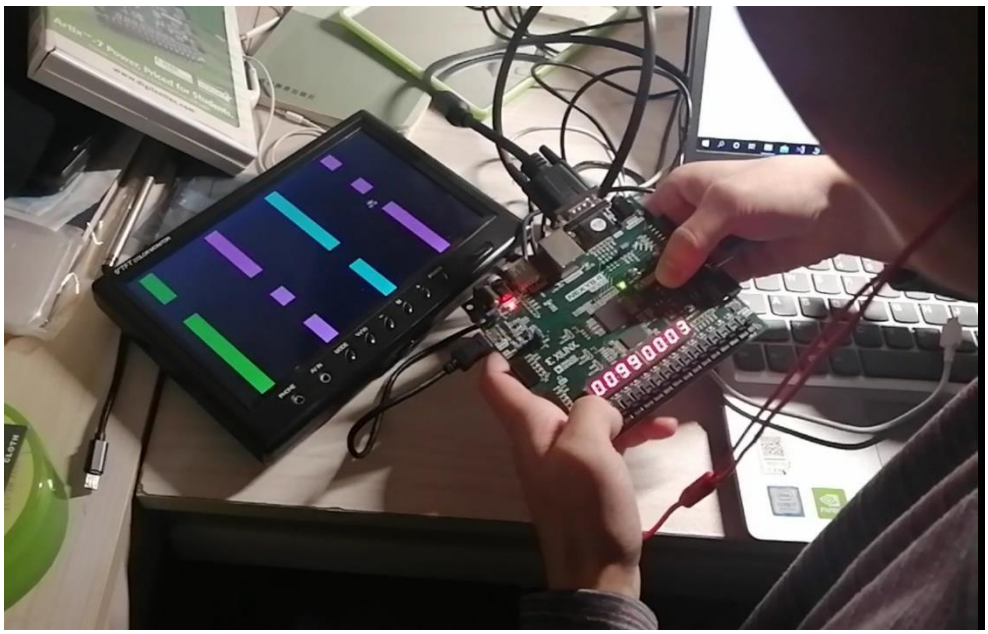


② 按下 START 键后 vga 跳转至如下游戏界面：

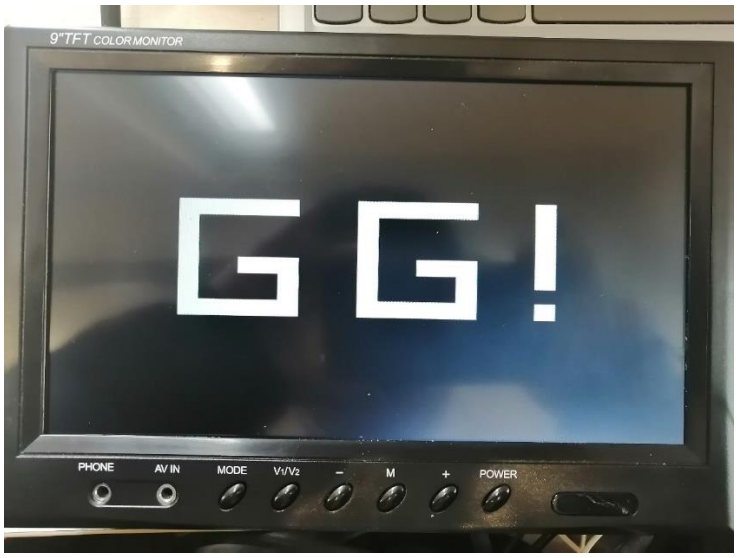
③ 游戏开始



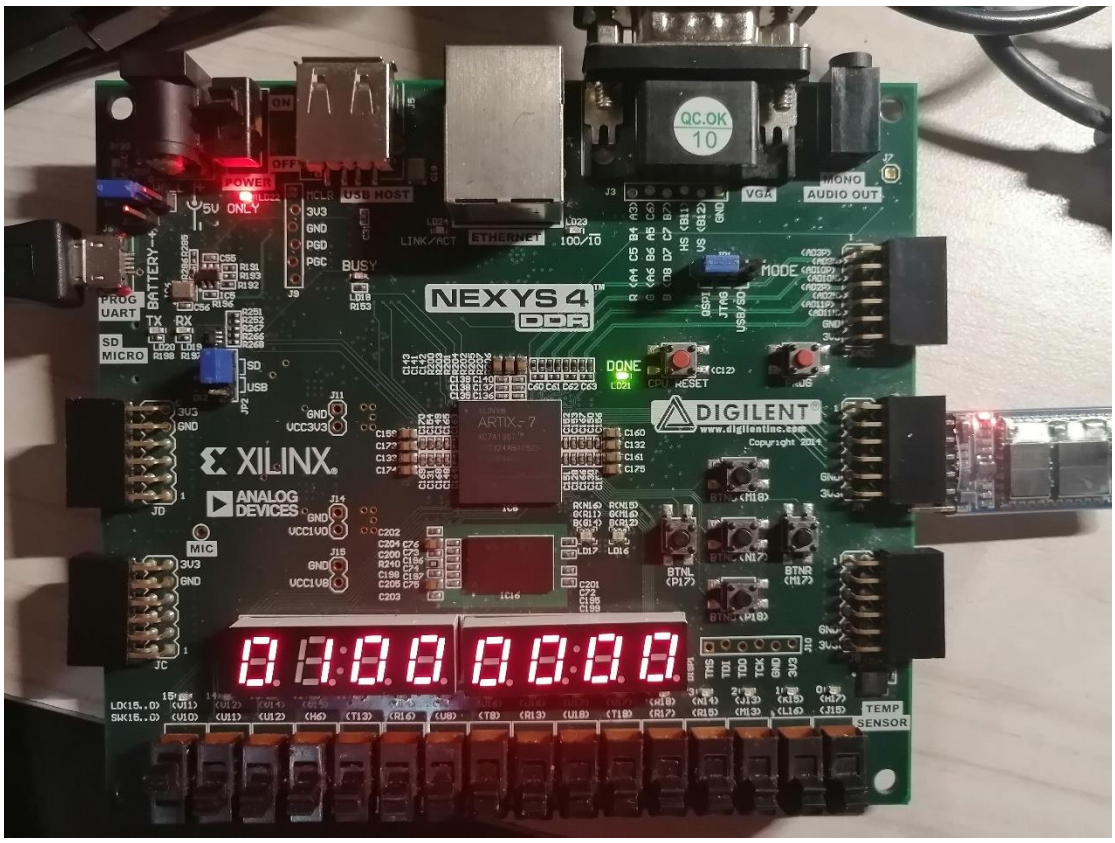
④ 顺利通过通道后加分

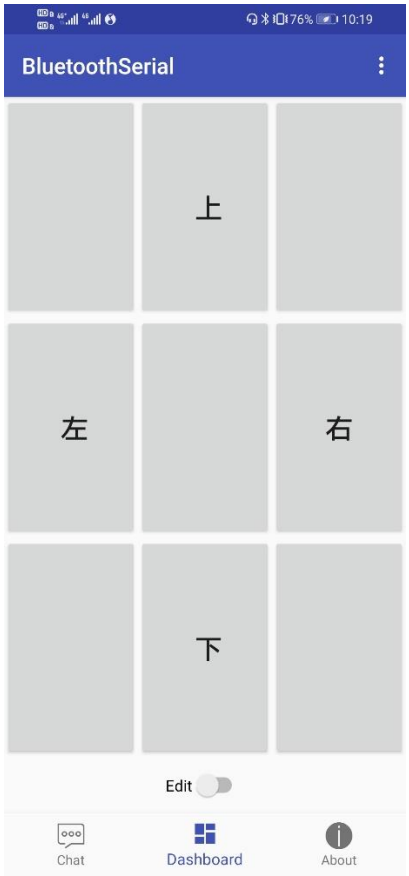


⑤ 马里奥撞到柱子后，游戏结束，显示结束界面 GG!



此时得分清零，倒计时恢复 100s





同时玩家还可以通过手机蓝牙对马里奥进行移动游戏。

七、结论

在分别经过下板前的 modelsim、logisim 等逻辑验证以及下板后的逻辑验证后，可得出结论：各个子模块以及整个数字系统均可按预期假设功能工作，因此这一数字系统是能够正常工作的。

八、心得体会及建议

1. 本课程收获

本学期我们开设了数字逻辑这门课程，此课程是计算机专业以及其他机电类专业均涉及到的课程，是后续计算机组成原理等硬件课的基础。对于这门课的学习，我有比较深刻的体会：从一开始阅读不懂的 Verilog 代码，不会写的小实验到后期对 Verilog 有了初步的基础了解以及能用它完成一个略有规模的数字系统设计。我觉得这是一个循序渐进的过程，在课程的不断推进过程中，我逐步了解了从逻辑电路导论到组合逻辑电路，再到学习完锁存器触发器等存储部件后对时序逻辑电路的学习与应用，期间通过各小作业对不同部件有了初步了解也对之前苦涩难懂的部分 Verilog 代码有了新的认识。总体来说，这门课让我认识到了什么是硬件编程，硬件设计以及具体操作步骤。

其次就是我学会了有关硬件编程以及代码调试、测试，逻辑分析等重要技能。在每次编写完一个程序后，学会了通过编写_tb.v 测试文件测试的方法，辅以 modelsim 波形分析对已完成代码逻辑进行验证，确保代码准确性；同时，我也学会了通过 logisim 画原理图以及调试分析的过程，这些都是正常完成一个程序设计很重要的方法。

最后就是在课程最后的大作业——设计一个数字系统。在设计过程中，我从一开始拿到外设后的一筹莫展，毫无头绪到查阅了老师发放的相关资料以及网上资料后，对 VGA 原理有所了解，知道了 VGA 是通过行扫描以及列扫描实现 VGA 的显示再到设计自己的游戏逻辑、在 VGA 上显示整个过程，我感受到了从未知通过许多渠道不断探索的过程，再到成功实现一个功能的喜悦。在此过程中，我同时也综合运用了波形图仿真以及直接下板分析的方法，对程序不断 debug，最终完成一个完整的数字系统设计。

2. 对本课程建议

首先是授课阶段，建议老师在完整概念讲解分析的基础上添加一些具体实例，以便能对相关知识有更加深刻的了解。

其次就是对于最后阶段的大作业——设计一个数字系统，老师提供的外设数量众多，但有的难度相差极大，资料完整度也有较大差别；这对于同学们选择外设以及真正着手操作造成了比较大的选择困难。同时类似于 OV2640 摄像头、ThinkPad 键盘等外设看似功能强大，但开发起来极其困难，资料匮乏以及复杂的协议均为此次大作业大大增加了难度。由此，我建议老师可以将各部件有关协议、寄存器等部分的核心代码上传至课程网站或微信群以供同学们参考，同学们便可以在大作业设计中真正将时间花在数字系统的构思、设计、完善上，而不是在摸索各部件协议、查询资料的过程中耗费了大量时间。

再者，我比较建议老师在大作业设计阶段明确得分细则，比如完成一个数字系统得多少基础分、在此基础上完成部分进阶功能加多少分，以防止老师模糊得

分要求，纯粹以总体水平定论，避免不必要的恶性竞争，也会使同学们得设计过程更有目的性。

3. 结合数字芯片设计国内外现状深入谈自己认识与体会

半导体实际上是广义的芯片概念，而集成电路是半导体中最重要的组成部分。集成电路设计是指以集成电路、超大规模集成电路为目标的设计流程。而随着近期美国加大对华制裁力度，随着美国对华芯片禁令的发布，不仅又引人深思：漫漫中国芯片，路在何方？

芯片设计是行业最上游，其作为集成电路产业链五大环节中的第一环，毫无疑问也是最重要的一环。如果连芯片的设计都无法完成，那么后续的其他环节自然也无法顺利的开展。芯片设计是将系统、逻辑和性能的设想转化为物理实现的过程，也是整个产业链中技术含量和利润率都相对比较高的环节。而在此环节中，美国一家独大，占据了超过一半的市场份额。而市场占有率前十的企业，中国大陆仅有华为海思一家，而且芯片制造尺寸，技术也与世界一流有较大差距。

如果说芯片设计是行业最上游，而整个设计工作，基本上是在 EDA 软件上完成，因此 EDA 是整个芯片行业的最上游。EDA 投入巨大，但收效甚微，产出不明显，需要长时间积累沉淀，因此 EDA 的国产化进展相当慢。目前国产的 EDA 还没有办法提供全流程的产品，只能提供一些细分领域的工具。

在底层构架方面，处理器的底层架构目前还基本无法进行国产化。因为底层架构是和操作系统绑定的，即英特尔的 CPU 和微软的 Windows 系统、安卓系统和 ARM 的芯片，以及苹果处理器与 IOS 系统等。所以这是一个垄断的细分行业。电脑处理器 x86 由英特尔和 AMD 垄断，而手机架构 ARM 由 ARM 公司独家垄断。

由以上可知，中国芯片设计产业在几大重要环节均存在较大差距，美国产业链拥有微处理器、EDA 设计软件、光刻机三大领先优势，同时成为中国芯片企业必须翻过的“三座大山”。芯片设计的国产化任重而道远，前路也依然漫漫。

而我认为面对复杂的国际科技局势，中国芯片想要突围可以考虑：

1. 开放结盟：学习 Intel 联合美国整个芯片产业组成“极紫外联盟(EUVLLC)”，中国应与非敏感国家形成新一代芯片研发产业链。比如，借助 RCEP 自由贸易框架，与日韩等亚太国家产业链融合发展，形成堪比欧盟、北美的亚洲多边研发阵营，投入共摊、利益共享。
2. 研发变轨：硅基芯片离“天花板”越来越近，选择新材料研发未来芯片将率先开展“终局竞争”。
3. 瞄准下一代 AIoT 终端的“算法+芯片”需求进行研发，将获得“弯道超车”的时代红利。
4. 通过政策引领，市场拉动，给国内芯片企业一张“网”，让企业可以去市场中抓到更多的“鱼”，是实现良性循环的动力源泉。国产处理器芯片已经具备一定基础，基本可以满足替代要求，相信通过政策引领，市场的大门一定会越开越大，国产芯片的路也会越走越宽。