浙江大学
数字逻辑设计•课程报告

基于数字系统的飞机大战游戏设计

组长：吴君洋

组员：林家丰、胡晨旭、沈刻

2019年1月12日

目录

一、游戏设计背景

1. FPGA 与开发平台

FPGA（Field－Programmable Gate Array，即现场可编程门阵列）是在PAL、GAL、CPLD等可编程器件的基础上进一步发展产物。以硬件描述语言（Verilog或VHDL）所完成的电路设计，可以经过简单的综合与布局，快速的烧录至FPGA上进行测试。这些可编程元件可以被用来实现一些基本的逻辑门电路或者更复杂的一些的组合功能。Xilinx的SWORD实验板为可编程的实验板，而且实验板还可以和许多外部设备进行连接，例如PS2、鼠标或者显示器，从而可以在FPGA的基础上设计使用PS2、鼠标的调用，或者显示器上显示图片。

本次设计所用的硬件编程软件为Xilinx公司设计的ISE 14.7软件，所使用的语言为Verilog语言。该软件的使用和Verilog语言的学习都较为简单，和c语言有相似之处。同方式或混合方式对设计建模。这些方式包括：行为描述方式建模；数据流方式建模；结构化方式建模等。Verilog HDL 中有两类数据类型：线网数据类型和寄存器数据类型。线网类型表示构件间的物理连线，而寄存器类型表示抽象的数据存储元件。通过模块化的设计与相应的外部接口，可以实现较好的人机交互。

本次设计通过原理图和行为描述相结合的方式实现游戏的设计，显示器、PS2 外接设备优化游戏。

2. 游戏介绍

《飞机大战》这是一款经典飞行射击类游戏，精美绚丽的画面，整体环境主要还是围绕太空为主，高保真的图像，为玩家呈现一场不一样射击体验。控制方向键，实现上下左右移动，便可自动攻击敌人，上下移动亦可躲避强敌。玩家在游戏中要做的就是驾驶着最新战机，躲避敌机的子

弹，并且用子弹击落尽可能多的敌方飞机，以获取更多的分数。

3. 设计介绍

本次课程设计是利用FPGA板，通过底层硬件控制的方法实现飞机大战这款游戏，但是由于板子资源的限制，进行了一定程度上的简化。

在本组设计中，我们用外接的ps2键盘进行游戏的控制，采用四个按键（W、S、A、D），分别控制我方飞机进行上下左右的移动，来躲避敌机的子弹和寻找好的击敌位置。敌方飞机会随机出现，和发射子弹。当敌机与我方飞机相撞时，游戏直接结束；被敌机子弹击中时，扣一滴血，血量一共三滴，扣完即游戏结束。

本组的游戏界面封面图为电影《星球大战》的情境图，开始游戏后背景变为随着时间移动的浩瀚星空，游戏界面由显示器显示，板子上的七段数码管显示游戏得分。

4. 重难点分析

(1)在显示器上正确的显示背景图以及背景的移动。

(2)能够调用键盘完美地控制飞机移动，并且能对键盘的传输信号进行正确的判断。

(3)能够正确判断大量射出子弹的位置，并对完成子弹命中目标后消失和后续扣血的逻辑。

(4)能够随机地产生敌机，记录敌机和我方飞机的血量，敌机的爆炸效果。

(5)能够正确实现敌机的爆炸和消失，并且通过七段数码管显示出当先获得的分数。

(6)撞机的检测，图片原本为矩形，用适当的方式选取采样点进行检测。

(7)图片能与背景完美融合，没有白边。

二、整体结构

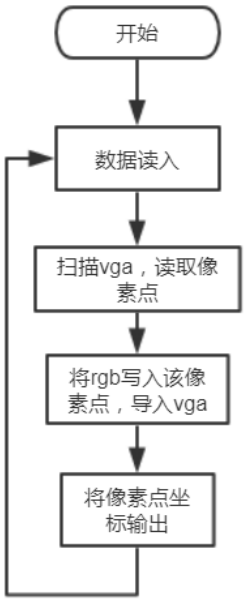
1. 输入与输出

本游戏用外接键盘进行输入，采用四个按键（Q、S、A、D），分别控制我方飞机进行上下左右的移动；回车键控制游戏从封面开始；空格键控制 game over 后，重新回到主菜单。

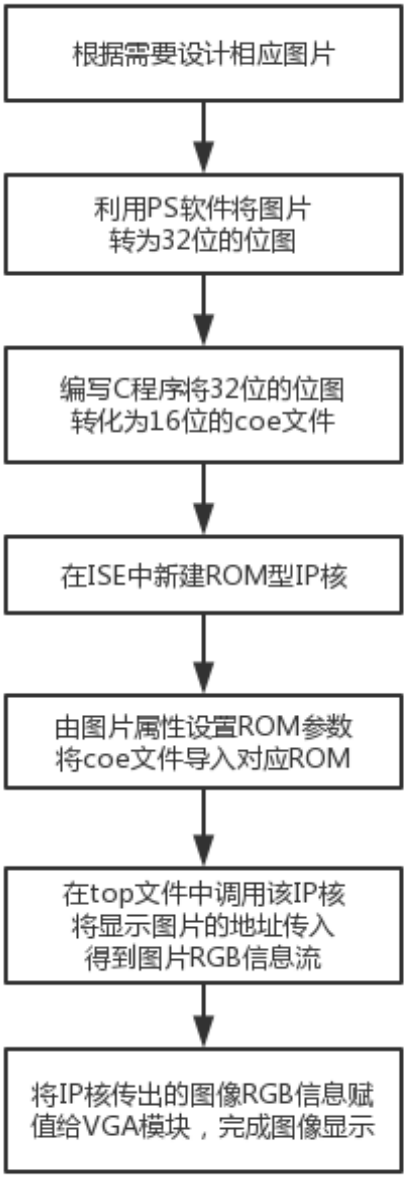
输出由两部分组成，一部分为 VGA 模块的显示器游戏界面输出；另一部分为板子上七段数码管显示的游戏得分。

2. 模块工作流程

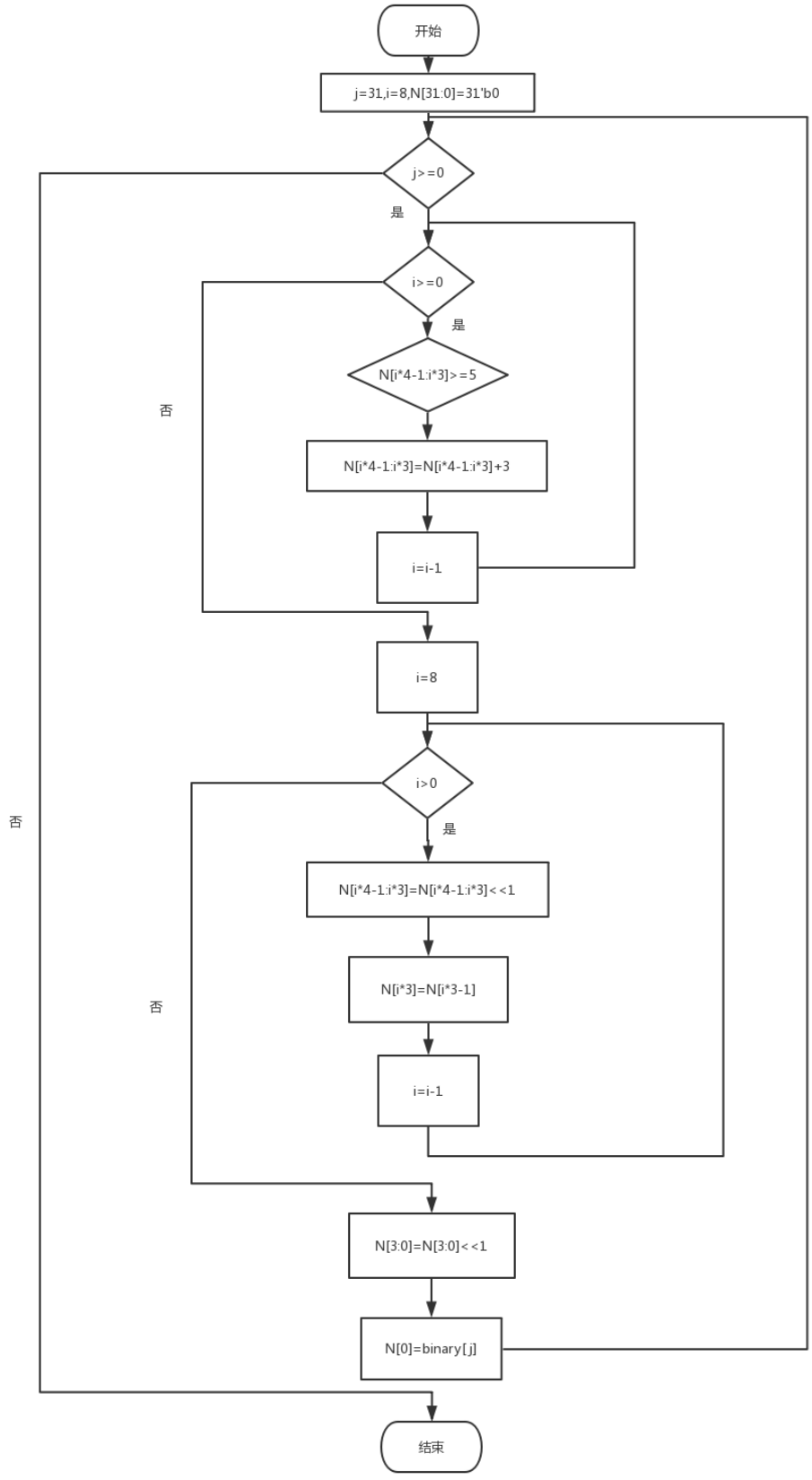
（一）VGA 模块程序流程图



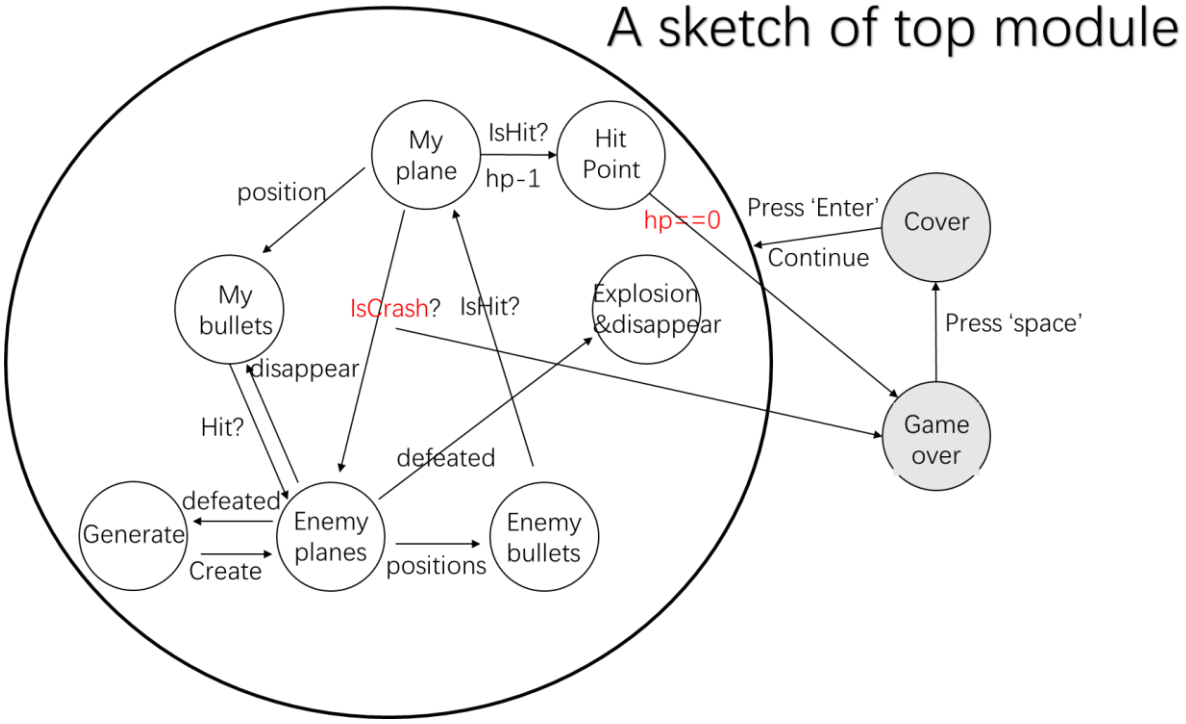
（二）图片导入模块程序流程图



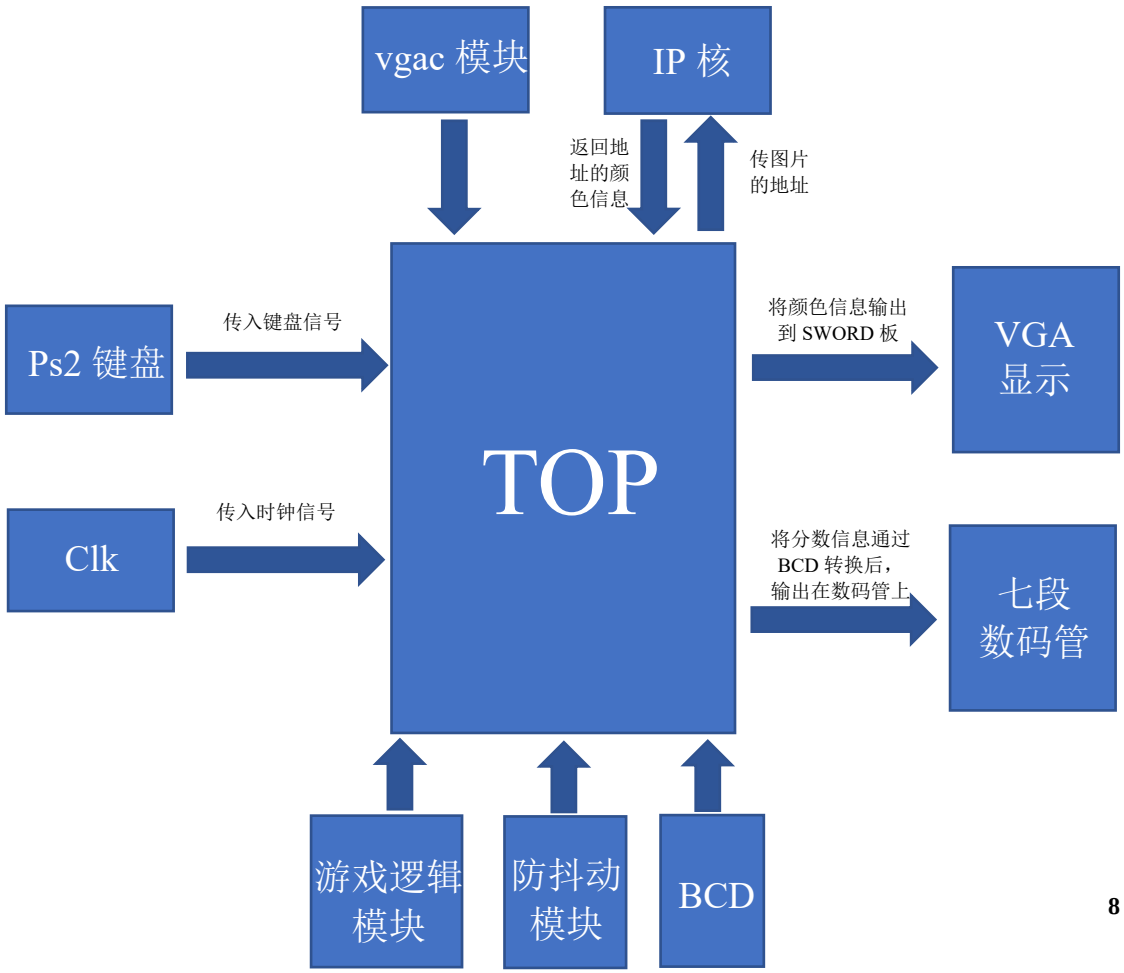
（三）二进制转 BCD 码模块流程图



（四）top 模块逻辑部分示意图



（五）各模块之间关系图



将 bit 文件下载到实验板后，显示封面图样。按回车键进入游戏。我方飞机发射炮弹，若击中敌人飞机，则炮弹消失，同时敌人生命值减少。待敌方飞机被击中十次后爆炸，随后消失。敌人被击落后，会在随机时间内重新产生，并发射炮弹。若在游戏过程中撞到敌人飞机，或是被敌方炮弹击中 3 次，游戏结束。

3. 游戏过程

游戏的总体过程如下图。

首先进入游戏封面，按下ENTER键后，进入游戏界面，开始游戏。在屏幕的上方随机产生敌方飞机，我方飞机在屏幕下部。我方和敌方飞机都会自行开始发射子弹，进行攻击。通过控制键盘来控制我方飞机移动，ps2 module会将ps2键盘10位的data传给top module，在top中进行判定和检测，以判断是不是方向控制键，进而由飞机控制逻辑控制飞机位置改变。与此同时VGA扫描将更新界面。

打掉敌方一架飞机后，会在敌方飞机处产生爆炸效果，并获得100分，分数通过七段数码管显示。

Game over状态：

1. 我方飞机与敌机相撞后，我方飞机直接死亡，游戏结束；

2. 敌方子弹击中我方飞机，扣血一滴，血量一共三滴，扣完即游戏结束；

之后进入Game over界面，按下空格键后回到封面。按下回车重新开始下一次游戏。

三、模块介绍

1．图片导入

（1） 我们通过对《星球大战》截图并重新编辑来获取游戏的背景。

（2） 对背景图进行简单的PS处理，包括去掉得分字符，修改像素大小，得到32位bmp文件。

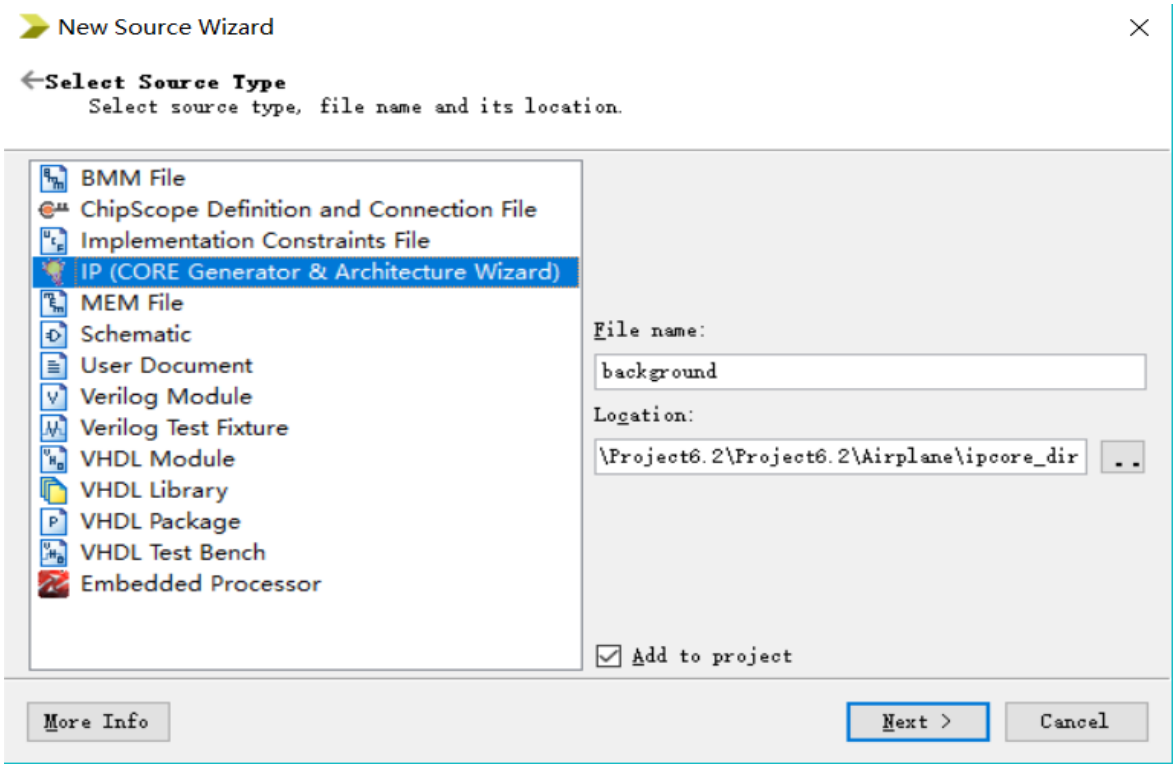
（3） 用c语言写的程序，处理图像颜色信息，保存为coe文件。

封面图片

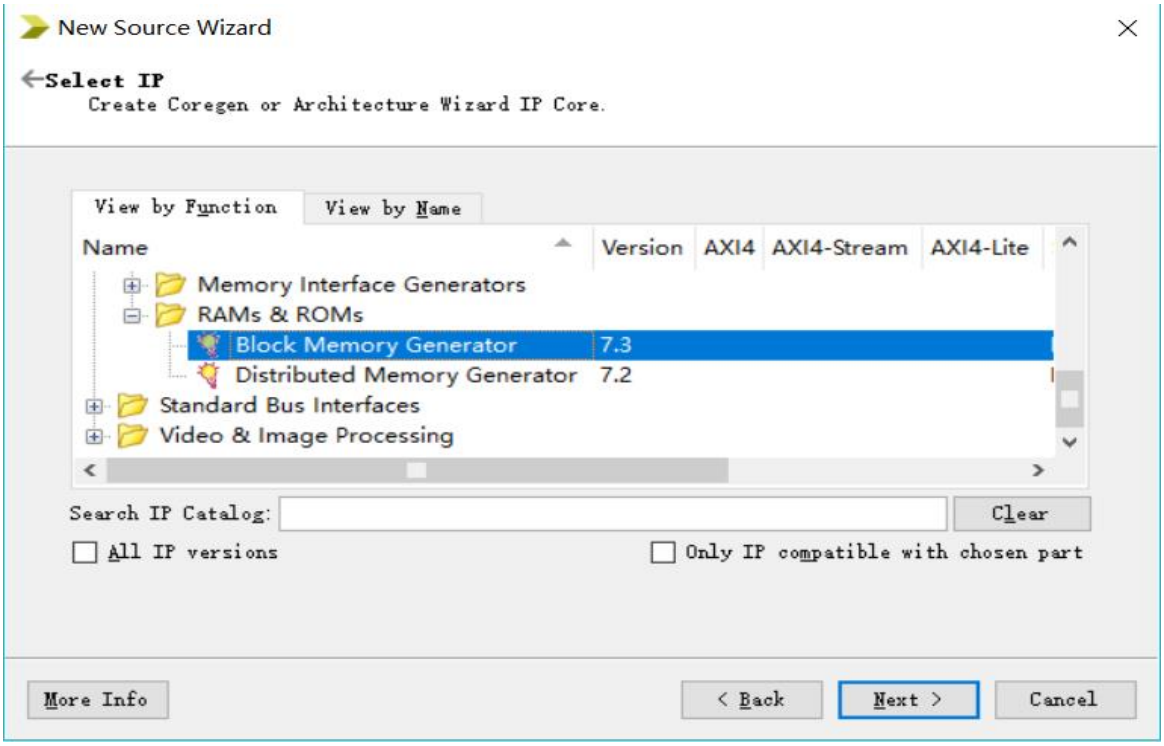
背景图片

具体操作如下：

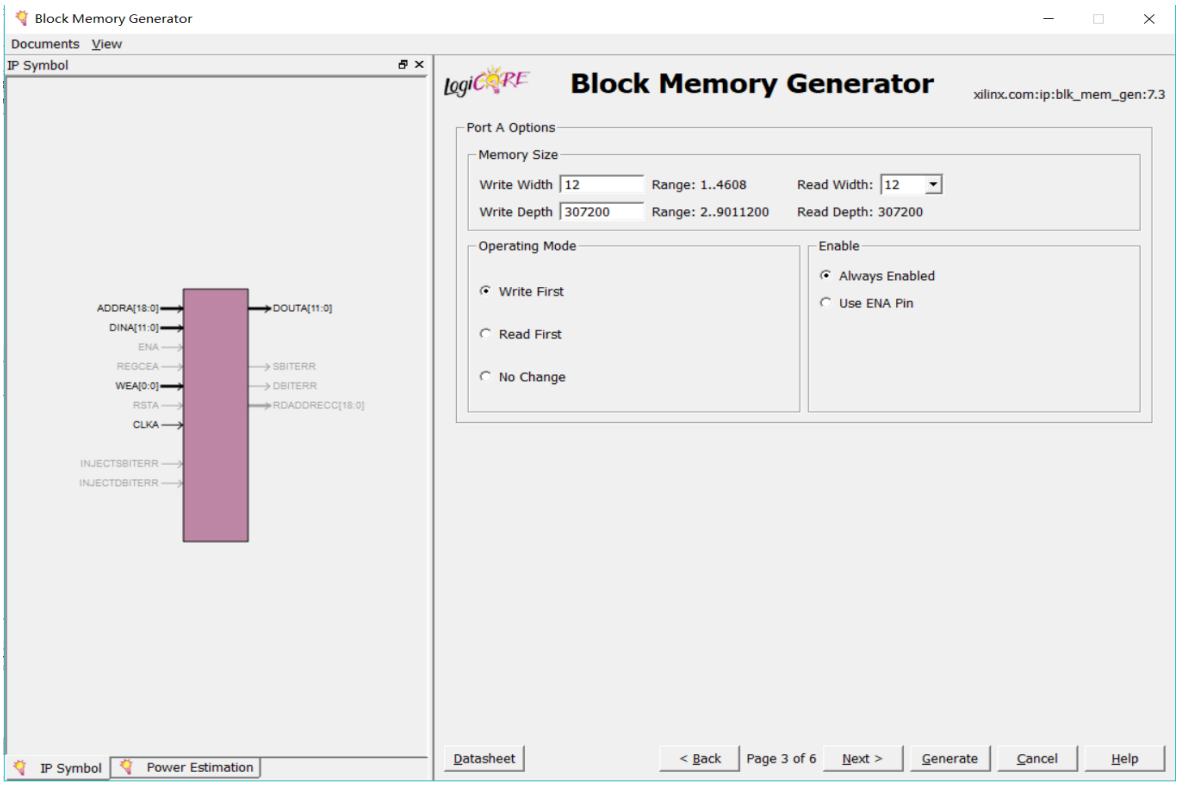
1、创建 ip 核，名字起为 background。



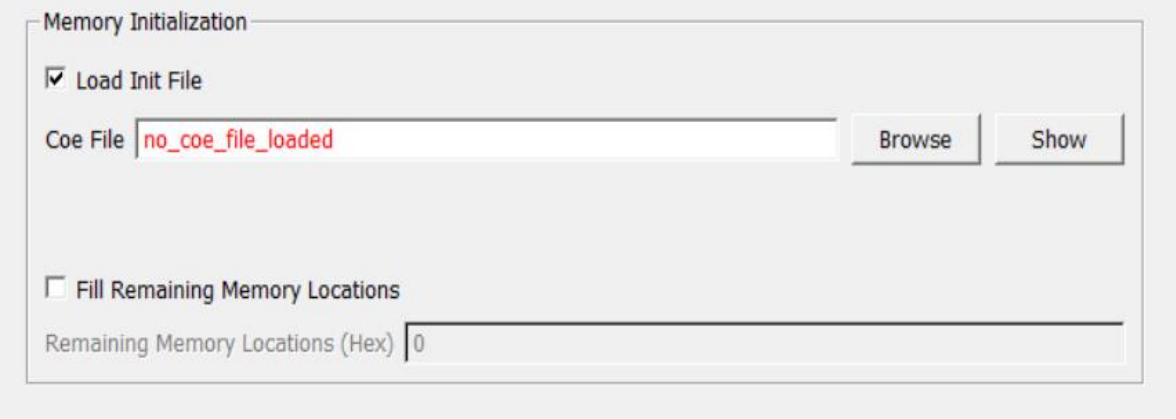
2、选择 Block Memory Generator，点击 next。



3、调整合适的参数。其中 Write Width 是输出的位数，由于 vga 是 12 位输出，所以我们选择 12。Wirte Depth 是图片的大小，由于 vga 的显示屏大小是 640*480，所以我们填入的参数是 307200（640*480）。由左边的 symbol 可以看出，ADDRA 是 ip 核的输入，也就是图片的地址，由于我们不需要写的操作，所以不用理会 DINA 和 WEA。CLKA 是扫描频率。



4、在page4选择想要显示的图片的coe文件，经过查阅资料我们发现其中的coe文件可以用c语言代码来实现从bmp到coe的转换。（代码将贴在附录）



5、选择完毕后，点击generate即可生成ip核，ip核会根据coe文件的大小来加载，所以图片越大，加载的时间越长，画面未响应的现象是非常正常的。

6、成功生成后，会出现这样的图标---->b1 - background (background.xco)

2．VGA显示

VGA的显示分为三个步骤，让我们来依次介绍一下。

1、在调用vgac模块之后，我们首先要调用自己生成的ip核的模块。

```
background b1(.addra(bg1),.douta(spob),.clka(clkdiv[1]));
```

2、接着，我们需要给ip核的输入进行赋值。我们采取用图片左上角的坐标来表示图片的位置。我们判定如果col_addr和row_addr在该显示的区域内，我们就给输入赋值成(col_addr-x)*图片高度+(row_addr-y),其中x，y即为图片左上角的坐标，col_addr和row_addr为vga的扫描横坐标和纵坐标。

```
bg1<=(col_addr>=0&&col_addr<=639&&row_addr>=BG_Y[8:0]&&
row_addr<=BG_Y[8:0]+479)?col_addr*480+(row_addr-BG_Y):0;
```

3、在给输入赋值完毕后，我们就可以选择合适的位置让图片显示。要想让图片显示，只需将ip核的地址赋给vga_data即可。

```
if(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)
vga_data<=spob[11:0];
```

3．PS2键盘输入

PS2键盘传输的数据会在data_out中输出

```
module ps2_ver2(
                input clk,
                   input rst,
```

```
                input ps2_clk,                              //键盘时钟输入
                input ps2_data,                             //键盘数据输入
                output [9:0] data_out,                       //键盘扫描骂输出
                output ready
    );

    reg ps2_clk_flag0,ps2_clk_flag1,ps2_clk_flag2;
    wire negedge_ps2_clk;

    always@(posedge clk or posedge rst)begin            //判断键盘输出时钟是否
有连续低电平
        if(rst)begin                                    //来应用请求发送
            ps2_clk_flag0 <= 1'b0;
               ps2_clk_flag1 <= 1'b0;
               ps2_clk_flag2 <= 1'b0;
         end
         else begin
            ps2_clk_flag0 <= ps2_clk;
               ps2_clk_flag1 <= ps2_clk_flag0;
               ps2_clk_flag2 <= ps2_clk_flag1;
         end
    end

    assign negedge_ps2_clk = !ps2_clk_flag1 &ps2_clk_flag2;
    reg [3:0] num;
    always @(posedge clk or posedge rst)begin
      if(rst)
          num <= 4'd0;
      else if(num==4'd11)
           num <= 4'd0;
      else if(negedge_ps2_clk)
          num <= num +1'b1;
    end
    reg negedge_ps2_clk_shift;
    always @(posedge clk) begin
       negedge_ps2_clk_shift <= negedge_ps2_clk;
    end

    reg [7:0] temp_data;
    always @(posedge clk or posedge rst)begin
       if(rst)
            temp_data <= 8'd0;
        else if(negedge_ps2_clk_shift) begin            //键盘时钟连续低电平接
收数据
```

```
            case(num)                                    //存储八个数据位

                  4'd2 : temp_data[0] <= ps2_data;
                   4'd3 : temp_data[1] <= ps2_data;
                   4'd4 : temp_data[2] <= ps2_data;
                   4'd5 : temp_data[3] <= ps2_data;
                   4'd6 : temp_data[4] <= ps2_data;
                   4'd7 : temp_data[5] <= ps2_data;
                   4'd8 : temp_data[6] <= ps2_data;
                   4'd9 : temp_data[7] <= ps2_data;
                   default : temp_data <= temp_data;
              endcase
        end else
            temp_data <= temp_data;
    end
    reg data_break,data_done,data_expand;
    reg [9:0] data;
    always @(posedge clk or posedge rst)begin
       if(rst) begin
           data_break <= 1'b0;
            data <= 10'd0;
            data_done <=1'b0;
            data_expand <= 1'b0;
        end
        else if(num == 4'd11)begin
           if(temp_data == 8'hE0)
                 data_expand <= 1'b1;                    //键盘扩张键
              else if(temp_data == 8'hF0)
                 data_break <= 1'b1;
              else begin
                 data <= {data_expand,data_break,temp_data};
                  data_done <= 1'b1;                     //键盘数据已准备好输
出
                  data_expand <= 1'b0;
                  data_break <=1'b0;
             end
        end
        else begin
          data <= data;
           data_done <= 1'b0;
           data_expand <= data_expand;
           data_break <= data_break;
        end
```

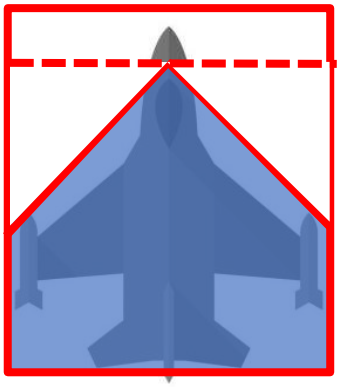
```
    end

    assign data_out = data;
    assign ready = data_done;                    //输出键盘数据
endmodule
```

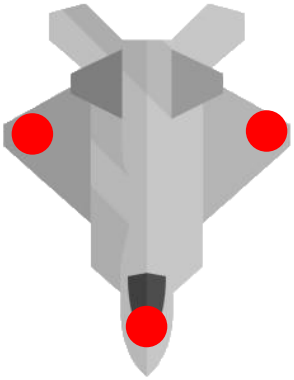
4. 七段数码管显示

将获得的分数输出在七段数码管上

```
module Seg7Device(
    input clkIO, input [1:0] clkScan, input clkBlink,
    input [31:0] data, input [7:0] point, input [7:0] LES,
    output [3:0] sout, output reg [7:0] segment, output reg [3:0] anode
);
    wire [63:0] dispData;
    wire [31:0] dispPattern;

    Seg7Decode U0(.hex(data), .point(point),
        .LE(LES & {8{clkBlink}}), .pattern(dispData));
    Seg7Remap U1(.I(data), .O(dispPattern));

    ShiftReg#(.WIDTH(64))U2(.clk(clkIO), .pdata(dispData), .sout(sout));

    always @*
        case(clkScan)
        2'b00: begin segment <= ~dispPattern[ 7: 0]; anode <= 4'b1110; end
        2'b01: begin segment <= ~dispPattern[15: 8]; anode <= 4'b1101; end
        2'b10: begin segment <= ~dispPattern[23:16]; anode <= 4'b1011; end
        2'b11: begin segment <= ~dispPattern[31:24]; anode <= 4'b0111; end
        endcase

endmodule
```

5. IsCrash模块-判断我方飞机是否被敌方飞机撞上





即判断敌方飞机的三个红点是否进入我方飞机的蓝色区域。

如果敌机的其中一个点进入蓝色框内，IsCrash返回1。

```
module IsCrash(
    input wire clk,
    input wire  [9:0] ei_x,
    input wire  [8:0] ei_y,
    input wire  [9:0] mi_x,
    input wire  [8:0] mi_y,
    output wire check
   );
     reg [9:0] e_xl;
     reg [8:0] e_yl;

     reg [9:0] e_xm;
     reg [8:0] e_ym;

     reg [9:0] e_xr;
     reg [8:0] e_yr;

     reg [9:0] m_x;
     reg [8:0] m_y;

     reg in;

     integer k;
     integer x;
     integer y;

     always @(posedge clk) begin
         e_xm<=ei_x+32;
         e_ym<=ei_y+77;

         e_xl<=ei_x+2;
```

```
        e_yl<=ei_y+30;

        e_xr<=ei_x+62;
        e_yr<=ei_y+30;

        m_x<=mi_x+40;
        m_y<=mi_y+41;
        in = 0;
        k = 30;
        x=30;
        y=30;
        if(e_xm< m_x+38 && e_xm>m_x-38 && e_ym <m_y+38 && e_ym >m_y-38)
begin//in the square
                if(e_ym<m_y) begin
                    if(e_xm<m_x)begin
                        k= m_x- e_xm + m_y- e_ym;
                    end
                    else begin
                        k= e_xm-m_x + m_y- e_ym;
                    end
                end
                if(k<38) begin
                    in = 1;
                end
        end

        if(e_xl< m_x+38 && e_xl>m_x-38 && e_yl <m_y+38 && e_yl >m_y-38)
begin//in the square
                if(e_yl<m_y) begin
                    if(e_xl<m_x)begin
                        x= m_x- e_xl + m_y- e_yl;
                    end
                    else begin
                        x= e_xl-m_x + m_y- e_yl;
                    end
                end
                if(x<38) begin
                    in = 1;
                end
        end

        if(e_xr< m_x+38 && e_xr>m_x-38 && e_yr <m_y+38 && e_yr >m_y-38)
begin//in the square
                if(e_yr<m_y) begin
```

```
                    if(e_xr<m_x)begin
                        y= m_x- e_xr + m_y- e_yr;
                    end
                    else begin
                        y= e_xr-m_x + m_y- e_yr;
                    end
                end
                if(y<38) begin
                    in = 1;
                end
            end
        end
        assign check = in;
endmodule
```

6. IsHit模块-判断我方飞机是否被敌方导弹击中

如果我方飞机被敌方击中，返回1

```
module IsHit(
    input wire clk,
    input wire  [9:0] ebi_x,
    input wire  [8:0] ebi_y,
    input wire  [9:0] mi_x,
    input wire  [8:0] mi_y,
    output wire check
    );
     reg [9:0] e_xm;
     reg [8:0] e_ym;

     reg [9:0] m_x;
     reg [8:0] m_y;

     reg in;

     integer k;

     always @(posedge clk) begin
         e_xm<=ebi_x;
         e_ym<=ebi_y;

         m_x<=mi_x+40;
```

```
        m_y<=mi_y+41;
        in = 0;
        k = 30;
        if(e_xm< m_x+38 && e_xm>m_x-38 && e_ym <m_y+38 && e_ym >m_y-38)
begin//in the square
                if(e_ym<m_y) begin
                    if(e_xm<m_x)begin
                        k= m_x- e_xm + m_y- e_ym;
                    end
                    else begin
                        k= e_xm-m_x + m_y- e_ym;
                    end
                end
                if(k<38) begin
                    in = 1;
                end
        end
    end
    assign check = in;
endmodule
```

五、模拟仿真

1. PS2键盘输入物理仿真

为了了解和判断键盘按动放开的具体输出数据，方便后续电路设计中按键的判断。我们进行了键盘物理验证，以确保PS2模块的正确性。

在键盘物理验证中，我们将该模块接入在之前实验课中制作的disp_num的4位七段数码管显示。这样在下板子时，在四位七段管显示中可以查看键盘输出的具体数据并记录。

test_sp2.v

```
module test_ps2(
    input wire clk_100mhz,
    input ps2_clk,
    input ps2_data,
    input SW,
    output wire [3:0] AN,
```

```
    output wire [7:0] SEGMENT,
    output wire [7:0] LED
    );
   //wire up,down,left,right,space,ctrl;

wire [7:0] data_out;
ps2_ver2 m0(
             .clk(clk_100mhz),
                  .rst(SW),
                  .ps2_clk(ps2_clk),                          //键盘时钟输入
                  .ps2_data(ps2_data),                        //键盘数据输入
                  .data_out(data_out[7:0])                    //键盘扫描骂输出

    );

disp_num m1(     .clk(clk_100mhz),
               .HEXS({8'b0,data_out[7:0]}),
               .LES(4'b0),
               .points(4'b0),
               .RST(1'b0),
               .AN(AN),
               .Segment(SEGMENT)
                     );

assign LED = 8'b11111111;
endmodule
```

在物理验证时，我们测试了设计电路时所需的W、S、A、D、Space键的键值，并发现5个按键按下时，键盘输出的键值与定义的make code相同，证实了PS2 module的逻辑的正确性。

在Top module中，我们要比较PS2键盘的输入和相应的键值是否一致，从而给相应的4位的方向MyP_x、MyP_y赋值。

Top内关于PS2代码

```
always@(posedge clock_20ms)begin
        if(cover==0) begin
                MyP_x <=250;
                MyP_y <=340;
                if(data_out[8]==1'b0)begin
```

```
                if(data_out[7:0]==8'h5a)begin
                    cover=1;
                end
            end
        end
        else begin
            if(data_out[8]==1'b0)begin
                if(finish ==0) begin
                    if(data_out[7:0]==8'h1c&&MyP_x>60)
                        MyP_x <=  MyP_x - 10'd5;
                     if(data_out[7:0]==8'h23&&MyP_x<510)
                        MyP_x <=  MyP_x + 10'd5;
                    if(data_out[7:0]==8'h1d &&MyP_y>10)
                        MyP_y <=  MyP_y - 9'd5;
                    if(data_out[7:0]==8'h1b &&MyP_y<360)
                        MyP_y <=  MyP_y + 9'd5;

                end
                else begin
                    if(data_out[7:0]==8'h29)begin
                        cover = 0;
                    end
                end
            end
        end
    end
```

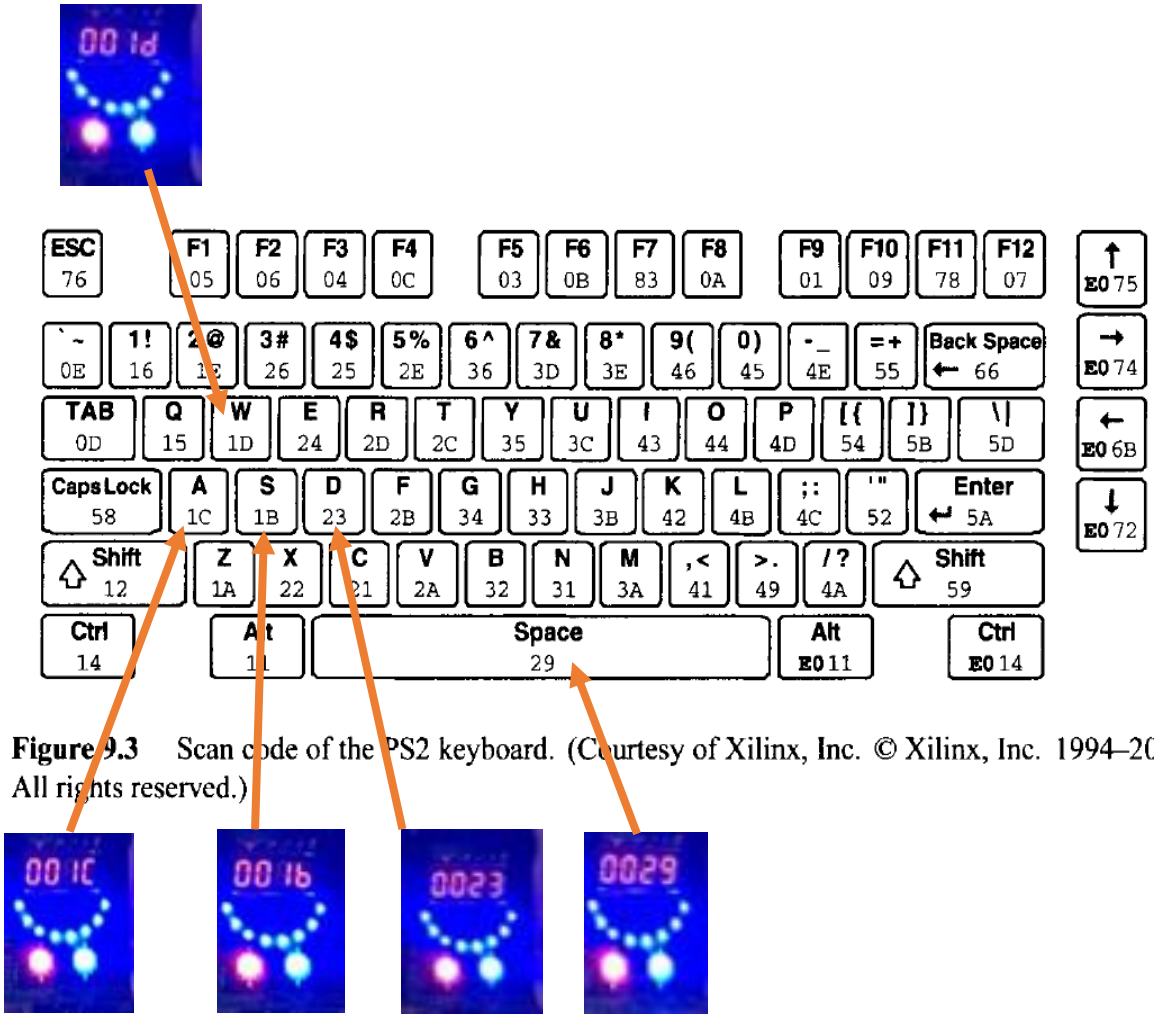


Figure 9.3 Scan code of the PS2 keyboard. (Courtesy of Xilinx, Inc. © Xilinx, Inc. 1994–2007. All rights reserved.)

可见 PS2 模块功能正确。

由于本次的设计中，代码的逻辑部分相对比较复杂，而且涉及到 IP 核内存的读取，采用仿真波形图的方式不太直观。所以所有的调试都通过生成 bit 文件烧录之后，通过 VGA 输出，具体实践来调试，具体的调试过程在上方第四部分已经给出。

六、游戏演示

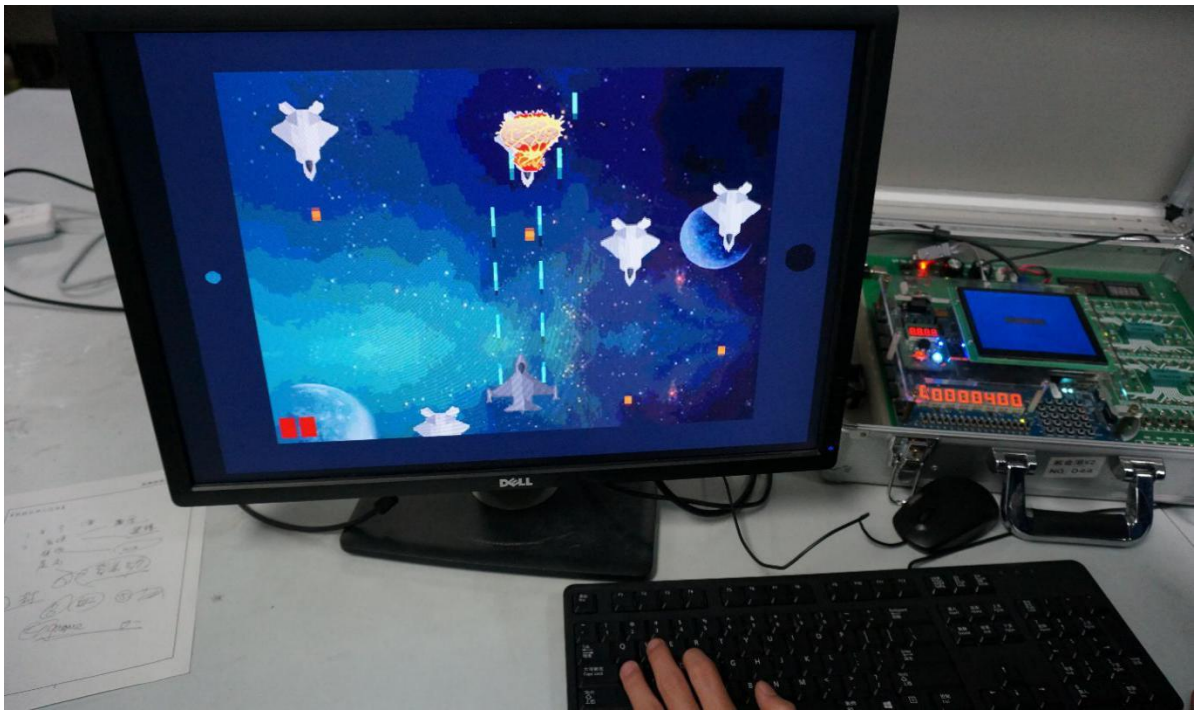
1.游戏初始封面

介绍了游戏名称—“FIGHTER FIGHT”和设计者名称。在游戏开始界面可看到“Missile Ready”字样闪烁，这时若按回车键即可进入游戏。

2.游戏界面

按下Enter键后，进入游戏界面。我方飞机出现，敌方飞机随机产生。背景会随时间缓缓移动，给我们的飞机正在浩瀚宇宙中前行的效果。屏幕左下角记录着我方的血量，最开始为三滴血。

3. 进入游戏后，我方飞机在屏幕下方可以进行上下左右移动。我方飞机发射炮弹为青蓝色，打到敌方飞机10发后敌方飞机爆炸，同时会有爆炸图案出现。同时敌方飞机也发射橘黄色炮弹，击中我方飞机后，我方飞机扣一滴血。

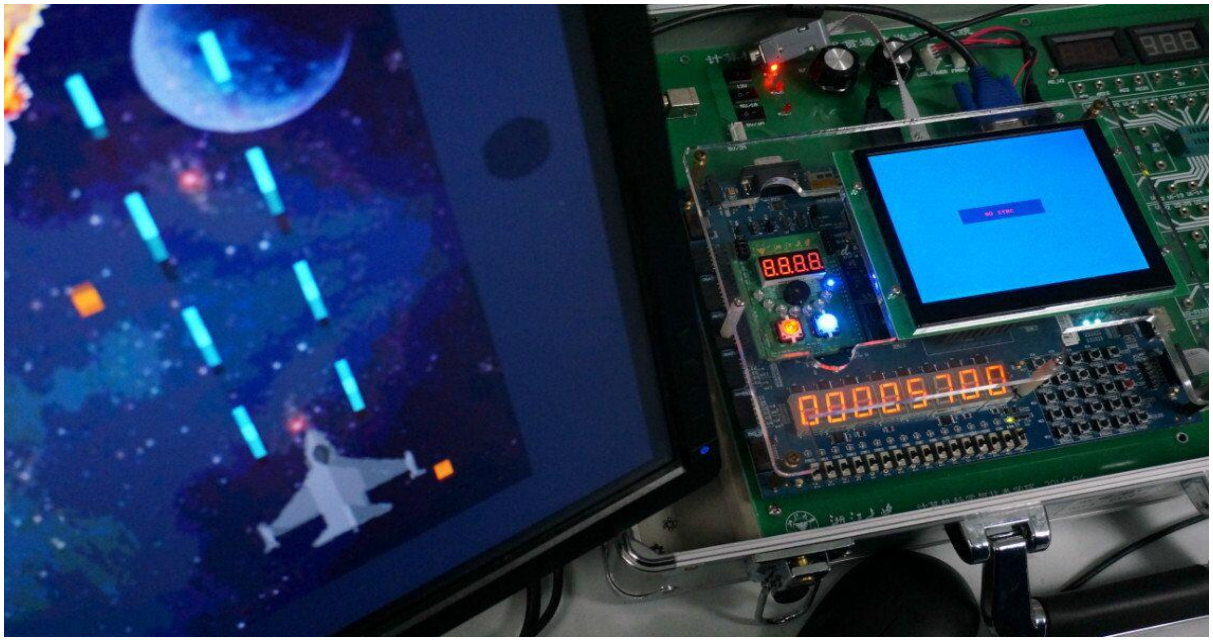


4. 若三滴血都用光，则游戏结束，进入GAME OVER的状态。

5.若我方飞机撞到敌方飞机，则游戏直接结束。

在游戏过程中死亡的话，我方飞机会闪烁两下，进入GAME OVER的状态。此时按下空格键即可重新回到游戏封面。

6.七段数码管会实时记录当前分数。每击落一架敌机，分数加100。可以看到本次游戏击落了飞机57架飞机，获得分数5700分。若此时重新开始游戏，则分数清零。



七、改进思路

1. 设计道具，游戏中飞机可以获得道具。屏幕右下角设计两个道具槽，控制键盘就可以发动道具效果，比如无敌、核弹清屏等等。

2. 加入商店系统，通过购买相关配件和物品升级武器和飞机。

3. 加入装备系统，可以让飞机变得更强悍，替换相关配件让飞机火力全开。

4. 在游戏中添加BOSS。BOSS的攻击丰富多彩,而且攻击的同时还会变形。每个BOSS都有着多样的武器和形态。增加可玩性。

5. 添加高保真的音效，给人以身临其境之感。

八、成员分工

吴君洋：代码逻辑部分，修改PPT，统筹规划 （组长）

林家丰：代码VGA部分，制作视频，VGA部分报告

胡晨旭：代码PS2部分，撰写报告

沈　刻：制作展示PPT前期

成员贡献度：

吴君洋：3170104246　　26%

林家丰：3170106269　　28%

胡晨旭：3170104243　　26%

沈　刻：3170106196　　20%

附：源代码

top.v

```
module top(input clock,
         input rstn,
             input [15:0]SW,
             input ps2_clk,
             input ps2_data,
             output SEGLED_CLK,
             output SEGLED_CLR,
             output SEGLED_DO,
             output SEGLED_PEN,
             output LED_CLK,
             output LED_CLR,
             output LED_DO,
             output LED_PEN,
             output VS,
             output HS,
             output [3:0]R,G,B
             );

    reg [31:0]scores;
    wire [31:0]BCD;
    reg [31:0] clkdiv;
    wire clk,clk_cover;
    reg cover;
    reg finish;
    reg spark;
    initial begin
       cover=0;
       finish=0;
    end
    assign clk= clock&cover;
    assign clk_cover=clock&(~cover)&(~finish);


    reg[18:0] first;
    wire[11:0] spo_first;


    always@(posedge clock)begin
    clkdiv <= clkdiv+1'b1;
```

```
     end//时钟信号
     wire [15:0] SW_OK;
     AntiJitter #(4) a0[15:0](.clk(clkdiv[15]),.I(SW),.O(SW_OK));//防抖动


    wire [31:0] segTestData;
     wire [3:0]sout;
     Seg7Device
segDevice(.clkIO(clkdiv[3]),.clkScan(clkdiv[15:14]),.clkBlink(clkdiv[25]),
     .data(segTestData),.point(8'h0),.LES(8'h0),
     .sout(sout));
     assign SEGLED_CLK = sout[3];
     assign SEGLED_DO = sout[2];
     assign SEGLED_PEN = sout[1];
    assign SEGLED_CLR = sout[0];

     reg [11:0] vga_data;//vga颜色显示
     wire [9:0] col_addr;//x的值
     wire [8:0] row_addr;//y的值

     wire rst;
    vgac
v0(.vga_clk(clkdiv[1]),.clrn(SW_OK[0]),.d_in(vga_data),.row_addr(row_addr),.col_a
ddr(col_addr),.r(R),.g(G),.b(B),.hs(HS),.vs(VS));

     reg [12:0] a1;//我方飞机
     wire [11:0] spo1;
     reg [14:0] gameover;//gameover
     wire [11:0] spo_finish;

     reg [12:0] a3;//敌方飞机
     wire [11:0] spo3;
     reg [12:0] a4;//敌方飞机
     wire [11:0] spo4;
     reg [12:0] a5;//敌方飞机
     wire [11:0] spo5;
     reg [12:0] a6;//敌方飞机
     wire [11:0] spo6;
     reg [12:0] a7;//敌方飞机
     wire [11:0] spo7;

     reg [18:0] bg1;//背景
     wire [11:0] spob;
```

```
reg[18:0] iPad;//ipad边框
wire [11:0] spoi;

reg[12:0] Boom0;//爆炸图片
wire [11:0] spo_boom;
reg[12:0] Boom1;//爆炸图片
wire [11:0] spo_boom1;
reg[12:0] Boom2;//爆炸图片
wire [11:0] spo_boom2;
reg[12:0] Boom3;//爆炸图片
wire [11:0] spo_boom3;
reg[12:0] Boom4;//爆炸图片
wire [11:0] spo_boom4;

/////////ip核模块调用/////////////////////////

begining f1(.addra(first),.douta(spo_first),.clka(clkdiv[1])); //调用封面模块
MyPlane P1(.a(a1),.spo(spo1));     //调用我方飞机模块
game_over g1(.a(gameover),.spo(spo_finish));    //调用游戏结束模块

EnemyPlane e1(.a(a3),.spo(spo3));  //调用敌方飞机模块
EnemyPlane e2(.a(a4),.spo(spo4));
EnemyPlane e3(.a(a5),.spo(spo5));
EnemyPlane e4(.a(a6),.spo(spo6));
EnemyPlane e5(.a(a7),.spo(spo7));

boom B2(.addra(Boom0),.douta(spo_boom),.clka(clkdiv[1]));//调用爆炸模块
boom B3(.addra(Boom1),.douta(spo_boom1),.clka(clkdiv[1]));
boom B4(.addra(Boom2),.douta(spo_boom2),.clka(clkdiv[1]));
boom B5(.addra(Boom3),.douta(spo_boom3),.clka(clkdiv[1]));
boom B6(.addra(Boom4),.douta(spo_boom4),.clka(clkdiv[1]));


ipad i1(.addra(iPad),.douta(spoi),.clka(clkdiv[1]));       //调用ipad边框模块

background b1(.addra(bg1),.douta(spob),.clka(clkdiv[1]));   //调用背景模块




//////////给ip核模块的输入赋值///////////////
always @(posedge clk_cover)begin
```

```
first<=(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)?col_addr*480+row_addr:0;     //给封面赋值
    end

    always@(posedge clock)begin

a1<=(col_addr>=MyP_x&&col_addr<=MyP_x+79&&row_addr>=MyP_y&&row_addr<=MyP_y+79)?(col_addr-MyP_x)*80+(row_addr-MyP_y):0;//给我方飞机赋值

        //给敌方飞机赋值

a3<=(col_addr>=e_x&&col_addr<=e_x+63&&row_addr>=e_y&&row_addr<=e_y+79)?(col_addr-e_x)*80+(row_addr-e_y):0;

a4<=(col_addr>=e_x1&&col_addr<=e_x1+63&&row_addr>=e_y1&&row_addr<=e_y1+79)?(col_addr-e_x1)*80+(row_addr-e_y1):0;

a5<=(col_addr>=e_x2&&col_addr<=e_x2+63&&row_addr>=e_y2&&row_addr<=e_y2+79)?(col_addr-e_x2)*80+(row_addr-e_y2):0;

a6<=(col_addr>=e_x3&&col_addr<=e_x3+63&&row_addr>=e_y3&&row_addr<=e_y3+79)?(col_addr-e_x3)*80+(row_addr-e_y3):0;

a7<=(col_addr>=e_x4&&col_addr<=e_x4+63&&row_addr>=e_y4&&row_addr<=e_y4+79)?(col_addr-e_x4)*80+(row_addr-e_y4):0;

        //给爆炸效果赋值

Boom0<=(col_addr>=bo_x&&col_addr<=bo_x+67&&row_addr>=bo_y&&row_addr<=bo_y+63)?(col_addr-bo_x)*64+(row_addr-bo_y):0;

Boom1<=(col_addr>=bo_x1&&col_addr<=bo_x1+67&&row_addr>=bo_y1&&row_addr<=bo_y1+63)?(col_addr-bo_x1)*64+(row_addr-bo_y1):0;

Boom2<=(col_addr>=bo_x2&&col_addr<=bo_x2+67&&row_addr>=bo_y2&&row_addr<=bo_y2+63)?(col_addr-bo_x2)*64+(row_addr-bo_y2):0;

Boom3<=(col_addr>=bo_x3&&col_addr<=bo_x3+67&&row_addr>=bo_y3&&row_addr<=bo_y3+63)?(col_addr-bo_x3)*64+(row_addr-bo_y3):0;

Boom4<=(col_addr>=bo_x4&&col_addr<=bo_x4+67&&row_addr>=bo_y4&&row_addr<=bo_y4+63)?(col_addr-bo_x4)*64+(row_addr-bo_y4):0;
```

```
iPad<=(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)?col_addr*480+(row_
addr):0;     //给 ipad 输入赋值

gameover<=(col_addr>=g_x&&col_addr<=g_x+191&&row_addr>=g_y&&row_addr<=g_y+39)?(co
l_addr-g_x)*40+row_addr-g_y:0;//给 gameover 输入赋值
        //判定背景移动到何处来对背景进行赋值
        if(row_addr>=BG_Y[8:0])
bg1<=(col_addr>=0&&col_addr<=639&&row_addr>=BG_Y[8:0]&&row_addr<=BG_Y[8:0]+479)?c
ol_addr*480+(row_addr-BG_Y):0;
        else bg1<=(col_addr>=0&&col_addr<=639)?col_addr*480+480-(BG_Y-
row_addr):0;

    end

    ////////////////////////////显示部分//////////////////////////////
    always@(posedge clock)begin
            if(cover==1)begin
                //背景显示
                if(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)
    vga_data<=spob[11:0];

                //我方飞机显示
                if(flash == 1)begin

    if(col_addr>=MyP_x&&col_addr<=MyP_x+79&&row_addr>=MyP_y&&row_addr<=MyP_y+79)b
egin
                        if(spo1[11:0]!=12'hfff)begin
                        vga_data<=spo1[11:0];
                        end
                    end
                end

                //我方子弹显示
                if(col_addr>=B_xl[ 9: 0]&&col_addr<=B_xl[ 9:
0]+3&&row_addr>=B_yl[ 8: 0]&&row_addr<=B_yl[ 8: 0]+19)
vga_data<=12'b1111_1110_0000;
                if(col_addr>=B_xr[ 9: 0]&&col_addr<=B_xr[ 9:
0]+3&&row_addr>=B_yr[ 8: 0]&&row_addr<=B_yr[ 8: 0]+19)
vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[19:10]&&col_addr<=B_xl[19:10]+3&&row_addr>=B_yl[17:
9]&&row_addr<=B_yl[17: 9]+19) vga_data<=12'b1111_1110_0000;
```

```
    if(col_addr>=B_xr[19:10]&&col_addr<=B_xr[19:10]+3&&row_addr>=B_yr[17:
9]&&row_addr<=B_yr[17: 9]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[29:20]&&col_addr<=B_xl[29:20]+3&&row_addr>=B_yl[26:18]&&row
_addr<=B_yl[26:18]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[29:20]&&col_addr<=B_xr[29:20]+3&&row_addr>=B_yr[26:18]&&row
_addr<=B_yr[26:18]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[39:30]&&col_addr<=B_xl[39:30]+3&&row_addr>=B_yl[35:27]&&row
_addr<=B_yl[35:27]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[39:30]&&col_addr<=B_xr[39:30]+3&&row_addr>=B_yr[35:27]&&row
_addr<=B_yr[35:27]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[49:40]&&col_addr<=B_xl[49:40]+3&&row_addr>=B_yl[44:36]&&row
_addr<=B_yl[44:36]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[49:40]&&col_addr<=B_xr[49:40]+3&&row_addr>=B_yr[44:36]&&row
_addr<=B_yr[44:36]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[59:50]&&col_addr<=B_xl[59:50]+3&&row_addr>=B_yl[53:45]&&row
_addr<=B_yl[53:45]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[59:50]&&col_addr<=B_xr[59:50]+3&&row_addr>=B_yr[53:45]&&row
_addr<=B_yr[53:45]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[69:60]&&col_addr<=B_xl[69:60]+3&&row_addr>=B_yl[62:54]&&row
_addr<=B_yl[62:54]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[69:60]&&col_addr<=B_xr[69:60]+3&&row_addr>=B_yr[62:54]&&row
_addr<=B_yr[62:54]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[79:70]&&col_addr<=B_xl[79:70]+3&&row_addr>=B_yl[71:63]&&row
_addr<=B_yl[71:63]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[79:70]&&col_addr<=B_xr[79:70]+3&&row_addr>=B_yr[71:63]&&row
_addr<=B_yr[71:63]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xl[89:80]&&col_addr<=B_xl[89:80]+3&&row_addr>=B_yl[80:72]&&row
_addr<=B_yl[80:72]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[89:80]&&col_addr<=B_xr[89:80]+3&&row_addr>=B_yr[80:72]&&row
_addr<=B_yr[80:72]+19) vga_data<=12'b1111_1110_0000;
```

```
    if(col_addr>=B_xl[99:90]&&col_addr<=B_xl[99:90]+3&&row_addr>=B_yl[89:81]&&row
_addr<=B_yl[89:81]+19) vga_data<=12'b1111_1110_0000;

    if(col_addr>=B_xr[99:90]&&col_addr<=B_xr[99:90]+3&&row_addr>=B_yr[89:81]&&row
_addr<=B_yr[89:81]+19) vga_data<=12'b1111_1110_0000;

                //敌方炮弹显示

    if(col_addr>=eb_x[9:0]&&col_addr<=eb_x[9:0]+7&&row_addr>=eb_y[8:0]&&row_addr<
=eb_y[8:0]+9) vga_data<=12'b0000_0111_1111;

    if(col_addr>=eb_x1[9:0]&&col_addr<=eb_x1[9:0]+7&&row_addr>=eb_y1[8:0]&&row_ad
dr<=eb_y1[8:0]+9) vga_data<=12'b0000_0111_1111;

    if(col_addr>=eb_x2[9:0]&&col_addr<=eb_x2[9:0]+7&&row_addr>=eb_y2[8:0]&&row_ad
dr<=eb_y2[8:0]+9) vga_data<=12'b0000_0111_1111;

    if(col_addr>=eb_x3[9:0]&&col_addr<=eb_x3[9:0]+7&&row_addr>=eb_y3[8:0]&&row_ad
dr<=eb_y3[8:0]+9) vga_data<=12'b0000_0111_1111;

    if(col_addr>=eb_x4[9:0]&&col_addr<=eb_x4[9:0]+7&&row_addr>=eb_y4[8:0]&&row_ad
dr<=eb_y4[8:0]+9) vga_data<=12'b0000_0111_1111;

                //敌方飞机显示

    if(col_addr>=e_x&&col_addr<=e_x+63&&row_addr>=e_y&&row_addr<=e_y+79)begin
                    if(spo3[11:0]!=12'hfff)begin
                        vga_data<=spo3[11:0];
                    end
                end


    if(col_addr>=e_x1&&col_addr<=e_x1+63&&row_addr>=e_y1&&row_addr<=e_y1+79)begin
                    if(spo4[11:0]!=12'hfff)begin
                        vga_data<=spo4[11:0];
                    end
                end


    if(col_addr>=e_x2&&col_addr<=e_x2+63&&row_addr>=e_y2&&row_addr<=e_y2+79)begin
                    if(spo5[11:0]!=12'hfff)begin
                        vga_data<=spo5[11:0];
                    end
                end
```

```
    if(col_addr>=e_x3&&col_addr<=e_x3+63&&row_addr>=e_y3&&row_addr<=e_y3+79)begin
                    if(spo6[11:0]!=12'hfff)begin
                        vga_data<=spo6[11:0];
                    end
                end


    if(col_addr>=e_x4&&col_addr<=e_x4+63&&row_addr>=e_y4&&row_addr<=e_y4+79)begin
                    if(spo7[11:0]!=12'hfff)begin
                        vga_data<=spo7[11:0];
                    end
                end

                //爆炸显示

    if(col_addr>=bo_x&&col_addr<=bo_x+67&&row_addr>=bo_y&&row_addr<=bo_y+63)begin
                    if(spo_boom[11:0]!=12'hfff)begin
                        vga_data<=spo_boom[11:0];
                    end
                end


    if(col_addr>=bo_x1&&col_addr<=bo_x1+67&&row_addr>=bo_y1&&row_addr<=bo_y1+63)b
egin
                    if(spo_boom1[11:0]!=12'hfff)begin
                        vga_data<=spo_boom1[11:0];
                    end
                end


    if(col_addr>=bo_x2&&col_addr<=bo_x2+67&&row_addr>=bo_y2&&row_addr<=bo_y2+63)b
egin
                    if(spo_boom2[11:0]!=12'hfff)begin
                        vga_data<=spo_boom2[11:0];
                    end
                end


    if(col_addr>=bo_x3&&col_addr<=bo_x3+67&&row_addr>=bo_y3&&row_addr<=bo_y3+63)b
egin
                    if(spo_boom3[11:0]!=12'hfff)begin
                        vga_data<=spo_boom3[11:0];
                    end
                end
```

```
    if(col_addr>=bo_x4&&col_addr<=bo_x4+67&&row_addr>=bo_y4&&row_addr<=bo_y4+63)b
egin
                    if(spo_boom4[11:0]!=12'hfff)begin
                        vga_data<=spo_boom4[11:0];
                    end
                end

                //当前生命值显示
                if(hp >2) begin

    if(col_addr>=blood_x2[9:0]&&col_addr<=blood_x2[9:0]+15&&row_addr>=blood_y[8:0
]&&row_addr<=blood_y[8:0]+25) vga_data<=12'b0000_0000_1100;
                end

                if(hp >1) begin

    if(col_addr>=blood_x1[9:0]&&col_addr<=blood_x1[9:0]+15&&row_addr>=blood_y[8:0
]&&row_addr<=blood_y[8:0]+25) vga_data<=12'b0000_0000_1100;
                end

                if(hp >0) begin

    if(col_addr>=blood_x[9:0]&&col_addr<=blood_x[9:0]+15&&row_addr>=blood_y[8:0]&
&row_addr<=blood_y[8:0]+25) vga_data<=12'b0000_0000_1100;
                end

                //ipad
                if(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)begin
//ipad
                  if(spoi[11:0]!=12'hfff)begin
                  vga_data<=spoi[11:0];
                     end
                 end

                 //game over后多显示字符图片
                 if(finish==1)begin

if(col_addr>=g_x&&col_addr<=g_x+191&&row_addr>=g_y&&row_addr<=g_y+39)begin//gameo
ver
                         vga_data<=spo_finish[11:0];
                    end
                 end
```

```
            end//对应 cover==1 时的 always

            else begin  //对应 cover==0 的情况，也就是显示封面
                if(col_addr>=0&&col_addr<=639&&row_addr>=0&&row_addr<=479)
vga_data<=spo_first;
                    if(spark == 1)begin

    if(col_addr>=253&&col_addr<=398&&row_addr>=251&&row_addr<=255)
vga_data<=12'b0001_0001_0011;
                    end
                end
    end//结束显示部分
////////////////////////////////////////////////////////////////////////////////
     //用于检测敌方飞机是否撞到我方飞机
     wire check;
     wire check1;
     wire check2;
     wire check3;
     wire check4;

     //用于检测我方飞机是否被敌方炮弹击中
     wire hit_check;
     wire hit_check1;
     wire hit_check2;
     wire hit_check3;
     wire hit_check4;

     wire clk_10ms;
     wire clock_20ms;

     clk_10ms c1(clock,clk_10ms);
     clk_20ms c2(clock,clock_20ms);

     //游戏结束条件
     always@(posedge clock)begin
        if(hp>0)
            finish=(check | check1 | check2 | check3 | check4);
        else finish = 1;
     end

     reg[9:0] blood_x;
     reg[9:0] blood_x1;
     reg[9:0] blood_x2;
     reg[8:0] blood_y;
```

```
    reg[9:0] MyP_x;    //飞机的横坐标
    reg[8:0] MyP_y;    //飞机的纵坐标

    reg[9:0] g_x;      //gameover 的横坐标
    reg[8:0] g_y;      //gameover 的纵坐标

    reg[9:0] e_x;      //敌方飞机横坐标
    reg[8:0] e_y;      //敌方飞机纵坐标

    reg[9:0] e_x1;
    reg[8:0] e_y1;

    reg[9:0] e_x2;
    reg[8:0] e_y2;

    reg[9:0] e_x3;
    reg[8:0] e_y3;

    reg[9:0] e_x4;
    reg[8:0] e_y4;

    reg[9:0] eb_x;      //敌方炮弹横坐标
    reg[8:0] eb_y;      //敌方炮弹纵坐标

    reg[9:0] eb_x1;
    reg[8:0] eb_y1;

    reg[9:0] eb_x2;
    reg[8:0] eb_y2;

    reg[9:0] eb_x3;
    reg[8:0] eb_y3;

   reg[9:0] eb_x4;
    reg[8:0] eb_y4;

    reg[9:0] bo_x;     //爆炸横坐标
    reg[8:0] bo_y;     //爆炸纵坐标

    reg[9:0] bo_x1;
    reg[8:0] bo_y1;

    reg[9:0] bo_x2;
```

```
    reg[8:0] bo_y2;

    reg[9:0] bo_x3;
    reg[8:0] bo_y3;

    reg[9:0] bo_x4;
    reg[8:0] bo_y4;

    reg[99:0] B_xr;    //100 右子弹的横坐标
    reg[89:0] B_yr;    //100 右子弹的从坐标
    reg[99:0] B_xl;    //100 左子弹的横坐标
    reg[89:0] B_yl;    //100 左子弹的从坐标

    reg[8:0] BG_Y;    //背景的纵坐标

    reg[5:0] Counter;  //飞机受到攻击次数
    reg[5:0] Counter1;
    reg[5:0] Counter2;
    reg[5:0] Counter3;
    reg[5:0] Counter4;

    reg flash;  //飞机闪烁


    integer i=0;
    integer cnt=0;

    integer count = 0;
    integer j=0;
    integer j1=0;
    integer j2=0;
    integer j3=0;
    integer j4=0;

    integer hp=3;

    localparam Left = 15;
    localparam Right = 60;
    localparam Ini = 9'b111_111_111; //子弹初始不可见位置
    localparam Zero = 6'b000_000; //飞机未被攻击的状态
    localparam Height = 500;

    //调用 我方飞机是否被地方飞机撞上 模块
    IsCrash modu(clock, e_x, e_y,    MyP_x, MyP_y , check);
```

```
 IsCrash modu1(clock, e_x1, e_y1, MyP_x, MyP_y ,  check1);
 IsCrash modu2(clock, e_x2, e_y2, MyP_x, MyP_y , check2);
 IsCrash modu3(clock, e_x3, e_y3, MyP_x, MyP_y ,  check3);
 IsCrash modu4(clock, e_x4, e_y4, MyP_x, MyP_y ,  check4);

 //调用 我方飞机是否被敌方飞机导弹击中 模块
 IsHit hit (clock, eb_x, eb_y,   MyP_x, MyP_y , hit_check);
 IsHit hit1(clock, eb_x1,eb_y1,  MyP_x, MyP_y , hit_check1);
 IsHit hit2(clock, eb_x2,eb_y2,  MyP_x, MyP_y , hit_check2);
IsHit hit3(clock, eb_x3,eb_y3,  MyP_x, MyP_y , hit_check3);
IsHit hit4(clock, eb_x4,eb_y4,  MyP_x, MyP_y , hit_check4);


always @(posedge clock_20ms)begin
    if(cover==0) BG_Y<=0;
    else begin
        if(BG_Y<480&&finish==0)       BG_Y<=BG_Y+1;
        else if(BG_Y>=480&&finish==0)  BG_Y<=0;
    end
end

always@(posedge clk_10ms)begin
if (cover == 0) begin
    //敌方飞机初始化
    e_x <= 100;
    e_y <= Ini;

    e_x1 <= 200;
    e_y1 <= Ini;

    e_x2 <= 300;
    e_y2 <= Ini;

    e_x3 <= 400;
    e_y3 <= Ini;

    e_x4 <= 500;
    e_y4 <= Ini;

    //初始化敌人炮弹
    eb_x = 128;
    eb_x1= 228;
    eb_x2= 328;
    eb_x3= 428;
    eb_x4= 528;
```

```
eb_y = Ini;
eb_y1= Ini;
eb_y2= Ini;
eb_y3= Ini;
eb_y4= Ini;

//初始化爆炸
bo_x <=100;
bo_y <=Ini;

bo_x1 <=200;
bo_y1 <=Ini;

bo_x2 <=300;
bo_y2 <=Ini;

bo_x3 <=400;
bo_y3 <=Ini;

bo_x4 <=500;
bo_y4 <=Ini;

//初始化 gameover 坐标
g_x <=224;
g_y <=180;

//血量
hp=3;
blood_x = 65;
blood_x1 =85;
blood_x2 =105;

blood_y =  410;

//将子弹初始化在屏幕看不到的地方
B_yr[ 8: 0] <= Ini;
B_yl[ 8: 0] <= Ini;
B_yr[17: 9] <= Ini;
B_yl[17: 9] <= Ini;
B_yr[26:18] <= Ini;
B_yl[26:18] <= Ini;
B_yr[35:27] <= Ini;
B_yl[35:27] <= Ini;
```

```
        B_yr[44:36] <= Ini;
        B_yl[44:36] <= Ini;
        B_yr[53:45] <= Ini;
        B_yl[53:45] <= Ini;
        B_yr[62:54] <= Ini;
        B_yl[62:54] <= Ini;
        B_yr[71:63] <= Ini;
        B_yl[71:63] <= Ini;
        B_yr[80:72] <= Ini;
        B_yl[80:72] <= Ini;
        B_yr[89:81] <= Ini;
        B_yl[89:81] <= Ini;

        //将敌方飞机血量初始化
        Counter =Zero;
        Counter1 = Zero;
        Counter2 = Zero;
        Counter3 = Zero;
        Counter4 = Zero;

        //分数初始化
        scores=64'd00000000;

        if(count<100) begin
            count = count + 1;
            if(count <50) spark =1;
            else spark =0;
        end
        else count = 0;
    end
    else begin
    if (finish==0)begin
        cnt=0;
       i=i+1;
        flash = 1;
        if(i==100) i=0;
        case(i)
        0:begin B_xr[ 9: 0] <= MyP_x+Right;
               B_yr[ 8: 0] <= MyP_y+20;
               B_xl[ 9: 0] <= MyP_x+Left;
               B_yl[ 8: 0] <= MyP_y+20;
               end
        10:begin B_xr[19:10] <= MyP_x+Right;
               B_yr[17: 9] <= MyP_y+20;
               B_xl[19:10] <= MyP_x+Left;
```

```
            B_yl[17: 9] <= MyP_y+20;
            end
        20:begin B_xr[29:20] <= MyP_x+Right;
            B_yr[26:18] <= MyP_y+20;
            B_xl[29:20] <= MyP_x+Left;
            B_yl[26:18] <= MyP_y+20;
            end
        30:begin B_xr[39:30] <= MyP_x+Right;
            B_yr[35:27] <= MyP_y+20;
            B_xl[39:30] <= MyP_x+Left;
            B_yl[35:27] <= MyP_y+20;
            end
        40:begin B_xr[49:40] <= MyP_x+Right;
            B_yr[44:36] <= MyP_y+20;
            B_xl[49:40] <= MyP_x+Left;
            B_yl[44:36] <= MyP_y+20;
            end
        50:begin B_xr[59:50] <= MyP_x+Right;
            B_yr[53:45] <= MyP_y+20;
            B_xl[59:50] <= MyP_x+Left;
            B_yl[53:45] <= MyP_y+20;
            end
        60:begin B_xr[69:60] <= MyP_x+Right;
            B_yr[62:54] <= MyP_y+20;
            B_xl[69:60] <= MyP_x+Left;
            B_yl[62:54] <= MyP_y+20;
            end
        70:begin B_xr[79:70] <= MyP_x+Right;
            B_yr[71:63] <= MyP_y+20;
            B_xl[79:70] <= MyP_x+Left;
            B_yl[71:63] <= MyP_y+20;
            end
        80:begin B_xr[89:80] <= MyP_x+Right;
            B_yr[80:72] <= MyP_y+20;
            B_xl[89:80] <= MyP_x+Left;
            B_yl[80:72] <= MyP_y+20;
            end
        90:begin B_xr[99:90] <= MyP_x+Right;
            B_yr[89:81] <= MyP_y+20;
            B_xl[99:90] <= MyP_x+Left;
            B_yl[89:81] <= MyP_y+20;
            end
        endcase
```

```
//飞机产生&敌人炮弹产生
if( clkdiv[20:13]==0 && e_y>Height) begin
    e_y<=0;
    if(eb_y >Height) begin
        eb_y = 45;
    end

end
if( clkdiv[17:10]==50 && e_y1>Height) begin
    e_y1<=0;
    if(eb_y1 >Height) begin
        eb_y1 = 30;
    end
end
if( clkdiv[19:12]==100 && e_y2>Height) begin
    e_y2<=0;
    if(eb_y2 >Height) begin
        eb_y2 = 13;
    end
end
if( clkdiv[18:11]==150 && e_y3>Height) begin
    e_y3<=0;
    if(eb_y3 >Height) begin
        eb_y3 = 27;
    end
end
if( clkdiv[21:14]==200 && e_y4>Height) begin
    e_y4<=0;
    if(eb_y4 >Height) begin
        eb_y4 = 40;
    end
end

//让子弹飞
if(eb_y < Height+1) begin
    if(hit_check==0)
        eb_y = eb_y +2;
    else begin
        eb_y =Ini;
        hp = hp - 1;
    end
end

if(eb_y1 < Height+1) begin
```

```
            if(hit_check1==0)
                eb_y1 = eb_y1 +2;
            else begin
                eb_y1 =Ini;
                hp = hp - 1;
            end
        end
        if(eb_y2 < Height+1) begin
            if(hit_check2==0)
                eb_y2 = eb_y2 +2;
            else begin
                eb_y2 =Ini;
                hp = hp - 1;
            end
        end
        if(eb_y3 < Height+1) begin
            if(hit_check3==0)
                eb_y3 = eb_y3 +2;
            else begin
                eb_y3 =Ini;
                hp = hp -1;
            end
        end
        if(eb_y4 < Height+1) begin
            if(hit_check4==0)
                eb_y4 = eb_y4 +2;
            else begin
                eb_y4 =Ini;
                hp= hp - 1;
            end
        end

        //敌方飞机状态检测 0
        if(Counter >=10)begin
            bo_y <= e_y;
            if(j<30) begin
                j=j+1;
            end
            else begin
                bo_y <= Ini;
                e_y <=Ini;
                Counter =0;
                scores=scores+100;
                j=0;
            end
```

```
        end
        else if(e_y[8:0]< Ini && Counter <10)begin
            e_y[8:0]<=e_y[8:0]+1;
        end


        //敌方飞机状态检测 1
        if(Counter1 >=10)begin
            bo_y1 <= e_y1;
            if(j1<30) begin
                j1=j1+1;
            end
            else begin
                bo_y1 <= Ini;
                e_y1 <=Ini;
                Counter1 =0;
                scores=scores+100;
                j1=0;
            end
        end
        else if(e_y1[8:0]< Ini && Counter1 <10) begin
            e_y1[8:0]<=e_y1[8:0]+1;
        end

        //敌方飞机状态检测 2
        if(Counter2 >=10)begin
            bo_y2 <= e_y2;
            if(j2<30) begin
                j2=j2+1;
            end
            else begin
                bo_y2 <= Ini;
                e_y2 <=Ini;
                Counter2 =0;
                scores=scores+100;
                j2=0;
            end
        end
        else if(e_y2[8:0]< Ini && Counter2 <10) begin
            e_y2[8:0]<=e_y2[8:0]+1;
        end

        //敌方飞机状态检测 3
        if(Counter3 >=10)begin
            bo_y3 <= e_y3;
```

```
            if(j3<30) begin
                j3=j3+1;
            end
            else begin
                bo_y3 <= Ini;
                e_y3 <=Ini;
                Counter3 =0;
                scores=scores+100;
                j3=0;
            end
        end
        else if(e_y3[8:0]< Ini && Counter3 <10) begin
            e_y3[8:0]<=e_y3[8:0]+1;
        end

        //敌方飞机状态检测 4
        if(Counter4 >=10)begin
            bo_y4 <= e_y4;
            if(j4<30) begin
                j4=j4+1;
            end
            else begin
                bo_y4 <= Ini;
                e_y4 <=Ini;
                Counter4 =0;
                scores=scores+100;
                j4=0;
            end
        end
        else if(e_y4[8:0]< Ini && Counter4 <10)begin
            e_y4[8:0]<=e_y4[8:0]+1;
        end

        if(B_yl[8:0]<9'b111_111_100)begin
          if(B_yl[8:0]<=3)begin
                B_yl[8:0] <= Ini;
          end

            else if((MyP_y[8:0]>e_y[8:0])&&(B_xl[9:0]<e_x[9:0]+64 &&
B_xl[9:0]>e_x[9:0])&& (B_yl[8:0]<e_y[8:0]+40)) begin
                Counter = Counter +1;
                B_yl[8:0] <= Ini;
            end
            else if((MyP_y[8:0]>e_y1[8:0])&&(B_xl[9:0]<e_x1[9:0]+64 &&
B_xl[9:0]>e_x1[9:0])&& (B_yl[8:0]<e_y1[8:0]+40)) begin
```

```
                Counter1 = Counter1 +1;
                B_yl[8:0] <= Ini;
            end
            else if((MyP_y[8:0]>e_y2[8:0])&&(B_xl[9:0]<e_x2[9:0]+64 &&
B_xl[9:0]>e_x2[9:0])&& (B_yl[8:0]<e_y2[8:0]+40)) begin
                Counter2 = Counter2 +1;
                B_yl[8:0] <= Ini;
            end
            else if((MyP_y[8:0]>e_y3[8:0])&&(B_xl[9:0]<e_x3[9:0]+64 &&
B_xl[9:0]>e_x3[9:0])&& (B_yl[8:0]<e_y3[8:0]+40)) begin
                Counter3 = Counter3 +1;
                B_yl[8:0] <= Ini;
            end
            else if((MyP_y[8:0]>e_y4[8:0])&&(B_xl[9:0]<e_x4[9:0]+64 &&
B_xl[9:0]>e_x4[9:0])&& (B_yl[8:0]<e_y4[8:0]+40)) begin
                Counter4 = Counter4 +1;
                B_yl[8:0] <= Ini;
            end



          else begin
          B_yl[8:0]<=B_yl[8:0]-9'd6;
          end
        end

        if(B_yr[8:0]<9'b111_111_100)begin
          if(B_yr[8:0]<=3)begin
                B_yr[8:0] <= Ini;
          end

            else if ((MyP_y[8:0]>e_y[8:0])&&(B_xr[9:0]<e_x[9:0]+64 &&
B_xr[9:0]>e_x[9:0])&& (B_yr[8:0]<e_y[8:0]+40))begin
                Counter =Counter +1;
                B_yr[8:0] <= Ini;
            end
            else if ((MyP_y[8:0]>e_y1[8:0])&&(B_xr[9:0]<e_x1[9:0]+64 &&
B_xr[9:0]>e_x1[9:0])&& (B_yr[8:0]<e_y1[8:0]+40))begin
                Counter1 =Counter1 +1;
                B_yr[8:0] <= Ini;
            end
            else if ((MyP_y[8:0]>e_y2[8:0])&&(B_xr[9:0]<e_x2[9:0]+64 &&
B_xr[9:0]>e_x2[9:0])&& (B_yr[8:0]<e_y2[8:0]+40))begin
                Counter2 =Counter2 +1;
                B_yr[8:0] <= Ini;
            end
```

```
            else if ((MyP_y[8:0]>e_y3[8:0])&&(B_xr[9:0]<e_x3[9:0]+64 &&
B_xr[9:0]>e_x3[9:0])&& (B_yr[8:0]<e_y3[8:0]+40))begin
                Counter3 =Counter3 +1;
                B_yr[8:0] <= Ini;
            end
            else if ((MyP_y[8:0]>e_y4[8:0])&&(B_xr[9:0]<e_x4[9:0]+64 &&
B_xr[9:0]>e_x4[9:0])&& (B_yr[8:0]<e_y4[8:0]+40))begin
                Counter4 =Counter4 +1;
                B_yr[8:0] <= Ini;
            end


          else begin
          B_yr[8:0]<=B_yr[8:0]-9'd6;
          end
        end
        //finish
        //判断一共十次，限于篇幅，只展示一次的
         //finish 10
    end// end if row:449
        else begin
            if(cnt < 200)begin
                cnt=cnt+1;
                if(cnt <50) flash =0;
                else if(cnt <100) flash = 1;
                else if(cnt <150) flash = 0;
                else flash =1;
            end
        end
    end
    end//end always row:448



   wire [9:0] data_out;
     wire ready;
     ps2_ver2 ps2(.clk(clock),.rst(SW[15]),.ps2_clk(ps2_clk),.ps2_data(ps2_data),
.data_out(data_out[9:0]),.ready(ready));

     always@(posedge clock_20ms)begin
        if(cover==0) begin
                MyP_x <=250;
                MyP_y <=340;
                if(data_out[8]==1'b0)begin
                    if(data_out[7:0]==8'h5a)begin
```

```
                        cover=1;
                    end
                end
        end
        else begin
            if(data_out[8]==1'b0)begin
                if(finish ==0) begin
                    if(data_out[7:0]==8'h1c&&MyP_x>60)
                        MyP_x <=  MyP_x - 10'd5;
                     if(data_out[7:0]==8'h23&&MyP_x<510)
                        MyP_x <=  MyP_x + 10'd5;
                    if(data_out[7:0]==8'h1d &&MyP_y>10)
                        MyP_y <=  MyP_y - 9'd5;
                    if(data_out[7:0]==8'h1b &&MyP_y<360)
                        MyP_y <=  MyP_y + 9'd5;

                end
                else begin
                    if(data_out[7:0]==8'h29)begin
                        cover = 0;
                    end
                end
            end
        end
    end

    BCD b9(scores,BCD);
    assign segTestData={BCD};
    wire [15:0]ledData;
    assign ledData = SW_OK;
    ShiftReg
#(.WIDTH(16))ledDevice(.clk(clkdiv[3]),.pdata(~ledData),.sout({LED_CLK,LED_DO,
LED_PEN,LED_CLR}));

endmodule
```

bmptocoe.c

```
#define _CRT_SECURE_NO_DEPRECATE

#include<windows.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

```
FILE* outfp;
int carry2;
int BMPtoCOE(char *BMPfilename, int isfinished)
{
    FILE *fp;
    if ((fp = fopen(BMPfilename, "rb")) == NULL)
    {
        printf("在该路径下找不到该图片\n");
        return 1;
    }
    char bm[2];
    fseek(fp, sizeof(BITMAPFILEHEADER), 0);
    BITMAPINFOHEADER head;
    fread(&head, sizeof(BITMAPINFOHEADER), 1, fp);
    int picwidth = head.biWidth, picheight = head.biHeight;
    short ind;
    fseek(fp, 0L, 0);
    fread(bm, 2, 1, fp);
    fseek(fp, 0x1CL, 0);
    fread(&ind, 2, 1, fp);
    if (bm[0] != 'B' || bm[1] != 'M' || ind != 32)
    {
        printf("该图片不是 32 位的位图\n");
        return 1;
    }
    if (picheight<0) picheight = -picheight;
    fseek(fp, 0x36L, 0);
    printf("图片尺寸%d*%d\n", picwidth, picheight);
    int buf = (picwidth * 3 % 4) ? 4 - (picwidth * 3) % 4 : 0;
    char *tmp = (char*)malloc(sizeof(char)*buf);
    int i, j;
    unsigned char r, g, b, t;
    for (i = 0; i <picheight; i++)
        for (j = 0; j < picwidth; j++)
        {
            fread(&b, 1, 1, fp);
            fread(&g, 1, 1, fp);
            fread(&r, 1, 1, fp);
            fprintf(outfp, "%3x%x%x", (int)(b / 256.0 * 16), (int)(g / 256.0 *
16), (int)(r / 256.0 * 16));
            if (i == picheight - 1 && j == picwidth - 1 && isfinished)
fprintf(outfp, ";");
            else fprintf(outfp, ",");
            carry2++;
            if (carry2 == 16) fprintf(outfp, "\n"), carry2 = 0;
```

```
            fread(&t, 1, 1, fp);
        }
    free(tmp);
    fclose(fp);
    return 0;
}
int main()
{
    outfp = fopen("D:/background.coe", "w");
    int i;
    char s[20] = "background.bmp";
    fprintf(outfp, "memory_initialization_radix=16;\nmemory_initialization_vector
=\n");
    BMPtoCOE(s, 1);
    system("pause");
    return 0;
}
```

Ucf 文件：

```
NET"clock" LOC = AC18 | IOSTANDARD = LVCMOS18;
NET"rstn" LOC = W13 | IOSTANDARD = LVCMOS18;

NET"ps2_clk" LOC = N18 | IOSTANDARD = LVCMOS33 | PULLUP;
NET"ps2_data" LOC = M19 | IOSTANDARD = LVCMOS33 | PULLUP;

NET"SW[0]"LOC = AA10 | IOSTANDARD = LVCMOS15;
NET"SW[1]"LOC = AB10 | IOSTANDARD = LVCMOS15;
NET"SW[2]"LOC = AA13 | IOSTANDARD = LVCMOS15;
NET"SW[3]"LOC = AA12 | IOSTANDARD = LVCMOS15;
NET"SW[4]"LOC = Y13 | IOSTANDARD = LVCMOS15;
NET"SW[5]"LOC = Y12 | IOSTANDARD = LVCMOS15;
NET"SW[6]"LOC = AD11 | IOSTANDARD = LVCMOS15;
NET"SW[7]"LOC = AD10 | IOSTANDARD = LVCMOS15;
NET"SW[8]"LOC = AE10 | IOSTANDARD = LVCMOS15;
NET"SW[9]"LOC = AE12 | IOSTANDARD = LVCMOS15;
NET"SW[10]"LOC = AF12 | IOSTANDARD = LVCMOS15;
NET"SW[11]"LOC = AE8 | IOSTANDARD = LVCMOS15;
NET"SW[12]"LOC = AF8 | IOSTANDARD = LVCMOS15;
NET"SW[13]"LOC = AE13 | IOSTANDARD = LVCMOS15;
NET"SW[14]"LOC = AF13 | IOSTANDARD = LVCMOS15;
NET"SW[15]"LOC = AF10 | IOSTANDARD = LVCMOS15;
```

```
NET"b[0]" LOC = T20 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"b[1]" LOC = R20 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"b[2]" LOC = T22 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"b[3]" LOC = T23 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"g[0]" LOC = R22 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"g[1]" LOC = R23 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"g[2]" LOC = T24 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"g[3]" LOC = T25 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"r[0]" LOC = N21 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"r[1]" LOC = N22 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"r[2]" LOC = R21 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"r[3]" LOC = P21 | IOSTANDARD = LVCMOS33 |SLEW = FAST;

NET"HS" LOC = M22 | IOSTANDARD = LVCMOS33 |SLEW = FAST;
NET"VS" LOC = M21 | IOSTANDARD = LVCMOS33 |SLEW = FAST;

NET"SEGLED_CLK"LOC = M24 | IOSTANDARD = LVCMOS33;
NET"SEGLED_CLR"LOC = M20 | IOSTANDARD = LVCMOS33;
NET"SEGLED_DO"LOC = L24 | IOSTANDARD = LVCMOS33;
NET"SEGLED_PEN"LOC = R18 | IOSTANDARD = LVCMOS33;

NET"LED_CLK"LOC = N26 | IOSTANDARD = LVCMOS33;
NET"LED_CLR"LOC = N24 | IOSTANDARD = LVCMOS33;
NET"LED_DO"LOC = M26 | IOSTANDARD = LVCMOS33;
NET"LED_PEN"LOC = P18 | IOSTANDARD = LVCMOS33;
```