****

****

**题目：基于数字系统的“抢座位大作战”游戏设计**

|  |  |  |
| --- | --- | --- |
| 组员姓名： | 焦笑然 | 宋天琪 |
| 完成时间： | 2020-1-5 | |

摘要

本次课程设计的内容为一款基于数字系统的原创游戏，题目为“抢座位大作战”。报告主要介绍了关于游戏设计的背景、功能、具体实现、结果验证与展望等要素，并对课题完成过程中所遇到的困难和解决方法做了分析，最后分享了小组成员课程学习的收获。

关键词：数字逻辑， 游戏， Verilog， IP Core， FPGA

目录

[一、游戏背景 4](#_Toc29323320)

[二、设计思路 4](#_Toc29323321)

[2.1 理论基础 4](#_Toc29323322)

[2.1.1 VGA显示 4](#_Toc29323323)

[2.1.2 PS/2键盘 5](#_Toc29323324)

[2.1.3有限状态机 6](#_Toc29323325)

[2.1.4硬件描述语言 7](#_Toc29323326)

[2.2 实验器材 7](#_Toc29323327)

[2.2.1 ISE软件 7](#_Toc29323328)

[2.2.2 SWORD板 7](#_Toc29323329)

[2.2.3计算机 8](#_Toc29323330)

[2.2.4 Adobe Photoshop软件 8](#_Toc29323331)

[2.2.5 Dev C++软件 8](#_Toc29323332)

[2.3 设计方案 9](#_Toc29323333)

[2.3.1 整体设计方案 9](#_Toc29323334)

[2.3.2逻辑核心模块设计 9](#_Toc29323335)

[三、实现过程 10](#_Toc29323336)

[3.1 图片导入 10](#_Toc29323337)

[3.1.1绘图 10](#_Toc29323338)

[3.1.2 PS处理 11](#_Toc29323339)

[3.1.3 生成COE文件 12](#_Toc29323340)

[3.1.4 生成ip核 12](#_Toc29323341)

[3.2 VGA显示 14](#_Toc29323342)

[3.3 PS2键盘输入 14](#_Toc29323343)

[3.4 七段数码管显示 17](#_Toc29323344)

[3.5 Top模块 18](#_Toc29323345)

[四、演示效果 25](#_Toc29323346)

[五、问题与改进思路 28](#_Toc29323347)

[5.1实现过程中的问题 28](#_Toc29323348)

[5.1.1内存不足的问题 28](#_Toc29323349)

[5.1.2解决方法 28](#_Toc29323350)

[5.2改进思路 28](#_Toc29323351)

[六、成员分工 29](#_Toc29323352)

[附录：源代码 29](#_Toc29323353)

# 一、游戏背景

游戏的灵感来自于小组成员日常生活的亲身感受。学校的一些大类课程选课人数众多，每次课前，求知心切的浙大学子总会提前到教室占座位，有时甚至会出现“抢座位”的现象。具体表现为教室的门打开后，早已在门外等候的学子蜂拥而入，寻找空余的位置，并用书包或者课本放在桌面上标记占座。有些同学甚至会一个人占多个座位。基于个人的亲身体会，设计者将平日里的教室抢座位场景进行抽象化，设计出“抢座位大作战”这款游戏。



图表1 教室占座位场景

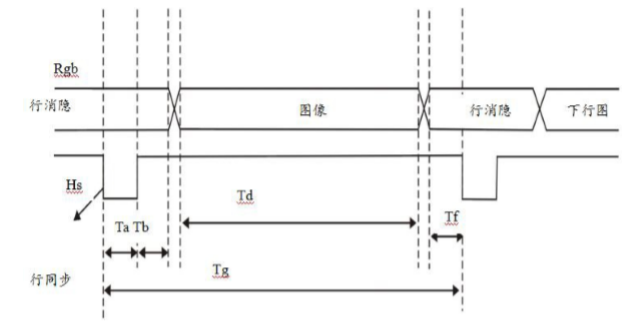
# 二、设计思路

## 2.1 理论基础

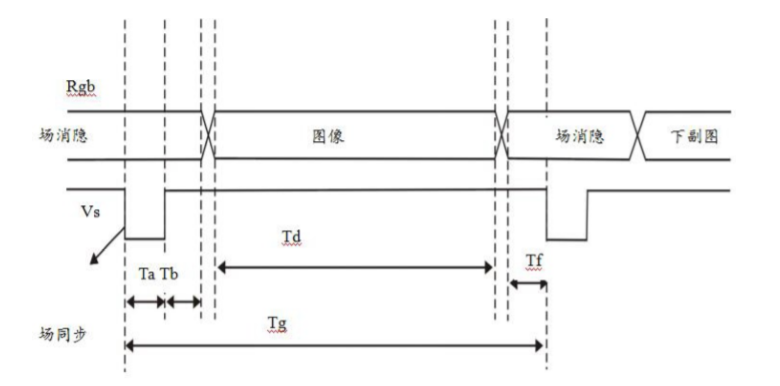
### 2.1.1 VGA显示

VGA( Video Graphics Array)作为一种标准的显示接口得到了广泛的应用，其信号类型为模拟类型，显示卡端的接口为15针母插座。VGA在任何时刻都必须工作在某一显示模式之下，其显示模式分为字符显示模式和图形显示模式，在应用中，讨论的都是图形显示模式。工业标准的VGA显示模式为：640\* 480\* 16 色\* 60Hz。

VGA 时序控制模块是本设计的重要部分，最终的输出信号行、场同步信号必须严格按照 VGA 时序标准产生相应的脉冲信号。在5个信号时序驱动时，时序信号包括前两个，它们都有图像显示区和图像消隐区，图像消隐区又分为消隐前肩、同步脉冲区和消隐后肩。



图表2 VGA行扫描时序图



图表3 VGA场扫描时序图

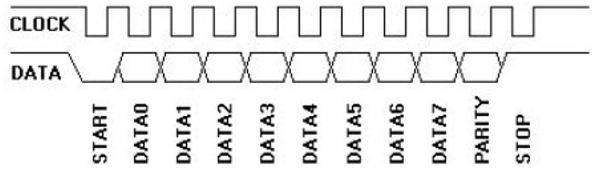
### 2.1.2 PS/2键盘

PS/2通信协议是一种双向同步串行通信协议。通信的两端通过CLOCK(时钟脚)同步，并通过DATA(数据脚)交换数据。一般两台设备间传输数据的最大时钟频率是33kHz，大多数PS/2设备工作在10--20kHz。推荐值在15kHz左右，也就是说，CLOCK高、低电平的持续时间都为40us。每一数据帧包含11—12位，具体含义如下图所示。

图表4 PS/2 通信数据帧格式

|  |  |
| --- | --- |
| 数据 | 含义 |
| 1个起始位 | 总是逻辑0 |
| 8个数据位 | (LSB)低位在前 |
| 1个奇偶校验位 | 奇校验 |
| 1个停止位 | 总是逻辑1 |
| 1个应答位 | 仅用在主机对设备的通信中 |

PS/2 到主机的通信时序如下图所示。数据在PS/2时钟的下降沿读取，PS/2的时钟频率为10—16.7kHz。对于PS/2设备，一般来说从时钟脉冲的上升沿到一个数据转变的时间至少要有 5us；数据变化到下降沿的时间至少要有5us，并且不大于25us，这个时序非常重要应该严格遵循。主机可以在第11个时钟脉冲停止位之前把时钟线拉低，使设备放弃发送当前字节，当然这种情况比较少见。在停止位发送后设备在发送下个包前应该至少等待50us，给主机时间做相应的处理。不过主机处理接收到的字节时一般会抑制发送(主机在收到每个包时通常自动做这个)。在主机释放抑制后，设备至少应该在发送任何数据前等50us。



图表5 设备到主机的通信

### 2.1.3有限状态机

有限状态机（Finite State Machine FSM）是时序电路设计中经常采用的一种方式，尤其适合设计数字系统的控制模块，在一些需要控制高速器件的场合，用状态机进行设计是一 种很好的解决问题的方案，具有速度快、结构简单、可靠性高等优点。有限状态机非常适合 用FPGA器件实现，用Verilog HDL的case语句能很好地描述基于状态机的设计，再通过 EDA工具软件的综合，一般可以生成性能极优的状态机电路，从而使其在执行时间、运行速度和占用资源等方面优于用CPU实现的方案。

### 2.1.4硬件描述语言

Verilog HDL 是一种硬件描述语言（HDL: Hardware Description Language），以文本形式来描述数字系统硬件的结构和行为的语言，用它可以表示逻辑电路图、逻辑表达式，还可以表示数字逻辑系统所完成的逻辑功能。使用Verilog描述硬件的基本设计单元是模块（module）。构建复杂的电子电路，主要是通过模块的相互连接调用来实现的。模块被包含在关键字module、endmodule之内。实际的电路元件。Verilog中的模块类似C语言中的函数，它能够提供输入、输出端口，可以实例调用其他模块，也可以被其他模块实例调用。

## 2.2 实验器材

### 2.2.1 ISE软件

ISE是使用XILINX的FPGA的必备的设计工具。它可以完成FPGA开发的全部流程，包括设计输入、仿真、综合、布局布线、生成 BIT 文件、配置以及在线调试等，功能非常强大。ISE除了功能完整，使用方便外，它的设计性能也非常好，以集成的时序收敛流程整合了增强性物理综合优化，提供最佳的时钟布局、更好的封装和时序收敛映射，从而获得更高的设计性能。

### 2.2.2 SWORD板

采用开放式体系构架和32位存储层次结构，通用性强，适用性灵活。实验方法采用基 于FPGA实体的虚拟实验箱和SOC集成技术，支持个性化开发、课程设计和创新实践。系统资源可很好地支持数字电路、计算机组成、计算机体系结构、接口通讯、编译技术、操作 系统、计算机网络、基于IP核的嵌入式系统和多媒体等课程的教学实践。

### 2.2.3计算机

### 2.2.4 Adobe Photoshop软件

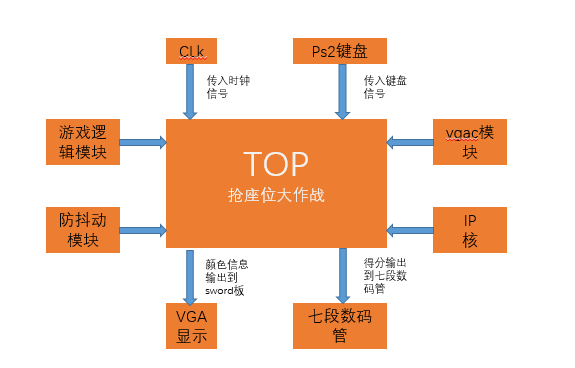
进行图片编辑工作。包括编辑图片内容、修改图片大小、生成对应像素的BMP格式文件。

### 2.2.5 Dev C++软件

将BMP格式图片转换为COE格式文件，以便后续产生IP核。

## 2.3 设计方案

### 2.3.1 整体设计方案



### 2.3.2逻辑核心模块设计

与其他复刻游戏不同，“抢座位大作战”是一款完全的原创游戏，首先需明确其运行机制，使其能够被基于数字系统的硬件语言描述，并且在可执行的过程中不失竞技和趣味性。

“抢座位大作战”游戏支持双人对战，玩家的角色会持续地在走廊区域移动，寻找最合适的桌子位置，一旦发现，可以通过按键触发投掷书本，进行“占座”。当所有座位都被占时，游戏结束，占座数量更多的玩家获得胜利。

实际生活中，临近的两个同学都想占同一个座位，只有出手果决的同学才能成功占座，这是一个兼具考验决策能力与速度的过程。为贴合现实，游戏同样设计了竞争机制：

1）**停下等待。**书本投掷出去后，玩家的角色会停下等待。当书到达桌子或者飞出屏幕后，角色才会继续移动。

2）**投掷精确度。**扔出的书只有在一定精确度内才能占座。如果偏差较大，即使碰到桌子也无效。

3）**占座不同时。**两个玩家对战时，一次只能有一人“占座”，即一次只能一人按键。如果两人同时按键，则两个按键无效，两人都不能占座。一个玩家的书本扔出到达座位之间的这段时间里，另一个玩家可以按键“占座”。

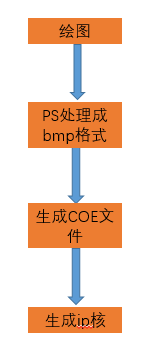
4）**占座取消。**在书本扔出到达座位或飞出屏幕的这段时间里，如果玩家再次按键，则会取消已经扔出的书本，使自己恢复移动能力。

5）**打断对方。**玩家可以在对方投掷出书本后使用另一按键出手“打断”，使得对方扔出的书无效。在此期间，双方的移动不受影响。

6）**改变速度。**玩家在走廊中移动的速度可以通过键盘调整，共有四档速度可以选择。

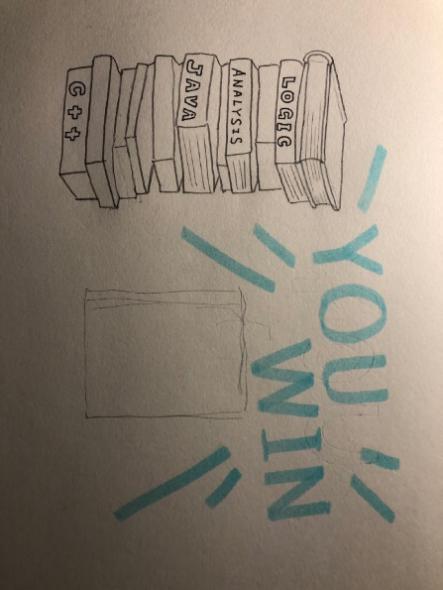
# 三、实现过程

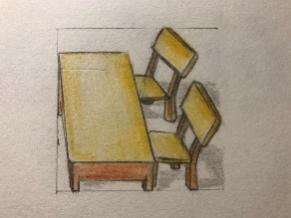
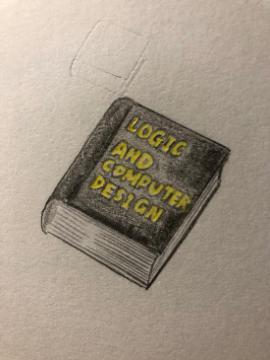
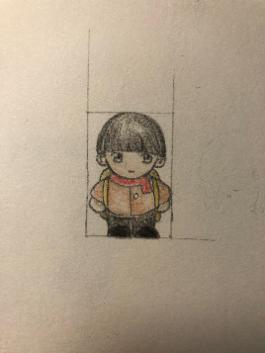
## 3.1 图片导入



### 3.1.1绘图

由于“抢座位大作战”是完全原创的游戏，因此游戏中出现的所有图片都为手绘原创，其中部分形象参考了浙江大学西区教室照片。

### 3.1.2 PS处理

通过PS对图片的亮度和对比度进行处理，存储为bmp格式图形，深度选择32位。





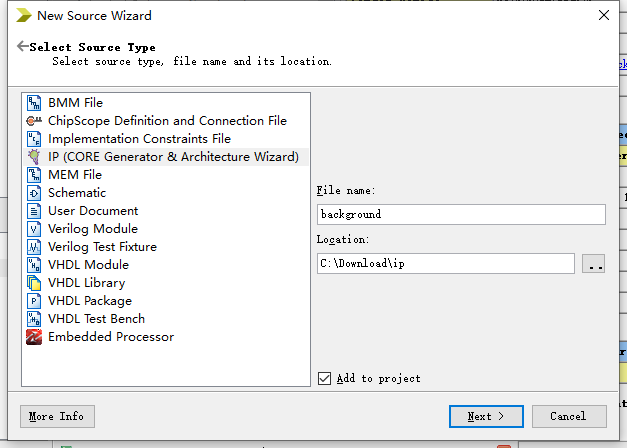
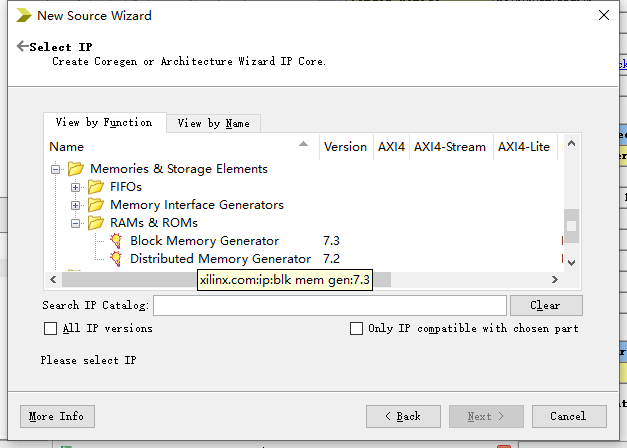
  C:\Users\123456\Desktop\program图片\girlwithback.bmp C:\Users\123456\Desktop\program图片\childwithback.bmp

### 3.1.3 生成COE文件

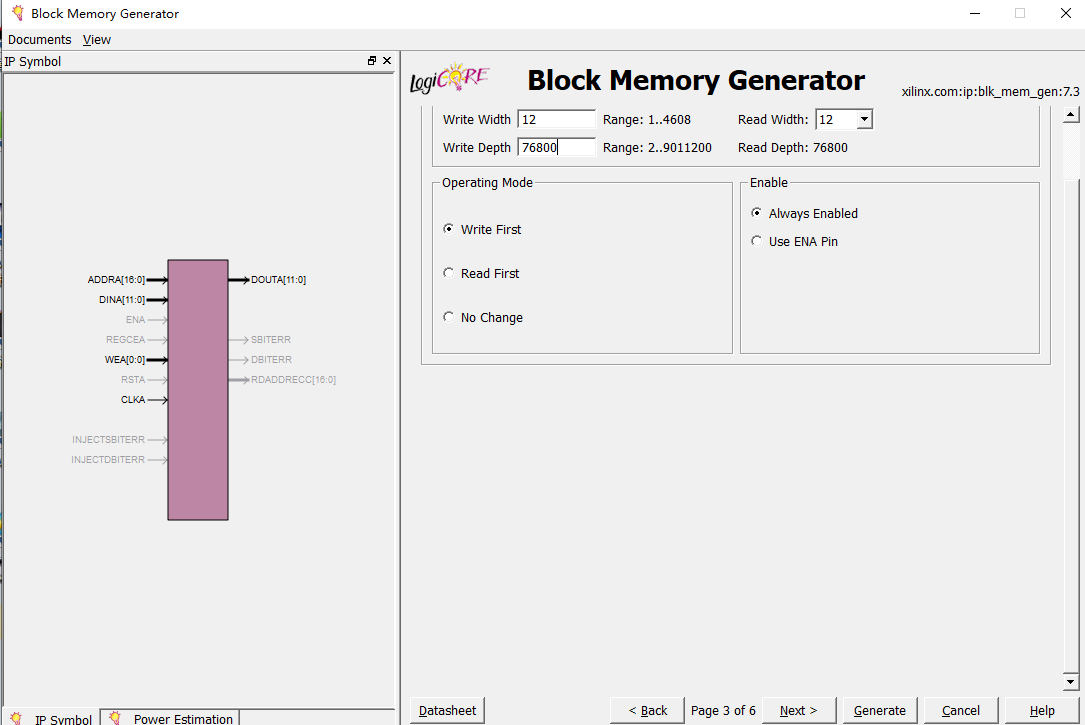
经过查询资料，bmp格式的图片可以通过C语言程序转换为COE文件。其详细代码将在附录中给出。

### 3.1.4 生成ip核

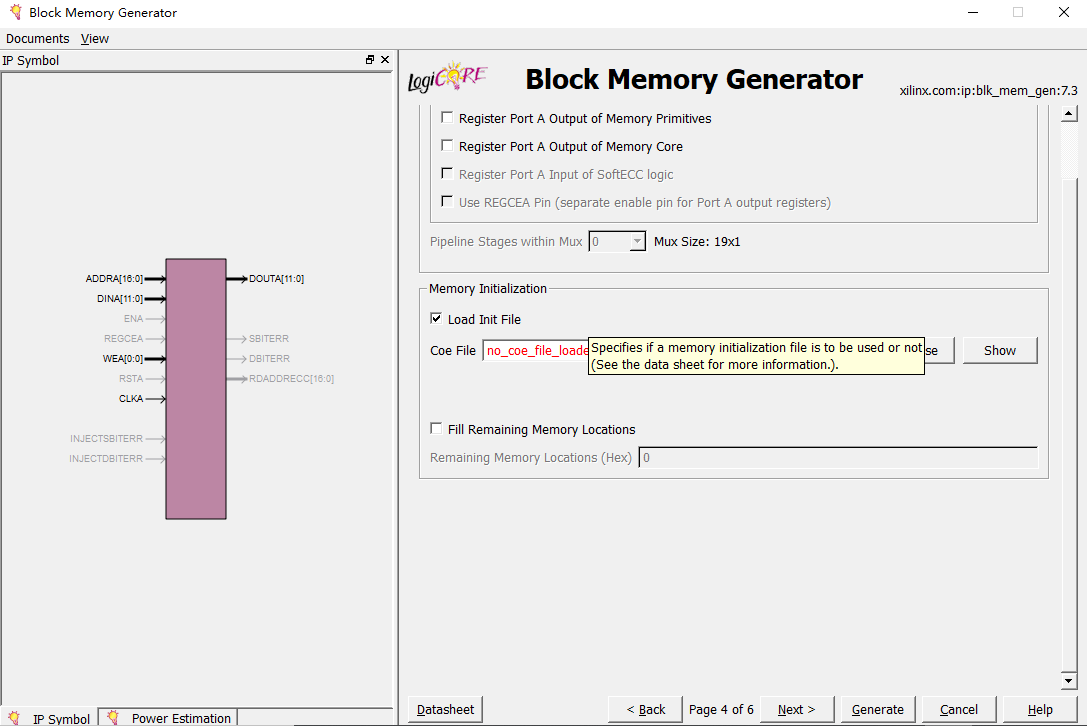
ip核在ise平台中生成。在RAMs&ROMs中选择需要的格式。由于程序input当中有时钟，所以选择Block Memory Generator（可以同步访问）。

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，vga的显示屏大小是 640\*480，而我们的封面图片大小为320\*240，所以我们填入的参数是,76800（320\*240）。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。



在page4选择想要显示的图片的coe文件。



## 3.2 VGA显示

reg [18:0] bg1;

background m0 (.addra(bg1),.douta(spob),.clka(clkdiv[1]));

always@(\*) begin

bg1 <= (col\_addr >= 0 && col\_addr <= 639 && row\_addr >= 0 && row\_addr <= 479)? (479-row\_addr)\*640+col\_addr:0;

## 3.3 PS2键盘输入

module ps2\_ver2(

input clk,

input rst,

input ps2\_clk,

input ps2\_data,

output [9:0] data\_out,

output ready

);

reg ps2\_clk\_falg0, ps2\_clk\_falg1, ps2\_clk\_falg2;

wire negedge\_ps2\_clk;

always@(posedge clk or posedge rst)begin

if(rst)begin

ps2\_clk\_falg0<=1'b0;

ps2\_clk\_falg1<=1'b0;

ps2\_clk\_falg2<=1'b0;

end

else begin

ps2\_clk\_falg0<=ps2\_clk;

ps2\_clk\_falg1<=ps2\_clk\_falg0;

ps2\_clk\_falg2<=ps2\_clk\_falg1;

end

end

assign negedge\_ps2\_clk=!ps2\_clk\_falg1&ps2\_clk\_falg2;

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

case(num)

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 [9:0] data;

reg data\_break, data\_done, data\_expand;

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)begin

data\_expand<=1'b1;

end

else if(temp\_data==8'hF0)begin

data\_break<=1'b1;

end

else begin

data<={data\_expand,data\_break,temp\_data};

data\_done<=1'b1;

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;

endmodule

## 3.4 七段数码管显示

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

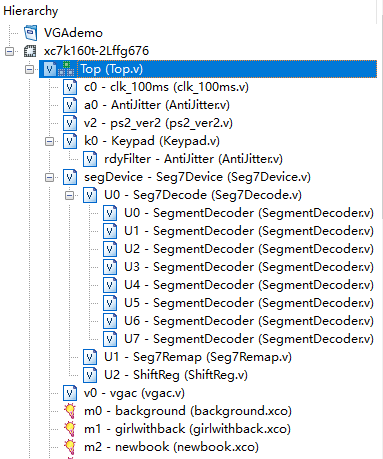
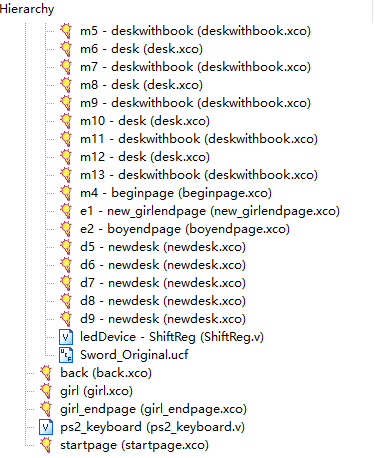
2'b10: begin segment <= ~dispPattern[23:16]; anode <= 4'b1011; end

2'b11: begin segment <= ~dispPattern[31:24]; anode <= 4'b0111; end

endcase

endmodule

## 3.5 Top模块

根据处理按键阵列得到的结果，对人物行走状态、书本位置等进行操作。

if (!wasReady&&keyReady) begin

case (keyCode)

5'hc: begin

walk = ~walk; //walk = ~walk;

b0 = ~b0;

x0 <= x ;

y0 <= y + 9'd50;

end

5'hf:begin

walks = ~walks; //walk = ~walk;

bs0 = ~bs0;

xs0 <= xs ;

ys0 <= ys + 9'd50;

end

5'h13: begin

walk = 1'b1;

b0 = 1'b0;

end

//5'h9: y <= y - 9'd20;

5'h10: begin

walks = 1'b1;

bs0 = 1'b0;

end

default: ;

endcase

当人物在行走时，每隔一段时间根据键盘获得的速度值和人物当前运动方向更改x坐标，若人物走到尽头，则更改方向；若书本到了尽头仍未碰到桌子，则更改书本和人物的状态。

if (walk && clock) begin

if (ahead) x <= x + speed1;

else x <= x - speed1;

end

if (walks && clock) begin

if (aheads) xs <= xs + speed2;

else xs <= xs - speed2;

end

if (!walk && clock) begin

b0 = 1'b1;

y0 <= y0 + 9'd10;

end

if (!walks && clock) begin

bs0 = 1'b1;

ys0 <= ys0 + 9'd10;

end

if (!walks && ys0 > 10'd460) begin

walks = 1'b1;

bs0 = 1'b0;

end

if (!walk && y0 > 10'd460) begin

walk = 1'b1;

b0 = 1'b0;

end

if (ahead && x > 10'd520 ) ahead = !ahead;

if ( !ahead && x < 10'd120) ahead = ~ahead;

if (aheads && xs > 10'd520 ) aheads = !aheads;

if ( !aheads && xs < 10'd120) aheads = ~aheads;

碰撞检测部分，人物停下、书本扔出时，对每一张桌子进行碰撞检测，若书本在期待的范围内，则更改桌子状态、人物得分。

// 碰撞检测

if (!walk) begin

if (!life\_1 && (x0 > desk\_1\_x1 - 30 && x0 < desk\_1\_x1 + 30

&& y0 > desk\_1\_y1 && y0 < desk\_1\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_1 = 2'd1;

score = score + 2;

end

if (!life\_2 && (x0 > desk\_2\_x1 - 30 && x0 < desk\_2\_x1 + 30

&& y0 > desk\_2\_y1 && y0 < desk\_2\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_2 = 2'd1;

score = score + 2;

end

if (!life\_3 && (x0 > desk\_3\_x1 - 30 && x0 < desk\_3\_x1 + 30

&& y0 > desk\_3\_y1 && y0 < desk\_3\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_3 = 2'd1;

score = score + 2;

end

if (!life\_4 && (x0 > desk\_4\_x1 - 30 && x0 < desk\_4\_x1 + 30

&& y0 > desk\_4\_y1 && y0 < desk\_4\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_4 = 2'd1;

score = score + 2;

end

if (!life\_5 && (x0 > desk\_5\_x1 - 30 && x0 < desk\_5\_x1 + 30

&& y0 > desk\_5\_y1 && y0 < desk\_5\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_5 = 2'd1;

score = score + 2;

end

end

if (!walks) begin

if (!life\_1 && (xs0 > desk\_1\_x1 - 30 && xs0 < desk\_1\_x1 + 30

&& ys0 > desk\_1\_y1 && ys0 < desk\_1\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_1 = 2'd2;

scores = scores + 2;

end

if (!life\_2 && (xs0 > desk\_2\_x1 - 30 && xs0 < desk\_2\_x1 + 30

&& ys0 > desk\_2\_y1 && ys0 < desk\_2\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_2 = 2'd2;

scores = scores + 2;

end

if (!life\_3 && (xs0 > desk\_3\_x1 - 30 && xs0 < desk\_3\_x1 + 30

&& ys0 > desk\_3\_y1 && ys0 < desk\_3\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_3 = 2'd2;

scores = scores + 2;

end

if (!life\_4 && (xs0 > desk\_4\_x1 - 30 && xs0 < desk\_4\_x1 + 30

&& ys0 > desk\_4\_y1 && ys0 < desk\_4\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_4 = 2'd2;

scores = scores + 2;

end

if (!life\_5 && (xs0 > desk\_5\_x1 - 30 && xs0 < desk\_5\_x1 + 30

&& ys0 > desk\_5\_y1 && ys0 < desk\_5\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_5 = 2'd2;

scores = scores + 2;

end

end

根据键盘输入，更改人物的速度，即定时更改人物横坐标时的步长。

always @(posedge clkdiv[20]) begin

if (data[7:0]==8'h1c && data[8] == 0 )

begin

speed1 = 1;

end

if (data[7:0]==8'h23 && data[8] == 0 ) begin

speed1 = 2;

end

if (data[7:0]==8'h1d && data[8] == 0 )

begin

speed1 = 3;

end

if (data[7:0]==8'h1b && data[8] == 0 ) begin

speed1 = 4;

end

if (data[7:0]==8'h6b && data[8] == 0 )

begin

speed2 = 1;

end

if (data[7:0]==8'h74 && data[8] == 0 ) begin

speed2 = 2;

end

if (data[7:0]==8'h75 && data[8] == 0 )

begin

speed2 = 3;

end

if (data[7:0]==8'h72 && data[8] == 0 ) begin

speed2 = 4;

end

end

调用IP核。

background m0 (.addra(bg1),.douta(spob),.clka(clkdiv[1]));

girlwithback m1 (.addra(stul),.douta(stu),.clka(clkdiv[1]));

newbook m2 (.addra(bookl),.douta(book),.clka(clkdiv[1]));

childnew s1 (.addra(sonl),.douta(son),.clka(clkdiv[1]));

book s2 (.addra(booksl),.douta(books),.clka(clkdiv[1]));

desk m3 (.addra(deskl\_1),.douta(desk\_1),.clka(clkdiv[1]));

deskwithbook m5 (.addra(deskl\_1),.douta(desk\_books\_1),.clka(clkdiv[1]));

desk m6 (.addra(deskl\_2),.douta(desk\_2),.clka(clkdiv[1]));

deskwithbook m7 (.addra(deskl\_2),.douta(desk\_books\_2),.clka(clkdiv[1]));

desk m8 (.addra(deskl\_3),.douta(desk\_3),.clka(clkdiv[1]));

deskwithbook m9 (.addra(deskl\_3),.douta(desk\_books\_3),.clka(clkdiv[1]));

desk m10 (.addra(deskl\_4),.douta(desk\_4),.clka(clkdiv[1]));

deskwithbook m11 (.addra(deskl\_4),.douta(desk\_books\_4),.clka(clkdiv[1]));

desk m12 (.addra(deskl\_5),.douta(desk\_5),.clka(clkdiv[1]));

deskwithbook m13 (.addra(deskl\_5),.douta(desk\_books\_5),.clka(clkdiv[1]));

beginpage m4 (.addra(startl),.douta(start),.clka(clkdiv[1]));

new\_girlendpage e1 (.addra(finishl),.douta(finish),.clka(clkdiv[1]));

boyendpage e2 (.addra(finishl0),.douta(finish0),.clka(clkdiv[1]));

newdesk d5 (.addra(deskl\_1),.douta(desk\_book\_1),.clka(clkdiv[1]));

newdesk d6 (.addra(deskl\_2),.douta(desk\_book\_2),.clka(clkdiv[1]));

newdesk d7 (.addra(deskl\_3),.douta(desk\_book\_3),.clka(clkdiv[1]));

newdesk d8 (.addra(deskl\_4),.douta(desk\_book\_4),.clka(clkdiv[1]));

newdesk d9 (.addra(deskl\_5),.douta(desk\_book\_5),.clka(clkdiv[1]));

# 四、演示效果

有些开始，进入开始页面。

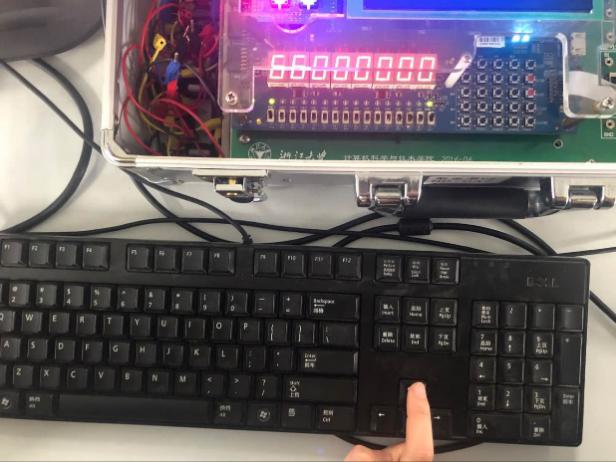


按下开关，进入游戏界面。

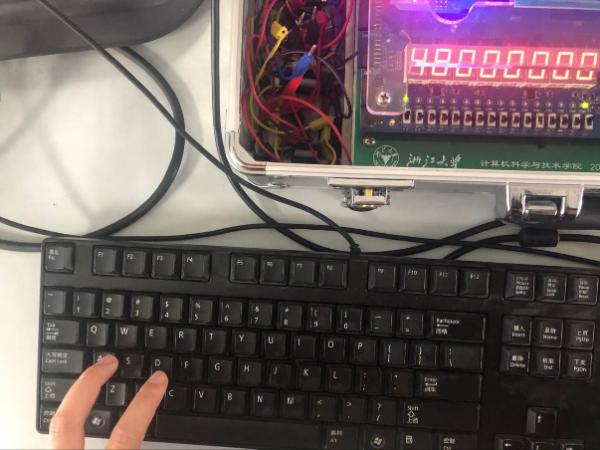


通关PS2键盘可以改变角色的速度，可以通过“↑”、“↓”、“←”、“→”改变男孩角色的速度，通过“W”、“A”、“S”、“D”改变女孩角色的速度。他们的速度会同时显示在七段数码管上。

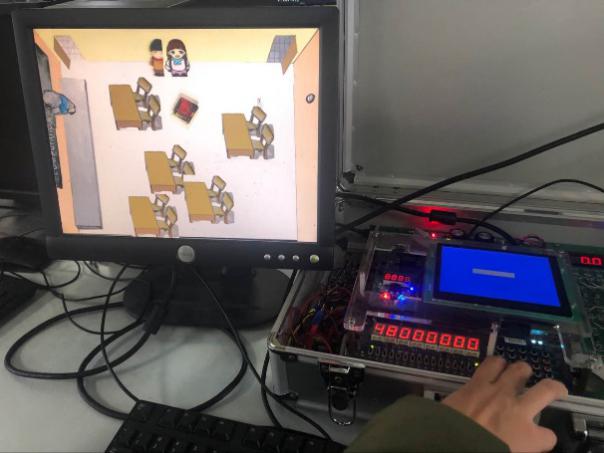
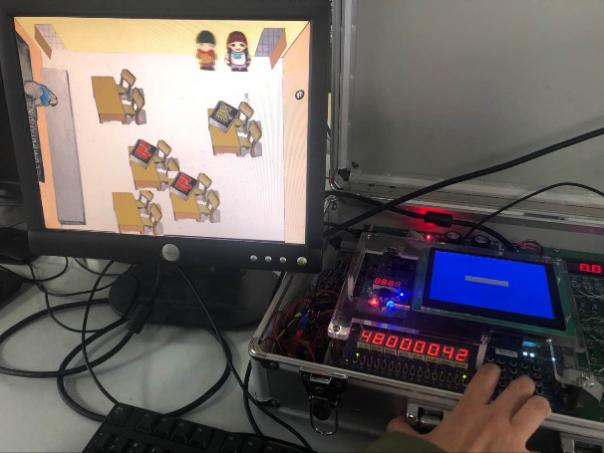
如图所示，通过“↑”、“↓”、“←”、“→”改变男孩角色的速度。

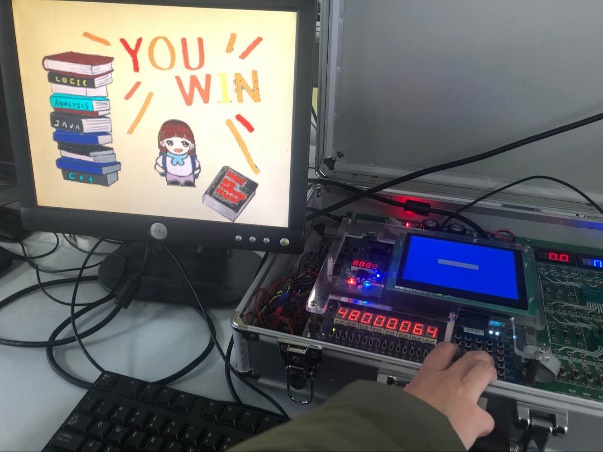
如图所示，通过“W”、“A”、“S”、“D”改变女孩角色的速度。

通过按键，控制角色投掷书本。

当所有座位都被占满后，显示胜利的结束页面。



# 五、问题与改进思路

## 5.1实现过程中的问题

### 5.1.1内存不足的问题

游戏实现过程中，出现了内存不足的问题。

在最初的设计里，开始页面和游戏过程中的教室场景图片都为480\*720，且教室内设计了九张桌子的布局。此时如果反复调用图像ip核时，系统就会报错，提示整体的内存不足。即使将桌子的数量减少到五张，系统依然无法编译通过。

### 5.1.2解决方法

为了最大程度地保留游戏的完整性，我们采取的解决方案是将开始页面、教室场景图片这两张比较占内存的图片的长、宽像素数量缩小到原来的1/2。同时在程序中将原本的每个像素位置与ip核中存储的位置信息一一对应，改为了每组相邻的2\*2的像素与ip核中存储的位置信息对应。这样就可以在不影响游戏完整性的前提下，减小ip核所占用的内存。

## 5.2改进思路

“抢座位大作战”作为一款原创游戏。作者虽然花费了很多精力去构思并实现，但是也知道其中存在一些问题，由于时间和技术的限制，只能在此记述。

1. 可以增加倒计时功能；我们设想可以加入一个倒计时模式，在有限的时间内结束游戏，以最终得分高低计算输赢。这样可以避免双方持续互相打断，丧失自己的投掷机会。

2. 可以同时增加单人和双人模式；当加入倒计时功能后，可以设置游戏为单人模式和双人模式，单人模式中以最终是否达到一定数量计算输赢，双人模式则维持现状。

3. 逻辑可以进一步完善：在实际的游戏过程中，我们发现，当一个角色拥有的座位超过总数的1/2（在当前情景下是三个）时，游戏的胜负已经确定。此时游戏可以直接结束。

# 六、成员分工

# 附录：源代码

bmp文件转Coe文件的C语言代码：

#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("C://Download//picture//new\_girlendpage.coe", "w");

int i;

char s[50] = "C://Download//picture//new\_girlendpage.bmp";

fprintf(outfp, "memory\_initialization\_radix=16;\nmemory\_initialization\_vector=\n");

BMPtoCOE(s, 1);

return 0;

}

Top文件源代码

`timescale 1ns / 1ps

module Top(

input clk,

input rstn,

input [15:0]SW,

input wire ps2\_clk, //ps2 clock

input wire ps2\_data,

output hs,

output vs,

output [3:0] r,

output [3:0] g,

output [3:0] b,

output SEGLED\_CLK,

output SEGLED\_CLR,

output SEGLED\_DO,

output SEGLED\_PEN,

output LED\_CLK,

output LED\_CLR,

output LED\_DO,

output LED\_PEN,

inout [4:0]BTN\_X,

inout [3:0]BTN\_Y,

output buzzer

);

// 输入模块

reg [31:0]clkdiv;

wire [9:0] data;

always@(posedge clk) begin

clkdiv <= clkdiv + 1'b1;

end

wire clock;

clk\_100ms c0 (

.clk(clk),

.clk\_100ms(clock)

);

assign buzzer = 1'b1;

wire [15:0] SW\_OK;

AntiJitter #(4) a0[15:0](.clk(clkdiv[15]), .I(SW), .O(SW\_OK));

wire [4:0] keyCode;

wire keyReady;

ps2\_ver2 v2(.clk(clk),.rst(),.ps2\_clk(ps2\_clk),.ps2\_data(ps2\_data),.data\_out(data),.ready());//use the ps2\_ver2 block

Keypad k0 (.clk(clkdiv[15]), .keyX(BTN\_Y), .keyY(BTN\_X), .keyCode(keyCode), .ready(keyReady));

// 数据初始化模块

parameter desk\_4\_x1 = 160;

parameter desk\_4\_x2 = 280;

parameter desk\_4\_y1 = 100;

parameter desk\_4\_y2 = 220;

parameter desk\_2\_x1 = 230;

parameter desk\_2\_x2 = 350;

parameter desk\_2\_y1 = 250;

parameter desk\_2\_y2 = 370;

parameter desk\_3\_x1 = 180;

parameter desk\_3\_x2 = 300;

parameter desk\_3\_y1 = 360;

parameter desk\_3\_y2 = 480;

parameter desk\_1\_x1 = 320;

parameter desk\_1\_x2 = 440;

parameter desk\_1\_y1 = 320;

parameter desk\_1\_y2 = 440;

parameter desk\_5\_x1 = 420;

parameter desk\_5\_x2 = 540;

parameter desk\_5\_y1 = 160;

parameter desk\_5\_y2 = 280;

reg [1:0] life\_1 = 2'b00;

reg [1:0] life\_2 = 2'b00;

reg [1:0] life\_3 = 2'b00;

reg [1:0] life\_4 = 2'b00;

reg [1:0] life\_5 = 2'b00;

// 七段数码管显示模块

wire [31:0] segTestData;

wire [3:0]sout;

reg [3:0] speed2 = 4'd2;

reg [3:0] speed1 = 4'd2;

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];

// 核心控制模块

// stu

reg [9:0] x;

reg [8:0] y;

reg [9:0] xs;

reg [8:0] ys;

//book

reg [9:0] x0;

reg [8:0] y0;

reg b0 = 1'b0;

reg [9:0] xs0;

reg [8:0] ys0;

reg bs0 = 1'b0;

// status

reg walk ;

reg ahead ;

reg walks ;

reg aheads ;

reg [11:0] vga\_data;

wire [9:0] col\_addr;

wire [8:0] row\_addr;

wire [19:0] x\_sqr, y\_sqr, r\_sqr;

vgac v0 (

.vga\_clk(clkdiv[1]), .clrn(SW\_OK[0]), .d\_in(vga\_data), .row\_addr(row\_addr), .col\_addr(col\_addr), .r(r), .g(g), .b(b), .hs(hs), .vs(vs)

);

reg wasReady;

reg [9:0] radius = 10'd15;

always @(posedge clk) begin

if (!rstn) begin

x <= 10'd120;

y <= 9'd20;

radius <= 10'd15;

walk = 1'b1;

ahead = 1'b1;

score = 4'd0;

xs <= 10'd480;

ys <= 9'd20;

walks = 1'b1;

aheads = 1'b0;

scores = 4'd0;

end else begin

wasReady <= keyReady;

if (!wasReady&&keyReady) begin

case (keyCode)

5'hc: begin

walk = ~walk; //walk = ~walk;

b0 = ~b0;

x0 <= x ;

y0 <= y + 9'd50;

end

5'hf:begin

walks = ~walks; //walk = ~walk;

bs0 = ~bs0;

xs0 <= xs ;

ys0 <= ys + 9'd50;

end

5'h13: begin

walk = 1'b1;

b0 = 1'b0;

end

//5'h9: y <= y - 9'd20;

5'h10: begin

walks = 1'b1;

bs0 = 1'b0;

end

default: ;

endcase

// lc-S 23-W ld-D lb-E 6b下 74上 75右 72左

if (clock && data[7:0]==8'h1c && data[8] == 0) begin

walk = ~walk; //walk = ~walk;

b0 = ~b0;

x0 <= x ;

y0 <= y + 9'd30;

score = score + 3;

end

end

if (walk && clock) begin

if (ahead) x <= x + speed1;

else x <= x - speed1;

end

if (walks && clock) begin

if (aheads) xs <= xs + speed2;

else xs <= xs - speed2;

end

// 碰撞检测

if (!walk) begin

if (!life\_1 && (x0 > desk\_1\_x1 - 30 && x0 < desk\_1\_x1 + 30

&& y0 > desk\_1\_y1 && y0 < desk\_1\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_1 = 2'd1;

score = score + 2;

end

if (!life\_2 && (x0 > desk\_2\_x1 - 30 && x0 < desk\_2\_x1 + 30

&& y0 > desk\_2\_y1 && y0 < desk\_2\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_2 = 2'd1;

score = score + 2;

end

if (!life\_3 && (x0 > desk\_3\_x1 - 30 && x0 < desk\_3\_x1 + 30

&& y0 > desk\_3\_y1 && y0 < desk\_3\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_3 = 2'd1;

score = score + 2;

end

if (!life\_4 && (x0 > desk\_4\_x1 - 30 && x0 < desk\_4\_x1 + 30

&& y0 > desk\_4\_y1 && y0 < desk\_4\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_4 = 2'd1;

score = score + 2;

end

if (!life\_5 && (x0 > desk\_5\_x1 - 30 && x0 < desk\_5\_x1 + 30

&& y0 > desk\_5\_y1 && y0 < desk\_5\_y2) )

begin

walk = 1'b1;

b0 = 1'b0;

life\_5 = 2'd1;

score = score + 2;

end

end

if (!walks) begin

if (!life\_1 && (xs0 > desk\_1\_x1 - 30 && xs0 < desk\_1\_x1 + 30

&& ys0 > desk\_1\_y1 && ys0 < desk\_1\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_1 = 2'd2;

scores = scores + 2;

end

if (!life\_2 && (xs0 > desk\_2\_x1 - 30 && xs0 < desk\_2\_x1 + 30

&& ys0 > desk\_2\_y1 && ys0 < desk\_2\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_2 = 2'd2;

scores = scores + 2;

end

if (!life\_3 && (xs0 > desk\_3\_x1 - 30 && xs0 < desk\_3\_x1 + 30

&& ys0 > desk\_3\_y1 && ys0 < desk\_3\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_3 = 2'd2;

scores = scores + 2;

end

if (!life\_4 && (xs0 > desk\_4\_x1 - 30 && xs0 < desk\_4\_x1 + 30

&& ys0 > desk\_4\_y1 && ys0 < desk\_4\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_4 = 2'd2;

scores = scores + 2;

end

if (!life\_5 && (xs0 > desk\_5\_x1 - 30 && xs0 < desk\_5\_x1 + 30

&& ys0 > desk\_5\_y1 && ys0 < desk\_5\_y2) )

begin

walks = 1'b1;

bs0 = 1'b0;

life\_5 = 2'd2;

scores = scores + 2;

end

end

if (!walk && clock) begin

b0 = 1'b1;

y0 <= y0 + 9'd10;

end

if (!walks && clock) begin

bs0 = 1'b1;

ys0 <= ys0 + 9'd10;

end

if (!walks && ys0 > 10'd460) begin

walks = 1'b1;

bs0 = 1'b0;

end

if (!walk && y0 > 10'd460) begin

walk = 1'b1;

b0 = 1'b0;

end

if (ahead && x > 10'd520 ) ahead = !ahead;

if ( !ahead && x < 10'd120) ahead = ~ahead;

if (aheads && xs > 10'd520 ) aheads = !aheads;

if ( !aheads && xs < 10'd120) aheads = ~aheads;

end

end

always @(posedge clkdiv[20]) begin

if (data[7:0]==8'h1c && data[8] == 0 )

begin

speed1 = 1;

end

if (data[7:0]==8'h23 && data[8] == 0 ) begin

speed1 = 2;

end

if (data[7:0]==8'h1d && data[8] == 0 )

begin

speed1 = 3;

end

if (data[7:0]==8'h1b && data[8] == 0 ) begin

speed1 = 4;

end

if (data[7:0]==8'h6b && data[8] == 0 )

begin

speed2 = 1;

end

if (data[7:0]==8'h74 && data[8] == 0 ) begin

speed2 = 2;

end

if (data[7:0]==8'h75 && data[8] == 0 )

begin

speed2 = 3;

end

if (data[7:0]==8'h72 && data[8] == 0 ) begin

speed2 = 4;

end

end

assign x\_sqr = (x - col\_addr) \* (x - col\_addr);

assign y\_sqr = (y - row\_addr) \* (y - row\_addr);

assign r\_sqr = radius \* radius;

reg [3:0] score;

reg [3:0] scores;

wire [11:0] spob;

reg [18:0] bg1;

wire [11:0] stu;

reg [18:0] stul;

wire [11:0] son;

reg [18:0] sonl;

wire [11:0] books;

reg [18:0] booksl;

wire [11:0] book;

reg [18:0] bookl;

wire [11:0] desk\_1;

wire [11:0] desk\_book\_1;

reg [18:0] deskl\_1;

wire [11:0] desk\_2;

wire [11:0] desk\_book\_2;

reg [18:0] deskl\_2;

wire [11:0] desk\_3;

wire [11:0] desk\_book\_3;

reg [18:0] deskl\_3;

wire [11:0] desk\_4;

wire [11:0] desk\_book\_4;

reg [18:0] deskl\_4;

wire [11:0] desk\_5;

wire [11:0] desk\_book\_5;

wire [11:0] desk\_books\_1;

wire [11:0] desk\_books\_2;

wire [11:0] desk\_books\_3;

wire [11:0] desk\_books\_4;

wire [11:0] desk\_books\_5;

reg [18:0] deskl\_5;

wire [11:0] start;

reg [18:0] startl;

wire [11:0] finish;

reg [18:0] finishl;

wire [11:0] finish0;

reg [18:0] finishl0;

background m0 (.addra(bg1),.douta(spob),.clka(clkdiv[1]));

girlwithback m1 (.addra(stul),.douta(stu),.clka(clkdiv[1]));

newbook m2 (.addra(bookl),.douta(book),.clka(clkdiv[1])); //待修改，新书

childnew s1 (.addra(sonl),.douta(son),.clka(clkdiv[1]));

book s2 (.addra(booksl),.douta(books),.clka(clkdiv[1]));

desk m3 (.addra(deskl\_1),.douta(desk\_1),.clka(clkdiv[1]));

deskwithbook m5 (.addra(deskl\_1),.douta(desk\_books\_1),.clka(clkdiv[1]));

desk m6 (.addra(deskl\_2),.douta(desk\_2),.clka(clkdiv[1]));

deskwithbook m7 (.addra(deskl\_2),.douta(desk\_books\_2),.clka(clkdiv[1]));

desk m8 (.addra(deskl\_3),.douta(desk\_3),.clka(clkdiv[1]));

deskwithbook m9 (.addra(deskl\_3),.douta(desk\_books\_3),.clka(clkdiv[1]));

desk m10 (.addra(deskl\_4),.douta(desk\_4),.clka(clkdiv[1]));

deskwithbook m11 (.addra(deskl\_4),.douta(desk\_books\_4),.clka(clkdiv[1]));

desk m12 (.addra(deskl\_5),.douta(desk\_5),.clka(clkdiv[1]));

deskwithbook m13 (.addra(deskl\_5),.douta(desk\_books\_5),.clka(clkdiv[1]));

beginpage m4 (.addra(startl),.douta(start),.clka(clkdiv[1]));

new\_girlendpage e1 (.addra(finishl),.douta(finish),.clka(clkdiv[1]));

boyendpage e2 (.addra(finishl0),.douta(finish0),.clka(clkdiv[1]));

newdesk d5 (.addra(deskl\_1),.douta(desk\_book\_1),.clka(clkdiv[1]));

newdesk d6 (.addra(deskl\_2),.douta(desk\_book\_2),.clka(clkdiv[1])); // 待修改，新书+桌子

newdesk d7 (.addra(deskl\_3),.douta(desk\_book\_3),.clka(clkdiv[1]));

newdesk d8 (.addra(deskl\_4),.douta(desk\_book\_4),.clka(clkdiv[1]));

newdesk d9 (.addra(deskl\_5),.douta(desk\_book\_5),.clka(clkdiv[1]));

always@(\*) begin

bg1 <= (col\_addr >= 0 && col\_addr <= 639 && row\_addr >= 0 && row\_addr <= 479)? (479-row\_addr)\*640+col\_addr:0;

startl <= (col\_addr >= 0 && col\_addr <= 639 && row\_addr >= 0 && row\_addr <= 479)? (479-row\_addr)/2\*320+col\_addr/2:0;

finishl <= (col\_addr >= 0 && col\_addr <= 639 && row\_addr >= 0 && row\_addr <= 479)? (479-row\_addr)/2\*320+col\_addr/2:0;

finishl0 <= (col\_addr >= 0 && col\_addr <= 639 && row\_addr >= 0 && row\_addr <= 479)? (479-row\_addr)/2\*320+col\_addr/2:0;

sonl <= (col\_addr >= xs && col\_addr <= xs+59 && row\_addr >= ys && row\_addr <= ys+79)? (79-(row\_addr-ys))\*60+col\_addr-xs:0;

booksl <= (col\_addr >= xs0 && col\_addr <= xs0+79 && row\_addr >= ys0 && row\_addr <= ys0+79)? (79-(row\_addr-ys0))\*80+col\_addr-xs0:0;

stul <= (col\_addr >= x && col\_addr <= x+59 && row\_addr >= y && row\_addr <= y+79)? (79-(row\_addr-y))\*60+col\_addr-x:0;

bookl <= (col\_addr >= x0 && col\_addr <= x0+79 && row\_addr >= y0 && row\_addr <= y0+79)? (79-(row\_addr-y0))\*80+col\_addr-x0:0;

deskl\_1 <= (col\_addr >= desk\_1\_x1 && col\_addr < desk\_1\_x2 && row\_addr >= desk\_1\_y1 && row\_addr < desk\_1\_y2)? (119-(row\_addr-desk\_1\_y1))\*120+col\_addr-desk\_1\_x1:0;

deskl\_2 <= (col\_addr >= desk\_2\_x1 && col\_addr < desk\_2\_x2 && row\_addr >= desk\_2\_y1 && row\_addr < desk\_2\_y2)? (119-(row\_addr-desk\_2\_y1))\*120+col\_addr-desk\_2\_x1:0;

deskl\_3 <= (col\_addr >= desk\_3\_x1 && col\_addr < desk\_3\_x2 && row\_addr >= desk\_3\_y1 && row\_addr < desk\_3\_y2)? (119-(row\_addr-desk\_3\_y1))\*120+col\_addr-desk\_3\_x1:0;

deskl\_4 <= (col\_addr >= desk\_4\_x1 && col\_addr < desk\_4\_x2 && row\_addr >= desk\_4\_y1 && row\_addr < desk\_4\_y2)? (119-(row\_addr-desk\_4\_y1))\*120+col\_addr-desk\_4\_x1:0;

deskl\_5 <= (col\_addr >= desk\_5\_x1 && col\_addr < desk\_5\_x2 && row\_addr >= desk\_5\_y1 && row\_addr < desk\_5\_y2)? (119-(row\_addr-desk\_5\_y1))\*120+col\_addr-desk\_5\_x1:0;

if (SW[1]) // 开始界面

vga\_data <= start[11:0];

else if (SW[3])

vga\_data <= finish[11:0];

else if (SW[4])

vga\_data <= finish0[11:0];

else if (score + scores == 4'd10 && score > scores)

vga\_data <= finish[11:0];

else if (score + scores == 4'd10 && score < scores)

vga\_data <= finish0[11:0];

else begin //游戏界面

if ((col\_addr >= x && col\_addr <= x+59 && row\_addr >= y && row\_addr <= y+79))

vga\_data <= stu[11:0]; //学生

else if ((col\_addr >= xs && col\_addr <= xs+59 && row\_addr >= ys && row\_addr <= ys+79))

vga\_data <= son[11:0]; //学生

else if (!walks && (col\_addr >= xs0 && col\_addr <= xs0+79 && row\_addr >= ys0 && row\_addr <= ys0+79))

vga\_data <= books[11:0];

else if (!walk && (col\_addr >= x0 && col\_addr <= x0+79 && row\_addr >= y0 && row\_addr <= y0+79))

vga\_data <= book[11:0];

else if ( col\_addr > desk\_1\_x1 && col\_addr < desk\_1\_x2

&& row\_addr > desk\_1\_y1 && row\_addr < desk\_1\_y2 ) //桌子1

begin

if (!life\_1) vga\_data <= desk\_1[11:0];

else if (life\_1 == 1) vga\_data <= desk\_book\_1[11:0];

else if (life\_1 == 2) vga\_data <= desk\_books\_1[11:0];

end

else if ( col\_addr > desk\_2\_x1 && col\_addr < desk\_2\_x2

&& row\_addr > desk\_2\_y1 && row\_addr < desk\_2\_y2 ) //桌子2

begin

if (!life\_2) vga\_data <= desk\_2[11:0];

else if (life\_2 == 1) vga\_data <= desk\_book\_2[11:0];

else if (life\_2 == 2) vga\_data <= desk\_books\_2[11:0];

end

else if ( col\_addr > desk\_3\_x1 && col\_addr < desk\_3\_x2

&& row\_addr > desk\_3\_y1 && row\_addr < desk\_3\_y2 ) //桌子3

begin

if (!life\_3) vga\_data <= desk\_3[11:0];

else if (life\_3 == 1) vga\_data <= desk\_book\_3[11:0];

else if (life\_3 == 2) vga\_data <= desk\_books\_3[11:0];

end

else if ( col\_addr > desk\_4\_x1 && col\_addr < desk\_4\_x2

&& row\_addr > desk\_4\_y1 && row\_addr < desk\_4\_y2 ) //桌子4

begin

if (!life\_4) vga\_data <= desk\_4[11:0];

else if (life\_4 == 1) vga\_data <= desk\_book\_4[11:0];

else if (life\_4 == 2) vga\_data <= desk\_books\_4[11:0];

end

else if ( col\_addr > desk\_5\_x1 && col\_addr < desk\_5\_x2

&& row\_addr > desk\_5\_y1 && row\_addr < desk\_5\_y2 ) //桌子5

begin

if (!life\_5) vga\_data <= desk\_5[11:0];

else if (life\_5 == 1) vga\_data <= desk\_book\_5[11:0];

else if (life\_5 == 2) vga\_data <= desk\_books\_5[11:0];

end

else

vga\_data <= spob[11:0];

end

end

assign segTestData = {speed1,speed2,8'b0,9'b0,score,scores};

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