数字系统

综合项目— VGA贪吃蛇

小组成员：

姓名：刘智超 学号：U201713314 贡献比例：50%

姓名：李佳勋 学号：U201713304 贡献比例：50%

具体分工：

李佳勋负责eating apple,vga,seg\_dispaly

刘智超负责top,Snake,GameCtrl, Key以及PPT和报告

专业：电子信息与通信学院

班级：电信1703班

指导教师：罗杰

一．项目描述

1.这个游戏的简要介绍如下：

玩家可以通过NEXYS4DDR开发板上的上下左右4个按键来控制蛇的方向，寻找吃的东西，在游戏界面中会随机生成苹果，蛇每吃一个苹果就能得到一个积分，而且蛇的身子会越吃越长，身子越长玩的难度就越大，不能碰墙，不能咬到自己的身体，更不能咬自己的尾巴，以上任意一种情况一旦出现，游戏结束，自动刷新开始下一场游戏，并且可以通过开发板上5个按键的最中间按键来暂停游戏。

2.具体实现时的几个技术点如下：

* 随机数产生
  + 蛋出现的位置（X,Y)
* 蛇身体的移动显示
  + 调整显存数据
  + 直线显示，行、列组合
* 蛇碰墙，咬尾判断

3.功能描述：

我们做的VGA贪吃蛇除了上述所说的基本功能以外，主要有以下几个亮点：

（1）.速度控制：通过三个开关的和switch0+switch1+switch2来控制速度等级，通过计数循环来根据不同等级设置速度.

（2）.死亡闪烁：设置显示标志，当标志赋值为1时，显示蛇身，赋值为0时，蛇身不显示，通过0，1交替，时钟延迟达到闪烁的效果

（3）.苹果显示：整个苹果是由一个像素矩阵构成，不同像素点根据坐标设置成不同颜色，以此勾勒出苹果形状

（4）.分数记录：每吃到一个苹果，计数变量加1，通过动态扫描显示到数码管上，而死亡之后计数会清零，重新开始计数

4.主要设计思路：

（1）蛇身通过数组来存储，初始化为3块，最长为16块

（2）蛇身更新：每次更新时前一块赋值为后一块，用来完成蛇身的运动

（3）速度控制：通过3个开关来控制4个速度等级

二．概要设计

这次实验我们主要是用verilog语言来编写的，主要由一个顶层top模块和余下6个子模块构成。

1. Top:顶层模块，用来形成整个项目的架构

2. Game\_Ctrl\_Unit: 游戏控制模块，根据游戏状态产生相应控制信号

3. Snake\_Eatting\_Apple: 苹果和炸弹的产生控制模块

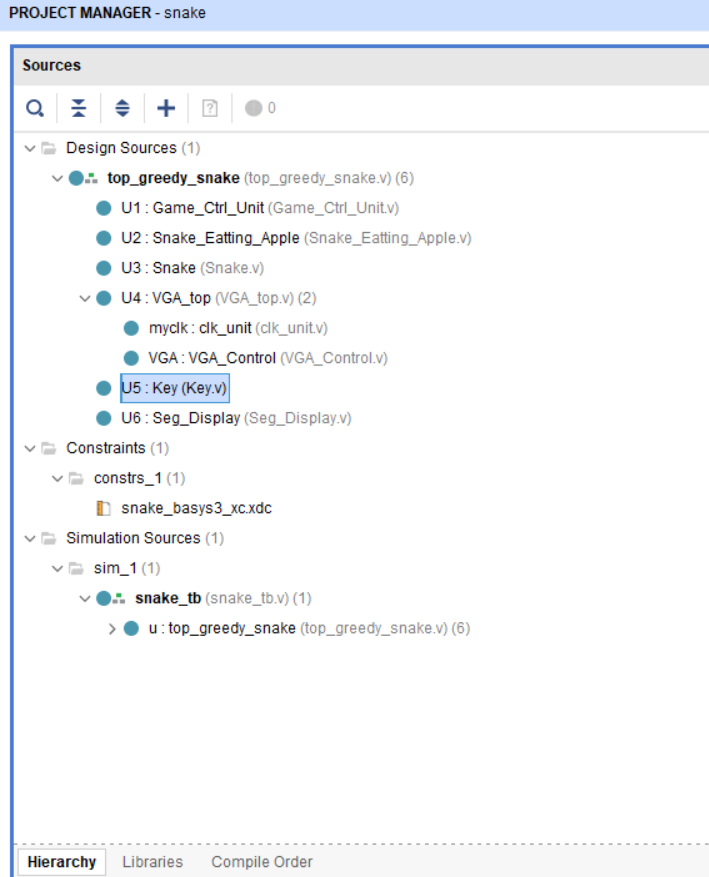
4. Snake：蛇运动情况控制模块.

5. VGA\_top：VGA显示模块

6. Key：按键检测模块 延时消抖.

7. Seg\_Display：数码管计分模块

具体结构如下图“



三．详细设计

1.各个模块代码如下:

(1)顶层模块：

这个模块主要是对于整个项目要用到的各个子模块和输入输出变量进行声明以及相应的注解，通过这个模块可以了解整个工程的架构。

module top\_greedy\_snake

(

input clk,

input rst,

input left,

input right,

input up,

input down,

input switch0,

input switch1,

input switch2,

output hsync,

output vsync,

output [11:0]color\_out,

output [7:0]seg\_out,

output [3:0]sel

);

wire left\_key\_press;

wire right\_key\_press;

wire up\_key\_press;

wire down\_key\_press;

wire [1:0]snake;

wire [9:0]x\_pos;

wire [9:0]y\_pos;

wire [5:0]apple\_x;

wire [4:0]apple\_y;

wire [5:0]head\_x;

wire [5:0]head\_y;

wire [5:0]boom\_x;

wire [4:0]boom\_y;

wire add\_cube;

wire[1:0]game\_status;//游戏状态

wire hit\_wall;

wire hit\_body;

wire eat\_boom;

wire die\_flash;

wire restart;

wire [6:0]cube\_num;

wire rst\_n;

assign rst\_n = ~rst;

Game\_Ctrl\_Unit U1 (

.clk(clk),

.rst(rst\_n),

.key1\_press(left\_key\_press),

.key2\_press(right\_key\_press),

.key3\_press(up\_key\_press),

.key4\_press(down\_key\_press),

.game\_status(game\_status),

.hit\_wall(hit\_wall),

.hit\_body(hit\_body),

.eat\_boom(eat\_boom),

.die\_flash(die\_flash),

.restart(restart)

);

Snake\_Eatting\_Apple U2 (

.clk(clk),

.rst(rst\_n),

.apple\_x(apple\_x),

.apple\_y(apple\_y),

.head\_x(head\_x),

.head\_y(head\_y),

.boom\_x(boom\_x),

.boom\_y(boom\_y),

.add\_cube(add\_cube) ,

.eat\_boom(eat\_boom),

.game\_status(game\_status)

);

Snake U3 (

.clk(clk),

.rst(rst\_n),

.left\_press(left\_key\_press),

.right\_press(right\_key\_press),

.up\_press(up\_key\_press),

.down\_press(down\_key\_press),

.snake(snake),

.x\_pos(x\_pos),

.y\_pos(y\_pos),

.head\_x(head\_x),

.head\_y(head\_y),

.add\_cube(add\_cube),

.game\_status(game\_status),

.cube\_num(cube\_num),

.hit\_body(hit\_body),

.hit\_wall(hit\_wall),

//.eat\_boom(eat\_boom),

.die\_flash(die\_flash),

.switch0(switch0),

.switch1(switch1),

.switch2(switch2)

);

VGA\_top U4 (

.clk(clk),

.rst(rst),

.hsync(hsync),

.vsync(vsync),

.snake(snake),

.color\_out(color\_out),

.x\_pos(x\_pos),

.y\_pos(y\_pos),

.apple\_x(apple\_x),

.apple\_y(apple\_y),

.boom\_x(boom\_x),

.boom\_y(boom\_y)

);

Key U5 (

.clk(clk),

.rst(rst\_n),

.left(left),

.right(right),

.up(up),

.down(down),

.left\_key\_press(left\_key\_press),

.right\_key\_press(right\_key\_press),

.up\_key\_press(up\_key\_press),

.down\_key\_press(down\_key\_press)

);

Seg\_Display U6 (

.clk(clk),

.rst(rst\_n),

.add\_cube(add\_cube),

.game\_status(game\_status),

.seg\_out(seg\_out),

.sel(sel)

);

endmodule

(2) Game\_Ctrl\_Unit:游戏控制模块

这个模块主要是描述如何通过Nexys4 DDR开发板来控制游戏中蛇的移动，即如何通过输入按键来输出相应的运动和游戏状态。

module Game\_Ctrl\_Unit

(

input clk,

input rst,

input key1\_press,

input key2\_press,

input key3\_press,

input key4\_press,

output reg [1:0]game\_status,//游戏状态

input hit\_wall,

input hit\_body,

input eat\_boom,

output reg die\_flash,

output reg restart

);

localparam RESTART = 2'b00;

localparam START = 2'b01;

localparam PLAY = 2'b10;

localparam DIE = 2'b11;

reg[31:0]clk\_cnt;

always@(posedge clk or negedge rst)

begin

if(!rst) begin

game\_status <= START;

clk\_cnt <= 0;

die\_flash <= 1;

restart <= 0;

end

else begin

case(game\_status)

RESTART:begin //游戏开始等待

if(clk\_cnt <= 5) begin

clk\_cnt <= clk\_cnt + 1;

restart <= 1;

end

else begin

game\_status <= START;

clk\_cnt <= 0;

restart <= 0;

end

end

START:begin

if (key1\_press | key2\_press | key3\_press | key4\_press)

game\_status <= PLAY;

else

game\_status <= START;

end

PLAY:begin

if(hit\_wall | hit\_body | eat\_boom)

game\_status <= DIE;

else

game\_status <= PLAY;

end

DIE:begin

if(clk\_cnt <= 200\_000\_000) begin

clk\_cnt <= clk\_cnt + 1'b1;

if(clk\_cnt == 25\_000\_000)

die\_flash <= 0;

else if(clk\_cnt == 50\_000\_000)

die\_flash <= 1'b1;

else if(clk\_cnt == 75\_000\_000)

die\_flash <= 1'b0;

else if(clk\_cnt == 100\_000\_000)

die\_flash <= 1'b1;

else if(clk\_cnt == 125\_000\_000)

die\_flash <= 1'b0;

else if(clk\_cnt == 150\_000\_000)

die\_flash <= 1'b1;

end //死亡闪烁

else

begin

die\_flash <= 1;

clk\_cnt <= 0;

game\_status <= RESTART;

end

end

endcase

end

end

endmodule

(3) Snake\_Eatting\_Apple: 苹果产生控制模块

这个模块主要是描述苹果如何通过伪随机数产生的，如果伪随机数生成的坐标超出设置的屏幕范围，则重新生成坐标，直到所生成坐标在屏幕范围以内才会正常产生苹果。

module Snake\_Eatting\_Apple

(

input clk,

input rst,

input [5:0]head\_x,

input [5:0]head\_y,

output reg [5:0]apple\_x,

output reg [4:0]apple\_y,

output reg [5:0]boom\_x,

output reg [4:0]boom\_y,

output reg add\_cube,

output reg eat\_boom,

input [1:0] game\_status

);

reg [31:0]clk\_cnt;

reg [10:0]random\_num;

reg [10:0]ano\_random;

reg [31:0]a\_clk\_cnt;

localparam RESTART = 2'b00;

always@(posedge clk)

begin

random\_num = random\_num + 999; //用加法产生随机数

//随机数高6位为苹果X坐标 低5位为苹果Y坐标

ano\_random[10:5] <= 1000 ;

ano\_random[4:0]<= 1000;

//该随机数高6位为炸弹X坐标 低5位为炸弹Y坐标

end

//苹果生成

always@(posedge clk or negedge rst) begin

if(!rst) begin

clk\_cnt <= 0;

apple\_x <= 24;

apple\_y <= 10;

add\_cube <= 0;

end

else begin

clk\_cnt <= clk\_cnt+1;

if(clk\_cnt == 250\_000) begin

clk\_cnt <= 0;

if(apple\_x == head\_x && apple\_y == head\_y) begin

add\_cube <= 1;

apple\_x <= (random\_num[10:5] > 38) ? (random\_num[10:5] - 25) : (random\_num[10:5] == 0) ? 1 : random\_num[10:5];

apple\_y <= (random\_num[4:0] > 28) ? (random\_num[4:0] - 3) : (random\_num[4:0] == 0) ? 1:random\_num[4:0];

end //判断随机数是否超出频幕坐标范围 将随机数转换为下个苹果的X Y坐标

else

add\_cube <= 0;

end

end

end

//炸弹生成

always@(posedge clk or negedge rst) begin

if(!rst) begin

clk\_cnt <= 0;

boom\_x <=80;

boom\_y <=80;

end

else begin

clk\_cnt <= clk\_cnt+1;

if(clk\_cnt == 250\_000) begin

clk\_cnt <= 0;

if(apple\_x == head\_x && apple\_y == head\_y) begin

boom\_x <= (ano\_random[10:5]< 1000) ? (ano\_random[10:5] <=200) : (ano\_random[10:5] <=200) ;

boom\_y <= (ano\_random[4:0] <1000) ? (ano\_random[4:0] <=200) : (ano\_random[4:0] <= 200) ;

end //判断随机数是否超出频幕坐标范围 将随机数转换为下个苹果的X Y坐标

end

end

end

always@(posedge clk or negedge rst) begin

if(!rst) begin

a\_clk\_cnt <= 0;

eat\_boom <= 0;

end

else if (game\_status == RESTART)

begin

a\_clk\_cnt <= 0;

eat\_boom <= 0;

end

else begin

a\_clk\_cnt <= a\_clk\_cnt+1;

if(a\_clk\_cnt == 250\_000) begin

a\_clk\_cnt <= 0;

if(boom\_x == head\_x && boom\_y == head\_y) begin

eat\_boom <= 1;

end

end

end

end

endmodule

(4) Snake: 蛇运动情况控制模块

这个模块是很重要的一个模块，包含了如何进行速度控制以及蛇身运动的机制，可以说是整个工程的核心模块，同时也把游戏屏幕范围内的区域分成几个部分，通过2位二进制数来分为4种，头，身体，墙，无，在之后VGA显示时有比较大的帮助。

module Snake

(

input clk,

input rst,

input left\_press,

input right\_press,

input up\_press,

input down\_press,

input [5:0]boom\_x,

input [4:0]boom\_y,

output reg [1:0]snake,//用于表示当前扫描扫描的部件 四种状态 00：无 01：头 10：身体 11：墙

input [9:0]x\_pos,

input [9:0]y\_pos,//扫描坐标 单位："像素点"

output [5:0]head\_x,

output [5:0]head\_y,//头部格坐标

input add\_cube,//增加体长信号

input [1:0]game\_status,//四种游戏状态

output reg [6:0]cube\_num,

output reg hit\_body, //撞到身子信号

output reg hit\_wall, //撞到墙信号

// output reg eat\_boom,

input die\_flash, //闪动信号

input switch0,

input switch1,

input switch2

);

localparam UP = 2'b00;

localparam DOWN = 2'b01;

localparam LEFT = 2'b10;

localparam RIGHT = 2'b11;

localparam NONE = 2'b00;

localparam HEAD = 2'b01;

localparam BODY = 2'b10;

localparam WALL = 2'b11;

localparam RESTART = 2'b00;

localparam PLAY = 2'b10;

localparam SPEED1 = 50\_000\_000;

localparam SPEED2 = 25\_000\_000;

localparam SPEED3 = 12\_500\_000;

localparam SPEED4 = 6\_250\_000;

reg[31:0]cnt; //计时器

wire[1:0]direct;

reg [1:0]direct\_r; //寄存方向

assign direct = direct\_r;//寄存下一个方向

reg[1:0]direct\_next;

reg change\_to\_left;

reg change\_to\_right;

reg change\_to\_up;

reg change\_to\_down;

reg [5:0]cube\_x[15:0];

reg [5:0]cube\_y[15:0];//体长坐标 单位："格子"

reg [15:0]is\_exist; //用于控制身子的亮灭，即控制身子长度

reg addcube\_state;

reg [25:0]speed\_level;

assign head\_x = cube\_x[0];

assign head\_y = cube\_y[0];

//调速

always @(posedge clk )

begin

case(switch0+switch1+switch2)

0: speed\_level <= SPEED1;

1: speed\_level <= SPEED2;

2: speed\_level <= SPEED3;

3: speed\_level <= SPEED4;

endcase

end

always @(posedge clk or negedge rst) begin

if(!rst)

direct\_r <= RIGHT; //默认一出来向右移动

else if(game\_status == RESTART)

direct\_r <= RIGHT;

else

direct\_r <= direct\_next;

end

always @(posedge clk or negedge rst) begin

if(!rst) begin

cnt <= 0;

cube\_x[0] <= 10;

cube\_y[0] <= 5;

cube\_x[1] <= 9;

cube\_y[1] <= 5;

cube\_x[2] <= 8;

cube\_y[2] <= 5;

cube\_x[3] <= 0;

cube\_y[3] <= 0;

cube\_x[4] <= 0;

cube\_y[4] <= 0;

cube\_x[5] <= 0;

cube\_y[5] <= 0;

cube\_x[6] <= 0;

cube\_y[6] <= 0;

cube\_x[7] <= 0;

cube\_y[7] <= 0;

cube\_x[8] <= 0;

cube\_y[8] <= 0;

cube\_x[9] <= 0;

cube\_y[9] <= 0;

cube\_x[10] <= 0;

cube\_y[10] <= 0;

cube\_x[11] <= 0;

cube\_y[11] <= 0;

cube\_x[12] <= 0;

cube\_y[12] <= 0;

cube\_x[13] <= 0;

cube\_y[13] <= 0;

cube\_x[14] <= 0;

cube\_y[14] <= 0;

cube\_x[15] <= 0;

cube\_y[15] <= 0;

hit\_wall <= 0;

hit\_body <= 0;//初始长度3 限长16 初始化时只显示3个长度-

//eat\_boom <= 0;

end

else if(game\_status == RESTART) begin

cnt <= 0;

cube\_x[0] <= 10;

cube\_y[0] <= 5;

cube\_x[1] <= 9;

cube\_y[1] <= 5;

cube\_x[2] <= 8;

cube\_y[2] <= 5;

cube\_x[3] <= 0;

cube\_y[3] <= 0;

cube\_x[4] <= 0;

cube\_y[4] <= 0;

cube\_x[5] <= 0;

cube\_y[5] <= 0;

cube\_x[6] <= 0;

cube\_y[6] <= 0;

cube\_x[7] <= 0;

cube\_y[7] <= 0;

cube\_x[8] <= 0;

cube\_y[8] <= 0;

cube\_x[9] <= 0;

cube\_y[9] <= 0;

cube\_x[10] <= 0;

cube\_y[10] <= 0;

cube\_x[11] <= 0;

cube\_y[11] <= 0;

cube\_x[12] <= 0;

cube\_y[12] <= 0;

cube\_x[13] <= 0;

cube\_y[13] <= 0;

cube\_x[14] <= 0;

cube\_y[14] <= 0;

cube\_x[15] <= 0;

cube\_y[15] <= 0;

hit\_wall <= 0;

hit\_body <= 0;//初始长度3 限长16 初始化时只显示3个长度

//eat\_boom <=0;

end

else begin

cnt <= cnt + 1;

if(cnt == speed\_level) begin //0.02us\*12'500'000=0.25s 每秒移动四次

cnt <= 0;

if(game\_status == PLAY) begin

if((direct == UP && cube\_y[0] == 1)|(direct == DOWN && cube\_y[0] == 28)|(direct == LEFT && cube\_x[0] == 1)|(direct == RIGHT && cube\_x[0] == 38))

hit\_wall <= 1; //撞到墙壁

// else if((cube\_x[0] == boom\_x) && (cube\_y[0] == {1'b0,boom\_y}))

// eat\_boom <= 1; //吃到了炸弹

else if((cube\_y[0] == cube\_y[1] && cube\_x[0] == cube\_x[1] && is\_exist[1] == 1)|

(cube\_y[0] == cube\_y[2] && cube\_x[0] == cube\_x[2] && is\_exist[2] == 1)|

(cube\_y[0] == cube\_y[3] && cube\_x[0] == cube\_x[3] && is\_exist[3] == 1)|

(cube\_y[0] == cube\_y[4] && cube\_x[0] == cube\_x[4] && is\_exist[4] == 1)|

(cube\_y[0] == cube\_y[5] && cube\_x[0] == cube\_x[5] && is\_exist[5] == 1)|

(cube\_y[0] == cube\_y[6] && cube\_x[0] == cube\_x[6] && is\_exist[6] == 1)|

(cube\_y[0] == cube\_y[7] && cube\_x[0] == cube\_x[7] && is\_exist[7] == 1)|

(cube\_y[0] == cube\_y[8] && cube\_x[0] == cube\_x[8] && is\_exist[8] == 1)|

(cube\_y[0] == cube\_y[9] && cube\_x[0] == cube\_x[9] && is\_exist[9] == 1)|

(cube\_y[0] == cube\_y[10] && cube\_x[0] == cube\_x[10] && is\_exist[10] == 1)|

(cube\_y[0] == cube\_y[11] && cube\_x[0] == cube\_x[11] && is\_exist[11] == 1)|

(cube\_y[0] == cube\_y[12] && cube\_x[0] == cube\_x[12] && is\_exist[12] == 1)|

(cube\_y[0] == cube\_y[13] && cube\_x[0] == cube\_x[13] && is\_exist[13] == 1)|

(cube\_y[0] == cube\_y[14] && cube\_x[0] == cube\_x[14] && is\_exist[14] == 1)|

(cube\_y[0] == cube\_y[15] && cube\_x[0] == cube\_x[15] && is\_exist[15] == 1))

hit\_body <= 1;//头的Y坐标=任一位身体的Y坐标 且 头的X坐标=任一位身体的X坐标 且 身体的该长度位存在 碰到身体

else begin

cube\_x[1] <= cube\_x[0];

cube\_y[1] <= cube\_y[0];

cube\_x[2] <= cube\_x[1];

cube\_y[2] <= cube\_y[1];

cube\_x[3] <= cube\_x[2];

cube\_y[3] <= cube\_y[2];

cube\_x[4] <= cube\_x[3];

cube\_y[4] <= cube\_y[3];

cube\_x[5] <= cube\_x[4];

cube\_y[5] <= cube\_y[4];

cube\_x[6] <= cube\_x[5];

cube\_y[6] <= cube\_y[5];

cube\_x[7] <= cube\_x[6];

cube\_y[7] <= cube\_y[6];

cube\_x[8] <= cube\_x[7];

cube\_y[8] <= cube\_y[7];

cube\_x[9] <= cube\_x[8];

cube\_y[9] <= cube\_y[8];

cube\_x[10] <= cube\_x[9];

cube\_y[10] <= cube\_y[9];

cube\_x[11] <= cube\_x[10];

cube\_y[11] <= cube\_y[10];

cube\_x[12] <= cube\_x[11];

cube\_y[12] <= cube\_y[11];

cube\_x[13] <= cube\_x[12];

cube\_y[13] <= cube\_y[12];

cube\_x[14] <= cube\_x[13];

cube\_y[14] <= cube\_y[13];

cube\_x[15] <= cube\_x[14];

cube\_y[15] <= cube\_y[14];

//身体运动算法 本长度位移动的下个坐标为下一个长度位当前坐标 运动节拍按分频后的节奏

case(direct)

UP: begin

if(cube\_y[0] == 1)

hit\_wall <= 1;

else

cube\_y[0] <= cube\_y[0]-1;

end

DOWN: begin

if(cube\_y[0] == 28)

hit\_wall <= 1;

else

cube\_y[0] <= cube\_y[0] + 1;

end

LEFT: begin

if(cube\_x[0] == 1)

hit\_wall <= 1;

else

cube\_x[0] <= cube\_x[0] - 1;

end

RIGHT: begin

if(cube\_x[0] == 38)

hit\_wall <= 1;

else

cube\_x[0] <= cube\_x[0] + 1;

end

endcase //根据按下按键判断是否撞墙 否则按规律改变头部坐标

end

end

end

end

end

always @(\*) begin //根据当前运动状态即按下键位判断下一步运动情况

direct\_next = direct;

case(direct)

UP: begin //根据按键进行三个方向的选择，这里是按键按下的时候，信号传导Direct\_next，然后由Direct\_next送给Direct\_r，最后再赋值给Direct

if(change\_to\_left)

direct\_next = LEFT;

else if(change\_to\_right)

direct\_next = RIGHT;

else

direct\_next = UP;

end

DOWN: begin

if(change\_to\_left)

direct\_next = LEFT;

else if(change\_to\_right)

direct\_next = RIGHT;

else

direct\_next = DOWN;

end

LEFT: begin

if(change\_to\_up)

direct\_next = UP;

else if(change\_to\_down)

direct\_next = DOWN;

else

direct\_next = LEFT;

end

RIGHT: begin

if(change\_to\_up)

direct\_next = UP;

else if(change\_to\_down)

direct\_next = DOWN;

else

direct\_next = RIGHT;

end

endcase

end

always @(posedge clk) begin //给四个按键赋值

if(left\_press == 1)

change\_to\_left <= 1;

else if(right\_press == 1)

change\_to\_right <= 1;

else if(up\_press == 1)

change\_to\_up <= 1;

else if(down\_press == 1)

change\_to\_down <= 1;

else begin

change\_to\_left <= 0;

change\_to\_right <= 0;

change\_to\_up <= 0;

change\_to\_down <= 0;

end

end

always @(posedge clk or negedge rst) begin

//吃下苹果没？，吃下则add\_cube==1，显示体长增加一位，"is\_exixt[cube\_num]<=1",让第cube\_num位"出现"

if(!rst) begin

is\_exist <= 16'd7;

cube\_num <= 3;

addcube\_state <= 0;//初始显示长度为3，is\_exist=0000\_0000\_0111,表示前三位出现

end

else if (game\_status == RESTART) begin

is\_exist <= 16'd7;

cube\_num <= 3;

addcube\_state <= 0;

end

else begin

case(addcube\_state) //判断是否吃下

0:begin

if(add\_cube) begin

cube\_num <= cube\_num + 1;

is\_exist[cube\_num] <= 1;

addcube\_state <= 1;//"吃下"信号

end

end

1:begin

if(!add\_cube)

addcube\_state <= 0;

end

endcase

end

end

reg[3:0]lox;

reg[3:0]loy;

always @(x\_pos or y\_pos ) begin

if(x\_pos >= 0 && x\_pos < 640 && y\_pos >= 0 && y\_pos < 480) begin

if(x\_pos[9:4] == 0 | y\_pos[9:4] == 0 | x\_pos[9:4] == 39 | y\_pos[9:4] == 29)

snake = WALL;//扫描墙

else if(x\_pos[9:4] == cube\_x[0] && y\_pos[9:4] == cube\_y[0] && is\_exist[0] == 1)

snake = (die\_flash == 1) ? HEAD : NONE;//扫描头

else if

((x\_pos[9:4] == cube\_x[1] && y\_pos[9:4] == cube\_y[1] && is\_exist[1] == 1)|

(x\_pos[9:4] == cube\_x[2] && y\_pos[9:4] == cube\_y[2] && is\_exist[2] == 1)|

(x\_pos[9:4] == cube\_x[3] && y\_pos[9:4] == cube\_y[3] && is\_exist[3] == 1)|

(x\_pos[9:4] == cube\_x[4] && y\_pos[9:4] == cube\_y[4] && is\_exist[4] == 1)|

(x\_pos[9:4] == cube\_x[5] && y\_pos[9:4] == cube\_y[5] && is\_exist[5] == 1)|

(x\_pos[9:4] == cube\_x[6] && y\_pos[9:4] == cube\_y[6] && is\_exist[6] == 1)|

(x\_pos[9:4] == cube\_x[7] && y\_pos[9:4] == cube\_y[7] && is\_exist[7] == 1)|

(x\_pos[9:4] == cube\_x[8] && y\_pos[9:4] == cube\_y[8] && is\_exist[8] == 1)|

(x\_pos[9:4] == cube\_x[9] && y\_pos[9:4] == cube\_y[9] && is\_exist[9] == 1)|

(x\_pos[9:4] == cube\_x[10] && y\_pos[9:4] == cube\_y[10] && is\_exist[10] == 1)|

(x\_pos[9:4] == cube\_x[11] && y\_pos[9:4] == cube\_y[11] && is\_exist[11] == 1)|

(x\_pos[9:4] == cube\_x[12] && y\_pos[9:4] == cube\_y[12] && is\_exist[12] == 1)|

(x\_pos[9:4] == cube\_x[13] && y\_pos[9:4] == cube\_y[13] && is\_exist[13] == 1)|

(x\_pos[9:4] == cube\_x[14] && y\_pos[9:4] == cube\_y[14] && is\_exist[14] == 1)|

(x\_pos[9:4] == cube\_x[15] && y\_pos[9:4] == cube\_y[15] && is\_exist[15] == 1))

snake = (die\_flash == 1) ? BODY : NONE;//扫描身体

else snake = NONE;

end

end

endmodule

(5) VGA\_top: VGA显示模块

这个模块主要就是用于在VGA屏幕上显示我们的游戏，有几个功能模块，比如死亡时的闪烁，以及如何把苹果绘制成有绿有红的样子，这2个技术都很关键。

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/01/23 23:55:53

// Design Name:

// Module Name: VGA\_top

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module VGA\_top(

input clk,

input rst,

input [1:0]snake,

input [5:0]apple\_x,

input [4:0]apple\_y,

input [5:0]boom\_x,

input [4:0]boom\_y,

output [9:0]x\_pos,

output [9:0]y\_pos,

output hsync,

output vsync,

output [11:0] color\_out

);

wire clk\_n;

clk\_unit myclk(

.clk(clk),

.rst(rst),

.clk\_n(clk\_n)

);

VGA\_Control VGA

(

.clk(clk\_n),

.rst(rst),

.hsync(hsync),

.vsync(vsync),

.snake(snake),

.color\_out(color\_out),

.x\_pos(x\_pos),

.y\_pos(y\_pos),

.apple\_x(apple\_x),

.apple\_y(apple\_y),

.boom\_x(boom\_x),

.boom\_y(boom\_y)

);

endmodule

VGA子模块(1):

module clk\_unit(

input clk,

input rst,

output clk\_n

);

reg clk\_n; //四分频时钟，25MHZ

reg clk\_tmp; //二分频时钟，50MHZ

always @(posedge clk\_tmp or posedge rst) begin

if (rst)

clk\_n <= 0;

else

clk\_n <= ~clk\_n;

end

always @(posedge clk or posedge rst) begin

if (rst)

clk\_tmp <= 0;

else

clk\_tmp <= ~clk\_tmp;

end

endmodule

VGA子模块(2):

module VGA\_Control

(

input clk,

input rst,

input [1:0]snake,

input [5:0]apple\_x,

input [4:0]apple\_y,

input [5:0]boom\_x,

input [4:0]boom\_y,

output reg[9:0]x\_pos,

output reg[9:0]y\_pos,

output reg hsync,

output reg vsync,

output reg [11:0] color\_out //red[0:3]=[0:3],green[0:3]=[4:7],blue[0:3]=[8:11]

);

reg [19:0]clk\_cnt; //列计数器

reg [9:0]line\_cnt; //行计数器

reg clk\_25M;

localparam NONE = 2'b00;

localparam HEAD = 2'b01;

localparam BODY = 2'b10;

localparam WALL = 2'b11;

localparam HEAD\_COLOR = 12'b1111\_1111\_0000;

localparam BODY\_COLOR = 12'b1111\_1111\_1111;

reg [3:0]lox;

reg [3:0]loy;

always@(posedge clk or posedge rst) begin //修改：negedge rst -> posedge rst

//进行行，场同步

if(rst) begin

clk\_cnt <= 0;

line\_cnt <= 0;

hsync <= 1;

vsync <= 1;

end

else begin

x\_pos <= clk\_cnt - 144;

y\_pos <= line\_cnt - 33;

if(clk\_cnt == 0) begin

hsync <= 0;

clk\_cnt <= clk\_cnt + 1;

end

else if(clk\_cnt == 96) begin

hsync <= 1; //行同步

clk\_cnt <= clk\_cnt + 1;

end

else if(clk\_cnt == 799) begin

clk\_cnt <= 0;

line\_cnt <= line\_cnt + 1;

end

else clk\_cnt <= clk\_cnt + 1;

if(line\_cnt == 0) begin

vsync <= 0;

end

else if(line\_cnt == 2) begin

vsync <= 1; //场同步

end

else if(line\_cnt == 521) begin

line\_cnt <= 0;

vsync <= 0;

end

if(x\_pos >= 0 && x\_pos < 640 && y\_pos >= 0 && y\_pos < 480) begin

lox = x\_pos[3:0];

loy = y\_pos[3:0];

if(x\_pos[9:4] == apple\_x && y\_pos[9:4] == apple\_y) //显示苹果

case({lox,loy})

8'b0000\_0000:color\_out = 12'b0000\_0000\_0000;

//第 1 列

8'b0001\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0001\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0001\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0001\_1010:color\_out = 12'b0000\_0000\_1111;

//第 2 列

8'b0010\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0010\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0010\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0010\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0010\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0010\_1011:color\_out = 12'b0000\_0000\_1111;

//第 3 列

8'b0011\_0001:color\_out = 12'b0000\_1111\_0000;

8'b0011\_0101:color\_out = 12'b0000\_0000\_1111;

8'b0011\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0011\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0011\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0011\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0011\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0011\_1011:color\_out = 12'b0000\_0000\_1111;

8'b0011\_1100:color\_out = 12'b0000\_0000\_1111;

//第 4 列

8'b0100\_0001:color\_out = 12'b0000\_1111\_0000;

8'b0100\_0101:color\_out = 12'b0000\_0000\_1111;

8'b0100\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0100\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0100\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0100\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0100\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0100\_1011:color\_out = 12'b0000\_0000\_1111;

8'b0100\_1100:color\_out = 12'b0000\_0000\_1111;

//第 5 列

8'b0101\_0010:color\_out = 12'b0000\_1111\_0000;

8'b0101\_0100:color\_out = 12'b0000\_0000\_1111;

8'b0101\_0101:color\_out = 12'b0000\_0000\_1111;

8'b0101\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0101\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1011:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1100:color\_out = 12'b0000\_0000\_1111;

8'b0101\_1101:color\_out = 12'b0000\_0000\_1111;

//第 6 列

8'b0110\_0010:color\_out = 12'b0000\_1111\_0000;

8'b0110\_0100:color\_out = 12'b0000\_0000\_1111;

8'b0110\_0101:color\_out = 12'b0000\_0000\_1111;

8'b0110\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0110\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1011:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1100:color\_out = 12'b0000\_0000\_1111;

8'b0110\_1101:color\_out = 12'b0000\_0000\_1111;

//第 7 列

8'b0111\_0011:color\_out = 12'b0000\_1111\_0000;

8'b0111\_0100:color\_out = 12'b0000\_0000\_1111;

8'b0111\_0101:color\_out = 12'b0000\_0000\_1111;

8'b0111\_0110:color\_out = 12'b0000\_0000\_1111;

8'b0111\_0111:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1000:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1001:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1010:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1011:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1100:color\_out = 12'b0000\_0000\_1111;

8'b0111\_1101:color\_out = 12'b0000\_0000\_1111;

//第 8 列

8'b1000\_0011:color\_out = 12'b0000\_1111\_0000;

8'b1000\_0100:color\_out = 12'b0000\_0000\_1111;

8'b1000\_0101:color\_out = 12'b0000\_0000\_1111;

8'b1000\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1000\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1011:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1100:color\_out = 12'b0000\_0000\_1111;

8'b1000\_1101:color\_out = 12'b0000\_0000\_1111;

//第 9 列

8'b1001\_0010:color\_out = 12'b0000\_1111\_0000;

8'b1001\_0100:color\_out = 12'b0000\_0000\_1111;

8'b1001\_0101:color\_out = 12'b0000\_0000\_1111;

8'b1001\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1001\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1011:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1100:color\_out = 12'b0000\_0000\_1111;

8'b1001\_1101:color\_out = 12'b0000\_0000\_1111;

//第 10 列

8'b1010\_0010:color\_out = 12'b0000\_1111\_0000;

8'b1010\_0100:color\_out = 12'b0000\_0000\_1111;

8'b1010\_0101:color\_out = 12'b0000\_0000\_1111;

8'b1010\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1010\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1011:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1100:color\_out = 12'b0000\_0000\_1111;

8'b1010\_1101:color\_out = 12'b0000\_0000\_1111;

//第 11 列

8'b1011\_0001:color\_out = 12'b0000\_1111\_0000;

8'b1011\_0101:color\_out = 12'b0000\_0000\_1111;

8'b1011\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1011\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1011\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1011\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1011\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1011\_1011:color\_out = 12'b0000\_0000\_1111;

8'b1011\_1100:color\_out = 12'b0000\_0000\_1111;

//第 12 列

8'b1100\_0001:color\_out = 12'b0000\_1111\_0000;

8'b1100\_0101:color\_out = 12'b0000\_0000\_1111;

8'b1100\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1100\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1100\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1100\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1100\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1100\_1011:color\_out = 12'b0000\_0000\_1111;

8'b1100\_1100:color\_out = 12'b0000\_0000\_1111;

//第 13 列

8'b1101\_0110:color\_out = 12'b0000\_0000\_1111;

8'b1101\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1101\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1101\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1101\_1010:color\_out = 12'b0000\_0000\_1111;

8'b1101\_1011:color\_out = 12'b0000\_0000\_1111;

//第 14 列

8'b1110\_0111:color\_out = 12'b0000\_0000\_1111;

8'b1110\_1000:color\_out = 12'b0000\_0000\_1111;

8'b1110\_1001:color\_out = 12'b0000\_0000\_1111;

8'b1110\_1010:color\_out = 12'b0000\_0000\_1111;

default:color\_out = 12'b0000\_0000\_0000;

endcase

else if(x\_pos[9:4] == boom\_x && y\_pos[9:4] == boom\_y)

case({lox,loy})

8'b0000\_0000:color\_out = 12'b0000\_0000\_0000;

8'b0010\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0010\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0011\_0111:color\_out = 12'b1111\_0000\_0000;

8'b0011\_1000:color\_out = 12'b1111\_0000\_0000;

8'b0011\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0011\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0011\_1011:color\_out = 12'b1111\_0000\_0000;

8'b0011\_1100:color\_out = 12'b1111\_0000\_0000;

8'b0100\_0110:color\_out = 12'b1111\_0000\_0000;

8'b0100\_0111:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1000:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1011:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1100:color\_out = 12'b1111\_0000\_0000;

8'b0100\_1101:color\_out = 12'b1111\_0000\_0000;

8'b0101\_0101:color\_out = 12'b1111\_0000\_0000;

8'b0101\_0110:color\_out = 12'b1111\_0000\_0000;

8'b0101\_0111:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1000:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1011:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1100:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1101:color\_out = 12'b1111\_0000\_0000;

8'b0101\_1110:color\_out = 12'b1111\_0000\_0000;

8'b0110\_0101:color\_out = 12'b1111\_0000\_0000;

8'b0110\_0110:color\_out = 12'b1111\_0000\_0000;

8'b0110\_0111:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1000:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1011:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1100:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1101:color\_out = 12'b1111\_0000\_0000;

8'b0110\_1110:color\_out = 12'b1111\_0000\_0000;

8'b0111\_0100:color\_out = 12'b1111\_0000\_0000;

8'b0111\_0101:color\_out = 12'b1111\_0000\_0000;

8'b0111\_0110:color\_out = 12'b1111\_0000\_0000;

8'b0111\_0111:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1000:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1001:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1010:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1011:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1100:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1101:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1110:color\_out = 12'b1111\_0000\_0000;

8'b0111\_1111:color\_out = 12'b1111\_0000\_0000;

8'b1000\_0010:color\_out = 12'b0000\_0000\_1111;

8'b1000\_0011:color\_out = 12'b0000\_0000\_1111;

8'b1000\_0100:color\_out = 12'b1111\_0000\_0000;

8'b1000\_0101:color\_out = 12'b1111\_0000\_0000;

8'b1000\_0110:color\_out = 12'b1111\_0000\_0000;

8'b1000\_0111:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1000:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1010:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1011:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1100:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1101:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1110:color\_out = 12'b1111\_0000\_0000;

8'b1000\_1111:color\_out = 12'b1111\_0000\_0000;

8'b1001\_0001:color\_out = 12'b0000\_0000\_1111;

8'b1001\_0101:color\_out = 12'b1111\_0000\_0000;

8'b1001\_0110:color\_out = 12'b1111\_0000\_0000;

8'b1001\_0111:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1000:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1010:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1011:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1100:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1101:color\_out = 12'b1111\_0000\_0000;

8'b1001\_1110:color\_out = 12'b1111\_0000\_0000;

8'b1010\_0000:color\_out = 12'b0000\_0000\_1111;

8'b1010\_0101:color\_out = 12'b1111\_0000\_0000;

8'b1010\_0110:color\_out = 12'b1111\_0000\_0000;

8'b1010\_0111:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1000:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1010:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1011:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1100:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1101:color\_out = 12'b1111\_0000\_0000;

8'b1010\_1110:color\_out = 12'b1111\_0000\_0000;

8'b1011\_0110:color\_out = 12'b1111\_0000\_0000;

8'b1011\_0111:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1000:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1010:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1011:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1100:color\_out = 12'b1111\_0000\_0000;

8'b1011\_1101:color\_out = 12'b1111\_0000\_0000;

8'b1100\_0111:color\_out = 12'b1111\_0000\_0000;

8'b1100\_1000:color\_out = 12'b1111\_0000\_0000;

8'b1100\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1100\_1010:color\_out = 12'b1111\_0000\_0000;

8'b1100\_1011:color\_out = 12'b1111\_0000\_0000;

8'b1100\_1100:color\_out = 12'b1111\_0000\_0000;

8'b1101\_1001:color\_out = 12'b1111\_0000\_0000;

8'b1101\_1010:color\_out = 12'b1111\_0000\_0000;

default:color\_out = 12'b0000\_0000\_0000;

endcase

else if(x\_pos[9:4] == boom\_x && y\_pos[9:4] == boom\_y)

color\_out = 12'b0000\_0000\_0000;

else if(snake == NONE)

color\_out = 12'b0000\_0000\_0000;

else if(snake == WALL)

color\_out = 12'b0000\_1100\_1111;

else if(snake == HEAD|snake == BODY) begin //根据当前扫描到的点是哪一部分输出相应颜色

case({lox,loy})

8'b0000\_0000:color\_out = 12'b0000\_0000\_0000;

default:color\_out = (snake == HEAD) ? HEAD\_COLOR : BODY\_COLOR;

endcase

end

end

else

color\_out = 12'b0000\_0000\_0000;

end

end

endmodule

(6) Key: 按键检测模块

这个模块主要就是用于检测玩家到底按下了哪个按键，以及相应的消抖操作，因为开发板上的按键按下去以后很容易造成抖动，对于蛇的操控很不好，所以需要消抖。

module Key

( input clk,

input rst,

input left,

input right,

input up,

input down,

output reg left\_key\_press,

output reg right\_key\_press,

output reg up\_key\_press,

output reg down\_key\_press

);

reg [31:0]clk\_cnt;

reg left\_key\_last;

reg right\_key\_last;

reg up\_key\_last;

reg down\_key\_last;

always@(posedge clk or negedge rst) begin

if(!rst) begin

clk\_cnt <= 0;

left\_key\_press <= 0;

right\_key\_press <= 0;

up\_key\_press <= 0;

down\_key\_press <= 0;

left\_key\_last <= 0;

right\_key\_last <= 0;

up\_key\_last <= 0;

down\_key\_last <= 0;

end

else begin

if(clk\_cnt == 5\_0000) begin

clk\_cnt <= 0;

left\_key\_last <= left;

right\_key\_last <= right;

up\_key\_last <= up;

down\_key\_last <= down;

if(left\_key\_last == 0 && left == 1)

left\_key\_press <= 1;

if(right\_key\_last == 0 && right == 1)

right\_key\_press <= 1;

if(up\_key\_last == 0 && up == 1)

up\_key\_press <= 1;

if(down\_key\_last == 0 && down == 1)

down\_key\_press <= 1;

end

else begin

clk\_cnt <= clk\_cnt + 1;

left\_key\_press <= 0;

right\_key\_press <= 0;

up\_key\_press <= 0;

down\_key\_press <= 0;

end

end

end

endmodule

(7) Seg\_Display: 数码管计分模块

最后一个模块主要就是用于在开发板的数码管上显示当前的游戏分数，即玩家控制贪吃蛇吃到了几个苹果，机制是每吃一个苹果加一分，死亡之后积分清零，是通过动态扫描显示到数码管上的。

module Seg\_Display

(

input clk,

input rst,

input add\_cube,

inout [1:0]game\_status,

output reg[7:0]seg\_out,

output reg[3:0]sel

);

localparam RESTART = 2'b00;

reg[15:0]point;

reg[31:0]clk\_cnt;

always@(posedge clk or negedge rst)

begin

if(!rst)

begin

seg\_out <= 0;

clk\_cnt <= 0;

sel <= 0;

end

else if (game\_status == RESTART) begin

seg\_out <= 0;

clk\_cnt <= 0;

sel <= 0;

end

else

begin

if(clk\_cnt <= 20\_0000)

begin

clk\_cnt <= clk\_cnt+1;

if(clk\_cnt == 5\_0000)

begin

sel <= 4'b0111;//一号数码管

case(point[3:0])

4'b0000:seg\_out <= 8'b1100\_0000;

4'b0001:seg\_out <= 8'b1111\_1001;

4'b0010:seg\_out <= 8'b1010\_0100;

4'b0011:seg\_out <= 8'b1011\_0000;

4'b0100:seg\_out <= 8'b1001\_1001;

4'b0101:seg\_out <= 8'b1001\_0010;

4'b0110:seg\_out <= 8'b1000\_0010;

4'b0111:seg\_out <= 8'b1111\_1000;

4'b1000:seg\_out <= 8'b1000\_0000;

4'b1001:seg\_out <= 8'b1001\_0000;

default;

endcase

end

else if(clk\_cnt == 10\_0000)

begin

sel <= 4'b1011; //二号数码管

case(point[7:4])

4'b0000:seg\_out <= 8'b1100\_0000;

4'b0001:seg\_out <= 8'b1111\_1001;

4'b0010:seg\_out <= 8'b1010\_0100;

4'b0011:seg\_out <= 8'b1011\_0000;

4'b0100:seg\_out <= 8'b1001\_1001;

4'b0101:seg\_out <= 8'b1001\_0010;

4'b0110:seg\_out <= 8'b1000\_0010;

4'b0111:seg\_out <= 8'b1111\_1000;

4'b1000:seg\_out <= 8'b1000\_0000;

4'b1001:seg\_out <= 8'b1001\_0000;

default;

endcase

end

else if(clk\_cnt == 15\_0000)

begin

sel <= 4'b1101;//三号数码管

case(point[11:8])

4'b0000:seg\_out <= 8'b1100\_0000;

4'b0001:seg\_out <= 8'b1111\_1001;

4'b0010:seg\_out <= 8'b1010\_0100;

4'b0011:seg\_out <= 8'b1011\_0000;

4'b0100:seg\_out <= 8'b1001\_1001;

4'b0101:seg\_out <= 8'b1001\_0010;

4'b0110:seg\_out <= 8'b1000\_0010;

4'b0111:seg\_out <= 8'b1111\_1000;

4'b1000:seg\_out <= 8'b1000\_0000;

4'b1001:seg\_out <= 8'b1001\_0000;

default;

endcase

end

else if(clk\_cnt == 20\_0000)

begin

sel <= 4'b1110;//四号数码管

case(point[15:12])

4'b0000:seg\_out <= 8'b1100\_0000;

4'b0001:seg\_out <= 8'b1111\_1001;

4'b0010:seg\_out <= 8'b1010\_0100;

4'b0011:seg\_out <= 8'b1011\_0000;

4'b0100:seg\_out <= 8'b1001\_1001;

4'b0101:seg\_out <= 8'b1001\_0010;

4'b0110:seg\_out <= 8'b1000\_0010;

4'b0111:seg\_out <= 8'b1111\_1000;

4'b1000:seg\_out <= 8'b1000\_0000;

4'b1001:seg\_out <= 8'b1001\_0000;

default;

endcase

end

end

else

clk\_cnt <= 0;

end

end

reg addcube\_state;

always@(posedge clk or negedge rst)

begin

if(!rst)//重制赋值

begin

point <= 0;

addcube\_state <= 0;

end

else if (game\_status == RESTART) begin

point <= 0;

addcube\_state <= 0;

end

else begin

case(addcube\_state)

0: begin

if(add\_cube) begin//进位系统

if(point[3:0] < 9)

point[3:0] <= point[3:0] + 1;

else begin

point[3:0] <= 0;

if(point[7:4] < 9)

point[7:4] <= point[7:4] + 1;

else begin

point[7:4] <= 0;

if(point[11:8] < 9)

point[11:8] <= point[11:8] + 1;

else begin

point[11:8] <= 0;

point[15:12] <= point[15:12] + 1;

end

end

end

addcube\_state <= 1;

end

end

1: begin

if(!add\_cube)

addcube\_state <= 0;

end

endcase

end

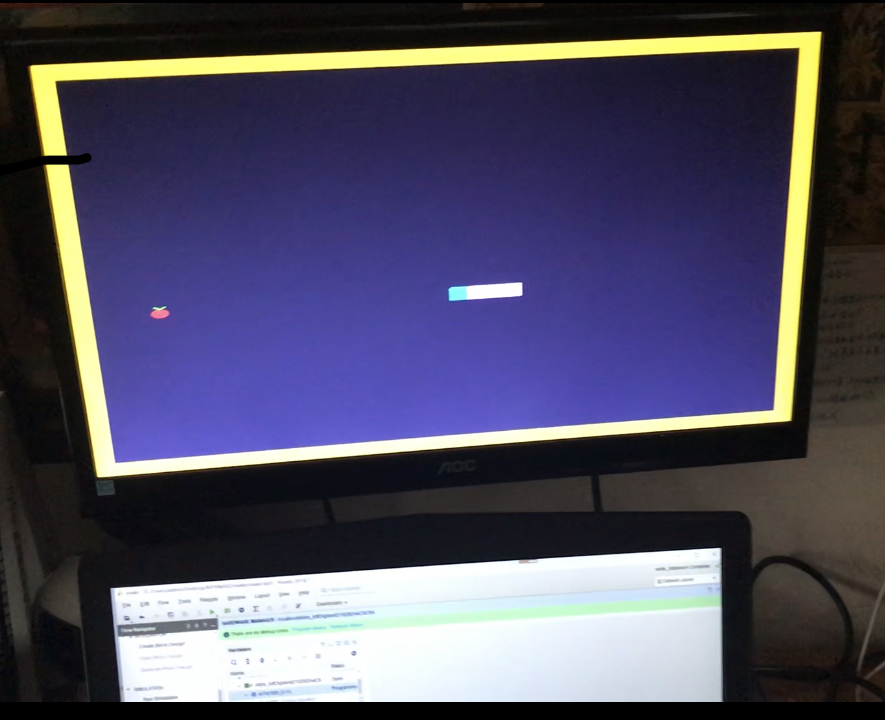
end

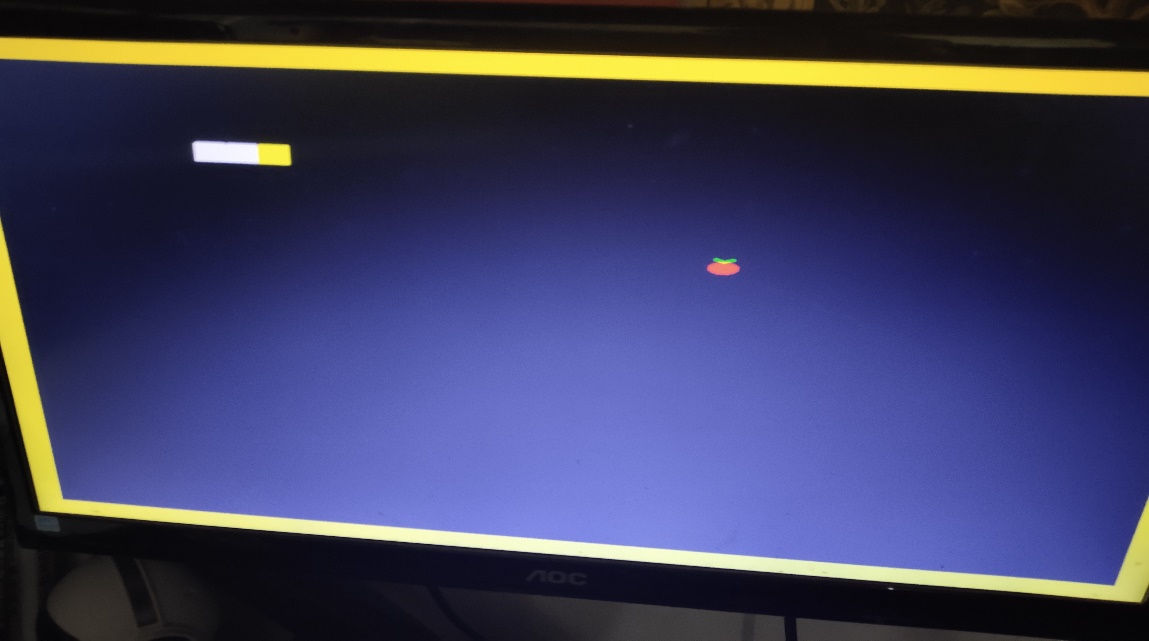
endmodule

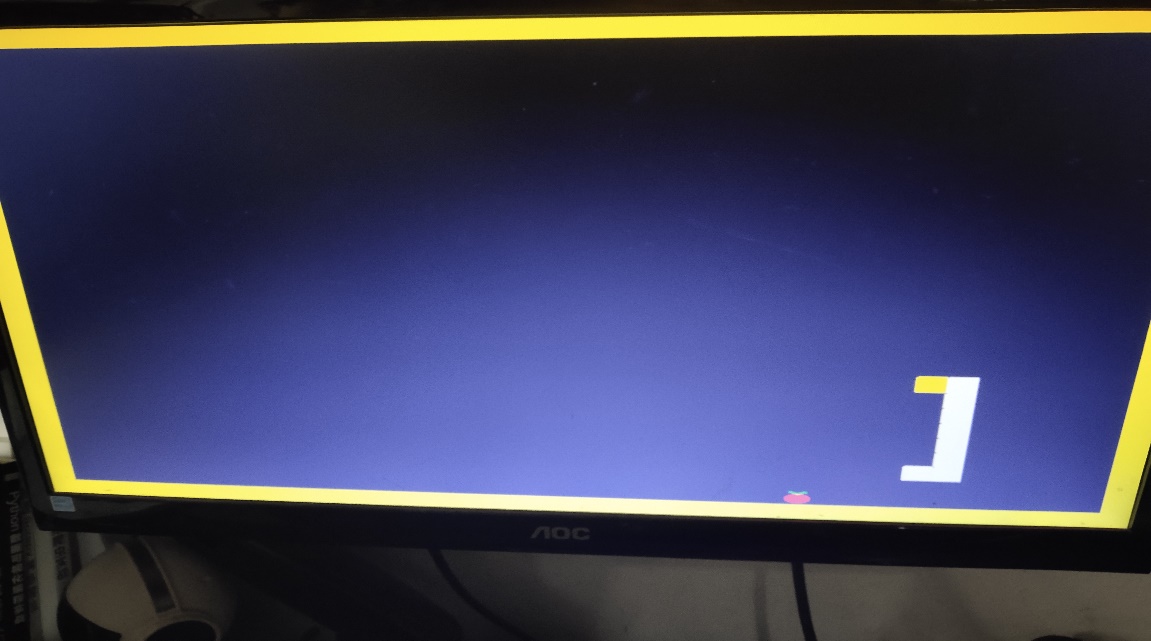
1. 项目操作说明

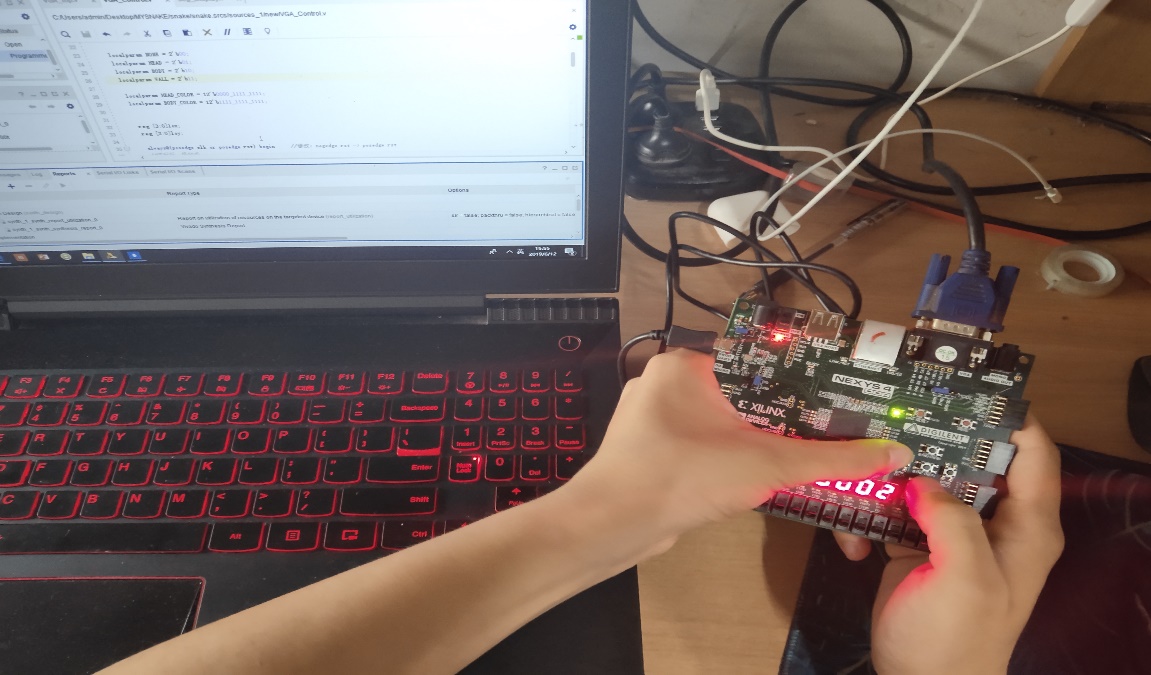
通过Nexys4 ddr开发板上的5个按键来控制游戏，其中上下左右控制蛇身的上下左右移动，中间的按键可以暂停游戏

**效果图：**

****

****

****

****

1. 综合项目以及微机原理实验课心得体会及建议

对于如何从无到有写一个完整项目有了更深的认识，先整体再局部，要有整体意识，要知道如何通过先搭一个框架，再慢慢添砖加瓦，来最后完成一个大的项目，这对我们以后无论是做课设还是去工作都很有帮助。还有就是，对于这门课有了更深的理解，也激发了我们的兴趣，愿意去努力探索和学习，一开始总觉得微机原理这门课很难，不是很好理解，但实际上你只要用心做实验，慢慢地就能从不会而变得很熟练，一些你之前觉得很难理解的东西也变得不过如此，这是我学的第二点。还有就是对于代码的调试更加熟练，也为大三做课设打好了一些基础，因为在进行编程时，往往构思好之后写代码的时间用的不是很多，但用在调试上的时间却很多，这是我们需要去关注的一点。一定要对于调试有一些自己的理解，要能够以最快的速度定位bug.

对于整门实验课的理解，对于接口技术的工作原理有了进一步的认识，对于查询和中断的工作方式有了更深的理解，学会了简单硬件电路的布置，对于GPIO IP核 有了了解。对于CPU的构造有了更深的理解，也知道了如何构建一个MIPS CPU，这种实验课我认为很有用，实现了由MIPS汇编语言编写程序。初步学会了使用MARS软件，能够看懂简单的程序的数据段与内存段，收获很大。也体会到了MIPS汇编语言的难处，C语言简单的循环结构代码，用汇编语言却需要几十行实现。

最后，这门实验课对于我来说确实是大有裨益，帮助我把之前没有搞懂，较难理解的接口和中断部分进行了梳理，虽然不能说完全吃透，但起码不再是一窍不通了，因此我觉得实验课的开设很有必要，也可以适当增加实验的比重。