浙江大学

Research project

吃豆人设计实验报告

Group members

蒋晨恺

丁昊

余南龙

王田园

Date：2017-1-13

摘要

作为计算机学院的学生，随着我们对软件技术的不断深入了解，对硬件技术的理解和应用，也变得十分重要，因为其对软件技术起到了关键性的决定作用。没有硬件的发展，软件技术会被牢牢束缚住。因此笔者根据对数字逻辑课程的理解，按照指导教师的要求，利用 sword 板和 VGA 技术，通过 Verilog 语言，制作可以在 VGA 显示器上显示的，能够人机互动的吃豆人游戏。

目录

一．　绪论

1.1 程序设计背景……………………………………4

1.2 国内外现况分析……………………………………4

二．程序设计说明

2.1 设计开发环境……………………………………5

2.2 输入输出交互选择………………………………5

2.3 核心模块设计……………………………………6

三．课程设计实现

3.1 各模块结构关系…………………………………8

3.2 模块分析及代码…………………………………9

3.3 仿真与调试………………………………………30

四．　系统测试验证与结果分析

4.1 功能测试与操作说明……………………………32

4.2 系统演示…………………………………………32

五．结论与展望

六．成员分工

一、绪论

1.1课程设计背景介绍

吃豆人是电子游戏历史上的经典街机游戏，由Namco公司的岩谷彻设计并由Midway Games在1980年发行。Pac-Man被认为是80年代最经典的街机游戏之一，游戏的主角小精灵的形象甚至被作为一种大众文化符号。它的开发商Namco也把这个形象作为其吉祥物和公司的标帜，一直沿用至今。该游戏的背景以黑色为主。当按动键盘上的方位键时，发现该黄色豆人符号可以行走，并且可以吞吃迷宫路径上的小黄豆，但遇到鬼面符号时就要被吃掉。

我们基于本学期对Verilog HDL语言的学习，及课后对PS / 2键盘和VGA显示的自主学习，在SWORD板实验平台上用Verilog HDL语言构建了一个“吃豆人”的游戏。该游戏拥有与传统吃豆人相似的设定与玩法，通过PS / 2键盘控制豆人方向，用VGA显示出游戏界面，实现了“吃豆人”小游戏。

1.2国内外现况分析

根据资料，可以发现利用Verilog制作VGA系列小游戏的不少，但制作吃豆人游戏的人很少。因为不像其他游戏可以利用七位段数码管和大量开关，吃豆人游戏的操作基本都是在VGA显示器上发生，以及通过ps2键盘来操纵游戏。由于ps2键盘端口设计较为麻烦，因此

以前即使是制作了吃豆人游戏，也是用其他方式代替 ps2键盘。因此这是本课程实验独特的一点。

二、 设计说明

A. 设计开发环境

实验平台:Xilinx Sword板

开发环境:Xilinx ISE

硬件描述语言:Verilog HDL

B. 输入输出交互选择

输入:PS／2键盘（上下左右四个方向键）

输出:VGA屏幕显示、四位 7 段数码管显示模块输出

C、具体设计

1. 基本布局：以硬件描述语言完成的电路设计，可以经过简单的综合与布局，快速的烧录至 FPGA 上进行测试，被用来实现一些基本的逻辑门电路或者更复杂一些的组合功能比如解码器或数学方程式。这些可编辑的元件里也包含记忆元件例如触发器或者其他更加完整的记忆块，作为标准的元器件来可以被多次调用。

2. Rom 存储：利用 xlinx 自带的 ip 核，建立 ROM，用于存储、读取和写入数据。
3. VGA 使用：通过 vga 扫描模块和颜色初始化模块，做到与 VGA 显示器同步扫描、使 VGA 显示器显示所需显示颜色的功能。
4. Ps2 键盘的使用：通过 ps2 端口模块，接受 ps2 的信号，从而对程序进行用户操作，包括移位、读写数据等。

D. 核心模块设计

1.PS / 2键盘模块

在该模块我们实现了通过键盘的输入，将键盘四个方向键的键盘码转化为ASCII码，该模块的输出就是不同按键的ASCII码，之后与其他模块的交互就是通过这个ASCII码以及一个判断按键是否终止的输出变量来实现的。

2.Object - disp模块

在此模块我们实现了VGA显示界面上的多种要素，包括墙体，豆子，豆人，怪物。

2.1墙体显示

墙体的背景我们通过ip核导入了一张红色砖墙的图片，在相应地扫描出豆人可以移动的通路，最后在VGA显示出一幅带有路径背景图片。

2.2豆子显示

由于每一颗豆子都是相同的，所以我们采用一个ROM来储存一颗白色豆子，然后将豆子分布在地图的各个角落中，即不同的坐标出，当且仅当豆人的位置和豆子的位置重合之时，豆子的显示才会消失。

2.3豆人显示

与豆子相似的，豆人也是通过一个ROM来储存，不过不同之处在于，我们需要设计四个ROM来实现豆子走向不同时嘴开口方向的面变化，所以我们设计了四个不同开口朝向不同的豆人，颜色为黄色，朝向的选择通过按键的输入来控制，而豆子的移动显示同样通过读入方向键的ASCII码并在时钟上升沿到来之时改变豆人的扫描左边，实现豆子的上下左右移动。

2.4怪物显示

怪物的显示同样通过ROM来实现，在游戏界面中有四个不同颜色的怪物，关于怪物的移动会在怪物移动模块详细说明。

3.移动模块

3.1怪物移动

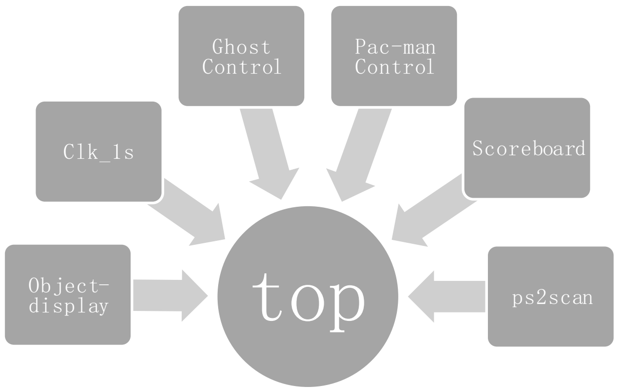
我们设置的怪物具有追踪豆人的功能，所以怪物的移动运用了AI，怪物会根据豆人的扫描坐标以及自己的扫描坐标判断自己的移动方向 ,选择一个方向能够靠近目标豆人。从而实现实时追踪豆人的功能。

3.2豆人控制移动

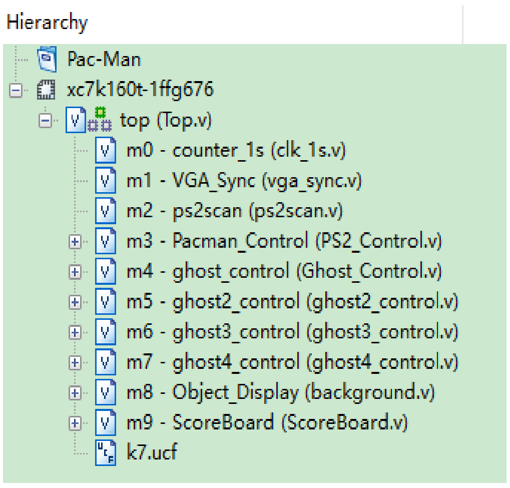
该模块读入键盘输入方向键的ACSII码以及按键是否终止来改变豆人的位置。例如在读到向上方向键的ASCII码值后，在ps2_state（即传入的按键状态值）的下降沿，其X方向的坐标值不变，Y方向减少一个单位值。但这一切的前提是我们将整个屏幕的所有像素点划分为一个个10*10的像素块，豆人的移动也因此是以这些像素块为单位实现的。

三、课程设计实现

3.1．各模块关系图



图表 1 各模块结构关系图



Hierarchy

```
Pac-Man
xc7k160t-1ffg676
  top (Top.v)
    m0 - counter_1s (clk_1s.v)
    m1 - VGA_Sync (vga_sync.v)
    m2 - ps2scan (ps2scan.v)
    m3 - Pacman_Control (PS2_Control.v)
    m4 - ghost_control (Ghost_Control.v)
    m5 - ghost2_control (ghost2_control.v)
    m6 - ghost3_control (ghost3_control.v)
    m7 - ghost4_control (ghost4_control.v)
    m8 - Object_Display (background.v)
    m9 - ScoreBoard (ScoreBoard.v)
    k7.ucf
```

3.2 模块分析及代码

A． Top主程序模块

1.输入输出

输入：

clk：时钟脉冲

ps2k_clk：PS2接口时钟信号

ps2k_data：PS2接口数据信号

rst：重置开关 sw：键盘使能开关

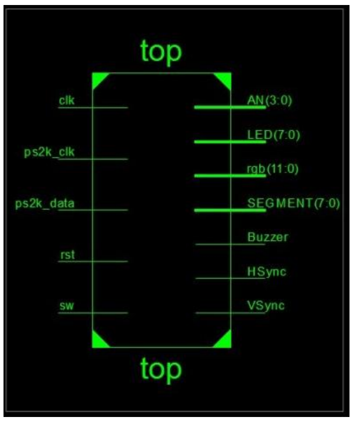
输出：

HSync、VSync、[7:0] LED、Buzzer

[11:0] rgb：十二位RGB码

[7:0] SEGMENT：七段数码管的显示信息输出

主要功能：主程序，调用各模块命令，并控制最外层的输入输出。



图表 2 top 模块输入输出端口示意图

2.核心代码

```
module top(
              input wire clk,  //100Mhz signal
        input wire rst, //reset signal
              input wire ps2k_clk,    //ps2 clk
              input wire ps2k_data, //ps2 data
              input wire sw,          //keyboard enable signal
        output wire HSync, VSync, //VGA sync
        output wire [11:0] rgb,   //RGB output
              output wire Buzzer,
              output wire [7:0] LED,
              output wire [3:0] AN,
              output wire [7:0] SEGMENT
);

assign LED = 8'b11111111;
assign Buzzer = 1'b1;

wire [9:0] p_x, p_y;   // pixel x,y
wire [9:0] pac_x, pac_y;//pacman location
wire p_tick;                //25Mhz signal
wire clk_1s;                //1s frequency divider
wire [9:0] pacman_l, pacman_r, pacman_b, pacman_t;//the left, right, bottom and top boundaries
of pacman
wire [9:0] ghost_x_l, ghost_x_r, ghost_y_t, ghost_y_b;//boundaries of ghost1
wire [9:0] ghost2_x_l,ghost2_x_r,ghost2_y_t,ghost2_y_b;//boundaries of ghost2
wire [9:0] ghost3_x_l,ghost3_x_r,ghost3_y_t,ghost3_y_b;//boundaries of ghost3
wire [9:0] ghost4_x_l,ghost4_x_r,ghost4_y_t,ghost4_y_b;//boundaries of ghost4
wire [7:0] ps2_byte;        //ascii code for keyboard buffer
wire ps2_state;             //keyboard input signal
wire [15:0] score;     //pacman score

//clk divider
counter_1s m0(
     .clk(clk),
     .clk_1s(clk_1s)
);

//VGA syncchronization
VGA_Sync m1(
     .clk(clk),
     .hs(HSync),
     .vs(VSync),
     .p_tick(p_tick),
     .p_x(p_x),
     .p_y(p_y)
);

//Keyboard buffer
ps2scan m2(
     .clk(clk),
     .rst(sw),
     .ps2k_clk(ps2k_clk),
     .ps2k_data(ps2k_data),
     .ps2_byte(ps2_byte),
     .ps2_state(ps2_state)
);

//Calculate the position of pacman
Pacman_Control m3(
   .rst(rst),
     .ps2_byte(ps2_byte),
     .ps2_state(ps2_state),
     .pacman_l(pacman_l),
     .pacman_r(pacman_r),
     .pacman_b(pacman_b),
     .pacman_t(pacman_t),
     .block1_x(pac_x),
     .block1_y(pac_y)
);

//The 1st ghost control
ghost_control m4(
    .clk(clk_1s),
    .sw(sw),
    .rst(rst),
    .GhostPosition_x(ghost_x_l),
      .GhostPosition_y(ghost_y_t),
      .ghost_r(ghost_x_r),
      .ghost_b(ghost_y_b),
      .PacManPosition_x(pac_x),
      .PacManPosition_y(pac_y)
);

//The 2nd ghost control
ghost2_control m5(
    .clk(clk_1s),
    .sw(sw),
    .rst(rst),
    .GhostPosition_x(ghost2_x_l),
      .GhostPosition_y(ghost2_y_t),
      .ghost_r(ghost2_x_r),
      .ghost_b(ghost2_y_b),
      .PacManPosition_x(pac_x),
      .PacManPosition_y(pac_y)
);

//The 3rd ghost control
ghost3_control m6(
    .clk(clk_1s),
    .sw(sw),
    .rst(rst),
    .GhostPosition_x(ghost3_x_l),
      .GhostPosition_y(ghost3_y_t),
      .ghost_r(ghost3_x_r),
      .ghost_b(ghost3_y_b),
      .PacManPosition_x(pac_x),
      .PacManPosition_y(pac_y)
);

//The 4th ghost control
ghost4_control m7(
    .clk(clk_1s),
    .sw(sw),
    .rst(rst),
    .GhostPosition_x(ghost4_x_l),
      .GhostPosition_y(ghost4_y_t),
      .ghost_r(ghost4_x_r),
      .ghost_b(ghost4_y_b),
      .PacManPosition_x(pac_x),
      .PacManPosition_y(pac_y)
);

//Display all objects on the monitor with VGA
Object_Display m8(
     .p_tick(p_tick),
    .sw(sw),
    .rst(rst),
     .p_x(p_x-120),
     .p_y(p_y-100),
     .ps2_byte(ps2_byte),
     .ps2_state(ps2_state),
     .pacman_l(pacman_l),
     .pacman_r(pacman_r),
     .pacman_b(pacman_b),
     .pacman_t(pacman_t),
     .ghost_x_l(ghost_x_l),
     .ghost_x_r(ghost_x_r),
     .ghost_y_t(ghost_y_t),
     .ghost_y_b(ghost_y_b),
     .ghost2_x_l(ghost2_x_l),
     .ghost2_x_r(ghost2_x_r),
     .ghost2_y_t(ghost2_y_t),
     .ghost2_y_b(ghost2_y_b),
     .ghost3_x_l(ghost3_x_l),
     .ghost3_x_r(ghost3_x_r),
     .ghost3_y_t(ghost3_y_t),
     .ghost3_y_b(ghost3_y_b),
     .ghost4_x_l(ghost4_x_l),
     .ghost4_x_r(ghost4_x_r),
     .ghost4_y_t(ghost4_y_t),
     .ghost4_y_b(ghost4_y_b),
     .rgb(rgb),
     .score(score)
);

//Display the score on Sword
ScoreBoard m9(
     .clk(clk),
     .score(score),
     .AN(AN),
     .SEGMENT(SEGMENT)
);

endmodule
```

B．ps2scan 模块

1.输入输出

输入：

clk：100M时钟信号

rst：复位信号

ps2k_clk：PS2接口时钟信号

ps2k_data：PS2接口数据信号

输出：

[7:0] ps2_byte：1byte键值，只做简单的按键扫描

ps2_state：键盘当前状态，ps2_state=1表示有键被按下

主要功能：

将键盘传入的键盘码转化为ASCII码输出，同时传出的还有ps2_state信号，该信号可以用来判断按键是否终止/松开。

2.核心代码

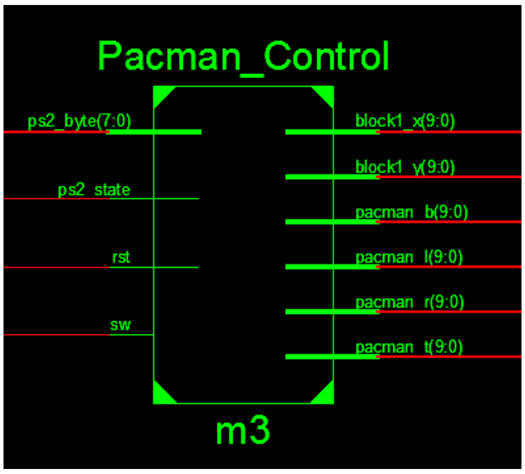
```
module ps2scan(clk, rst, ps2k_clk, ps2k_data, ps2_byte, ps2_state);
input wire clk; //100MHz clock signal
input wire rst;  //Reset signal
input ps2k_clk;   //Clock signal of the PS2 interface
input ps2k_data;  //Data signal of the PS2 interface
```

```
output[7:0] ps2_byte;    //The ASCII Code of the key which is pressed
output ps2_state;    //The status of the PS2, when a key is pressed, the value turns to 1
//-------------------------------------------
reg ps2k_clk_r0,ps2k_clk_r1,ps2k_clk_r2;  //ps2k_clk status registers
wire neg_ps2k_clk;   //ps2k_clk negtive edge flag
always @ (posedge clk or negedge rst) begin
   if(!rst) begin
         ps2k_clk_r0 <= 1'b0;
         ps2k_clk_r1 <= 1'b0;
         ps2k_clk_r2 <= 1'b0;
      end
   else begin  //A simple filtering circuit
         ps2k_clk_r0 <= ps2k_clk;
         ps2k_clk_r1 <= ps2k_clk_r0;
         ps2k_clk_r2 <= ps2k_clk_r1;
      end
end
assign neg_ps2k_clk = ~ps2k_clk_r1 & ps2k_clk_r2;
//-------------------------------------------
reg[7:0] ps2_byte_r;     //The register which stores the make code received from the PS2
reg[7:0] temp_data;  //The register which stores the current scan code received from the PS2
reg[3:0] num; //Counter register
always @ (posedge clk or negedge rst) begin
   if(!rst) begin
         num <= 4'd0;
         temp_data <= 8'd0;
      end
   else if(neg_ps2k_clk) begin
         case (num)
            4'd0:  num <= num+1'b1;
            4'd1:  begin
                     num <= num+1'b1;
                     temp_data[0] <= ps2k_data;
                  end
            4'd2:  begin
                     num <= num+1'b1;
                     temp_data[1] <= ps2k_data;
                  end
            4'd3:  begin
                     num <= num+1'b1;
                     temp_data[2] <= ps2k_data;
                  end
            4'd4:  begin
                     num <= num+1'b1;
                     temp_data[3] <= ps2k_data;
                  end
            4'd5:  begin
                     num <= num+1'b1;
                     temp_data[4] <= ps2k_data;
                  end
            4'd6:  begin
                     num <= num+1'b1;
                     temp_data[5] <= ps2k_data;
                  end
            4'd7:  begin
                     num <= num+1'b1;
                     temp_data[6] <= ps2k_data;
                  end
            4'd8:  begin
                     num <= num+1'b1;
                     temp_data[7] <= ps2k_data;
                  end
            4'd9:  begin
                     num <= num+1'b1;  //Parity bit
                  end
            4'd10: begin
                     num <= 4'd0;  //Clear num
                  end
            default: ;
            endcase
      end
end
reg key f0;      //A flag to show whether the break code has been transmitted or not
reg ps2_state_r;  //PS2 status, when a key is pressed, the value turns to 1

always @ (posedge clk or negedge rst) begin //Dealing with the scan code received from PS2
   if(!rst) begin
         key f0 <= 1'b0;
         ps2 state r <= 1'b0;
      end
   else if(num==4'd10&&neg ps2k clk) begin   //A byte data has just been transmitted
         if(temp data == 8'hf0) key f0 <= 1'b1;
                else if(temp_data == 8'he0) ;
         else begin
               if(!key f0) begin
                     ps2_state_r <= 1'b1;
                     ps2 byte r <= temp data;

                  end
                           else begin
                     ps2 state r <= 1'b0;
                     key_f0 <= 1'b0;


                  end
            end
      end
end
reg[7:0] ps2 asci=0;   //The register which stores ASCII Code corresponding to the make code
received from PS2
always @ (posedge clk) begin
   case (ps2 byte r)    //Convert the make code to the ASCII Code
          8'h75: ps2 asci = 8'h48; //Up
          8'h6B: ps2_asci = 8'h4B;    //Left
          8'h72: ps2 asci = 8'h50;    //Down
          8'h74: ps2 asci = 8'h4D;   //Right
      default: ps2_asci = 8'hFF;
      endcase
end
assign ps2_byte = ps2_asci;
assign ps2 state = ps2 state r;
endmodule
```

C. Pacman_control 模块

1.输入输出

主要功能：通过键盘传入的信号控制 Pacman 的当前位置



图表 3 Pacman_Control 输入输出端口示意图

2.核心代码

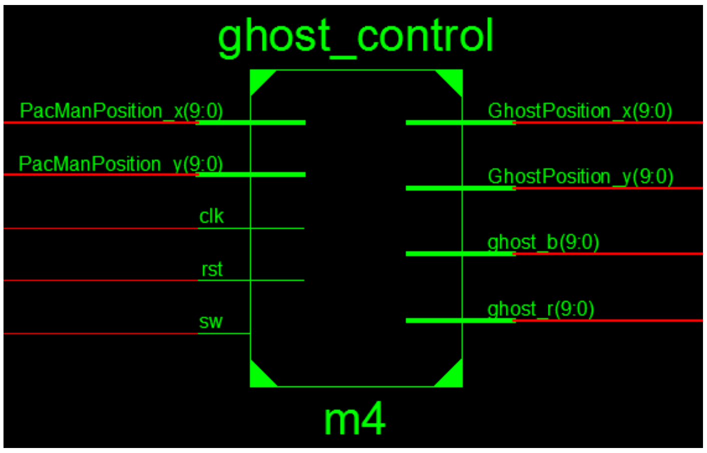
```
module Pacman Control(
     input [7:0] ps2 byte,    // input key ascii code
     input ps2_state,             //ps2 input done signal
   input wire rst,        //reset signal
     output wire [9:0] pacman l,
     output wire [9:0] pacman_r,
     output wire [9:0] pacman b,
     output wire [9:0] pacman t,
     output wire [9:0] block1_x,//currnt x location (0,39)
     output wire [9:0] block1 y//current y location (1,28)
   );
reg [9:0] block_x, block_y;
reg [9:0] block x reg, block y reg;
wire [3:0] valid;
assign block1_x=block_x_reg;
assign block1 y=block y reg;

assign pacman_l = 10*block_x_reg-10;//assign the left boundary of pacman
assign pacman r = 10*block x reg;//assign the right boundary of pacman
assign pacman b = 10*block y reg-10;//assign the bottom of pacman
assign pacman_t = 10*block_y_reg;//assign the top of pacman

always @(negedge ps2 state or posedge rst)
begin
     if (rst)//Initialize the location
     begin
          block_x_reg = 20;
          block y reg = 16;
     end
     else
          case (ps2 byte)
               8'h48: //Up
                    begin
                         block x reg = block x reg;
                         block y reg = block y reg - valid[2];
                    end
               8'h4B://Left
                    begin
                         block_x_reg = block_x_reg - valid[0];
                         block y reg = block y reg;
                    end
               8'h50://Down
                    begin
                         block x reg = block x reg;
```

```
                       block_y_reg = block_y_reg + valid[3];
                  end
              8'h4D://Right
                  begin
                       block_x_reg = block_x_reg + valid[1];
                       block_y_reg = block_y_reg;
                  end
              default://don't move
                  begin
                       block_x_reg = block_x_reg;
                       block_y_reg = block_y_reg;
                  end
         endcase
end

     //Find the direction that pacman can walk
     walk_detect m4(.block_x_reg(block_x_reg), .block_y_reg(block_y_reg),.valid(valid));
endmodule
```

D．ghost_control 模块

1.输入和输出

主要功能：这实际上是一个简单的 AI 算法，通过输入 Pacman 的当前位置来判断两者的相对位置，决定 ghost 的移动方向，组后输出 ghost 的当前位置。



图表 4 ghost_control 输入输出端口示意图

2.核心代码

```
module ghost_control(
     input wire clk,
     input wire sw,
     input rst,
         output wire[9:0] GhostPosition_x,//The left border of the ghost;
         output wire[9:0] GhostPosition_y,//The top border of the ghost;
         output wire[9:0] ghost_r,       //The right border of the ghost;
         output wire[9:0] ghost_b,       //The bottom border of the ghost;
         input wire[9:0] PacManPosition_x,//Input the location of Pacman;
         input wire[9:0] PacManPosition y //Input the location of Pacman;
   );
reg [9:0] ghost x, ghost y;
reg [9:0] ghost x reg, ghost y reg;

wire [3:0] validDirection;//The direction that the ghost can go when it comes to a point on the
map;
reg [3:0] BlinkyDirection;//The direction that the ghost decide to go when it comes to a point
on the map;
initial BlinkyDirection=4'b0010;//Initial the BlinkyDirection;
//Translate the coordinate of ghost into pixel;
assign GhostPosition x = 10*ghost x-10;
assign ghost r = 10*ghost x;
assign GhostPosition_y = 10*ghost_y-10;
assign ghost b = 10*ghost y;
//The direction that the ghost can go;
ghost_detect m5(
        .block x reg(ghost x reg),
        .block y reg(ghost y reg),
              .valid(validDirection)
              );
//4'bxxx1:left;
//4'bxx1x:right;
//4'bx1xx:up;
//4'b1xxx;down;

always@ * begin

         case(validDirection)
              4'b0000: BlinkyDirection <= 4'b0000;
              4'b0001: BlinkyDirection <= 4'b0001;
              4'b0010: BlinkyDirection <= 4'b0010;
              4'b0011: begin // can go Left or Right

                                 // if PacMan directly above go Left
                                 if(((ghost x reg  ==  PacManPosition x))  &&  ((ghost y reg  >
PacManPosition y)))
                                      BlinkyDirection <= 4'b0001;

                                 // if PacMan directly below go Right
                                 else     if(((ghost_x_reg     ==     PacManPosition_x))     &&
((PacManPosition y > ghost y reg)))
                                      BlinkyDirection <= 4'b0010;

                                 // if PacMan to the Right go Right
                                 else if(PacManPosition x > ghost x reg)
                                      BlinkyDirection <= 4'b0010;

                                 // if PacMan to the Left go Left
                                 else if(ghost_x_reg > PacManPosition_x)
                                      BlinkyDirection <= 4'b0001;

                                 else
                                      BlinkyDirection <= BlinkyDirection;

                            end
              4'b0100: BlinkyDirection <= 4'b0100;
              4'b0101: begin // can go Up or Left

                                 // if PacMan directly above go Up
                                 if((ghost x reg  ==  PacManPosition x)   &&  (ghost y reg  >
PacManPosition_y))
                                      BlinkyDirection <= 4'b0100;

                                 // if PacMan directly to the Left go Left
                                 else if((ghost y reg == PacManPosition y) && (ghost x reg >
PacManPosition x))
                                      BlinkyDirection <= 4'b0001;
```

```
                              // if PacMan directly to the Right go Up
                              else     if((ghost_y_reg     ==     PacManPosition_y)     &&
(PacManPosition x > ghost x reg))
                                  BlinkyDirection <= 4'b0100;

                              // if PacMan directly below go Left
                              else    if((ghost x reg    ==    PacManPosition x)       &&
(PacManPosition_y > ghost_y_reg))
                                  BlinkyDirection <= 4'b0001;

                              // if PacMan to the Left and Up and Up is closer go Up
                              else if((ghost y reg > PacManPosition y) &&
                                  (ghost x reg > PacManPosition x) &&
                                  ((ghost_y_reg  -  PacManPosition_y)  <  (ghost_x_reg  -
PacManPosition x)))

                                      BlinkyDirection <= 4'b0100;

                              // if PacMan to the Left and Up and Left is closer go Left
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (ghost x reg > PacManPosition x) &&
                                  ((ghost x reg  -  PacManPosition x)  <  (ghost y reg  -
PacManPosition_y)))

                                      BlinkyDirection <= 4'b0001;

                              // if PacMan to the Right and Down and Right is closer go Go
Up
                              else if((PacManPosition_y > ghost_y_reg) &&
                                  (PacManPosition x > ghost x reg) &&
                                  ((PacManPosition x - ghost x reg) < (PacManPosition y -
ghost_y_reg)))

                                      BlinkyDirection <= 4'b0100;

                              // if PacMan Up and to the Right go Up
                              else if((ghost y reg > PacManPosition y) && (PacManPosition x >
ghost_x_reg))
                                  BlinkyDirection <= 4'b0100;

                              //if PacMan Down and to the Left go Left
                              else  if((PacManPosition y  >  ghost y reg)  &&  (ghost x reg >
PacManPosition x))
                                  BlinkyDirection <= 4'b0001;


                              // if PacMan to the Right and Down and Down is closer go Go Left
                              else if((PacManPosition y > ghost y reg) &&
                                  (PacManPosition x > ghost x reg) &&
                                  ((PacManPosition_y - ghost_y_reg) < (PacManPosition_x -
ghost x reg)))

                                      BlinkyDirection <= 4'b0001;

                              else
                                  BlinkyDirection <= BlinkyDirection;

                          end
              4'b0110: begin // can go Up or Right

                              // if PacMan directly above go Up
                              if((ghost_x_reg  ==  PacManPosition_x)  &&  (ghost_y_reg  >
PacManPosition y))
                                  BlinkyDirection <= 4'b0100;

                              // if PacMan directly to the Right go Right
                              else     if((ghost y reg     ==     PacManPosition y)     &&
(PacManPosition_x > ghost_x_reg))
                                  BlinkyDirection <= 4'b0010;
```

```
                                        // if PacMan directly to the Left go Up
                                        else if((ghost_y_reg == PacManPosition_y) && (ghost_x_reg >
PacManPosition_x))
                                              BlinkyDirection <= 4'b0100;

                                        // if PacMan directly below go Right
                                        else    if((ghost_x_reg    ==    PacManPosition_x)        &&
(PacManPosition_y > ghost_y_reg))
                                              BlinkyDirection <= 4'b0010;

                                        // if PacMan to the Right and Up and Up is closer go Up
                                        else if((ghost_y_reg > PacManPosition_y) &&
                                              (PacManPosition_x > ghost_x_reg) &&
                                              ((ghost_y_reg - PacManPosition_y) < (PacManPosition_x -
ghost_x_reg)))

                                                    BlinkyDirection <= 4'b0100;

                                        // if PacMan to the Right and Up and Right is closer go Right
                                        else if((ghost_y_reg > PacManPosition_y) &&
                                              (PacManPosition_x > ghost_x_reg) &&
                                              ((PacManPosition_x  -  ghost_x_reg)  <  (ghost_y_reg  -
PacManPosition_y)))

                                                    BlinkyDirection <= 4'b0010;

                                        // if PacMan to the Left and Down and Left is closer go Up
                                        else if((PacManPosition_y > ghost_y_reg) &&
                                              (ghost_x_reg > PacManPosition_x) &&
                                              ((ghost_x_reg - PacManPosition_x) < (PacManPosition_y -
ghost_y_reg)))

                                                    BlinkyDirection <= 4'b0100;

                                        // if PacMan Up and to the Left go Up
                                        else  if((ghost_y_reg  >  PacManPosition_y)  &&  (ghost_x_reg >
PacManPosition_x))
                                              BlinkyDirection <= 4'b0100;

                                        //if PacMan Down and to the Right go Right
                                        else if((PacManPosition_y > ghost_y_reg) && (PacManPosition_x >
ghost_x_reg))
                                              BlinkyDirection <= 4'b0010;

                                        // if PacMan to the Left and Down and Down is closer go Right
                                        else if((PacManPosition_y > ghost_y_reg) &&
                                              (ghost_x_reg > PacManPosition_x) &&
                                              ((PacManPosition_y  -  ghost_y_reg)  <  (ghost_x_reg  -
PacManPosition_x)))

                                                    BlinkyDirection <= 4'b0010;

                                        else
                                              BlinkyDirection <= BlinkyDirection;


                                  end
                 4'b0111: begin // can go Left, Right, or Up

                                        // if PacMan directly above go Up
                                        if((ghost_x_reg  ==  PacManPosition_x)  &&  (ghost_y_reg  >
PacManPosition_y))
                                              BlinkyDirection <= 4'b0100;

                                        // if PacMan directly to the Right go Right
                                        else     if((ghost_y_reg     ==     PacManPosition_y)     &&
(PacManPosition_x > ghost_x_reg))
                                              BlinkyDirection <= 4'b0010;
```

```
                              // if PacMan directly to the Left go Left
                              else if((ghost_y_reg == PacManPosition_y) && (ghost_x_reg >
PacManPosition_x))
                                  BlinkyDirection <= 4'b0001;

                              // if PacMan directly below go Right
                              else    if((ghost_x_reg    ==    PacManPosition_x)    &&
(PacManPosition_y > ghost_y_reg))
                                  BlinkyDirection <= 4'b0010;

                              // if PacMan below and to the Right go Right
                              else if((PacManPosition_y > ghost_y_reg) && (PacManPosition_x >
ghost_x_reg))
                                  BlinkyDirection <= 4'b0010;

                              // if PacMan below and to the Left go Left
                              else if((PacManPosition_y > ghost_y_reg) && (ghost_x_reg >
PacManPosition_x))
                                  BlinkyDirection <= 4'b0001;

                              // if PacMan above and to the Right and Right is closer go Right
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (PacManPosition_x > ghost_x_reg) &&
                                  ((PacManPosition_x  -  ghost_x_reg)  <  (ghost_y_reg  -
PacManPosition_y)))

                                      BlinkyDirection <= 4'b0010;

                              // if PacMan above and to the Right and Up is closer go Up
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (PacManPosition_x > ghost_x_reg) &&
                                  ((ghost_y_reg - PacManPosition_y) < (PacManPosition_x -
ghost_x_reg)))

                                      BlinkyDirection <= 4'b0100;

                              // if PacMan above and to the Left and Left is closer go Left
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (ghost_x_reg > PacManPosition_x) &&
                                  ((ghost_x_reg  -  PacManPosition_x)  <  (ghost_y_reg  -
PacManPosition_y)))

                                      BlinkyDirection <= 4'b0001;

                              // if PacMan above and to the Left and Up is closer go Up
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (ghost_x_reg > PacManPosition_x) &&
                                  ((ghost_y_reg  -  PacManPosition_y)  <  (ghost_x_reg  -
PacManPosition_x)))

                                      BlinkyDirection <= 4'b0100;

                              else
                                  BlinkyDirection <= BlinkyDirection;

                          end
              4'b1000: BlinkyDirection <= 4'b1000;
              4'b1001: begin // can go Down or Left

                              // if PacMan directly below go Down
                              if((ghost_x_reg == PacManPosition_x) && (PacManPosition_y >
ghost_y_reg))
                                  BlinkyDirection <= 4'b1000;

                              // if PacMan directly to the Left go Left
                              else if((ghost_y_reg == PacManPosition_y) && (ghost_x_reg >
PacManPosition_x))
                                  BlinkyDirection <= 4'b0001;
```

```
                                 // if PacMan directly to the Right go Down
                                 else     if((ghost y reg     ==     PacManPosition y)     &&
(PacManPosition x > ghost x reg))
                                      BlinkyDirection <= 4'b1000;

                                 // if PacMan directly above go Left
                                 else if((ghost_x_reg == PacManPosition_x) && (ghost_y_reg >
PacManPosition y))
                                      BlinkyDirection <= 4'b0001;

                                 // if PacMan to the Left and Down and Down is closer go Down
                                 else if((PacManPosition y > ghost y reg) &&
                                      (ghost_x_reg > PacManPosition_x) &&
                                      ((PacManPosition y  -  ghost y reg)  <  (ghost x reg  -
PacManPosition x)))

                                           BlinkyDirection <= 4'b1000;

                                 // if PacMan to the Left and Down and Left is closer go Left
                                 else if((PacManPosition y > ghost y reg) &&
                                      (ghost x reg > PacManPosition x) &&
                                      ((ghost_x_reg - PacManPosition_x) < (PacManPosition_y -
ghost y reg)))

                                           BlinkyDirection <= 4'b0001;

                                 // if PacMan to the Right and Up and Right is closer go Down
                                 else if((ghost_y_reg > PacManPosition_y) &&
                                      (PacManPosition x > ghost x reg) &&
                                      ((PacManPosition x  -  ghost x reg)  <  (ghost y reg  -
PacManPosition_y)))

                                           BlinkyDirection <= 4'b1000;

                                 // if PacMan Down and to the Right go Down
                                 else if((PacManPosition y > ghost y reg) && (PacManPosition x >
ghost_x_reg))
                                      BlinkyDirection <= 4'b1000;

                                 // if PacMan Up and to the Left go Left
                                 else  if((ghost y reg  >  PacManPosition y)  &&  (ghost x reg >
PacManPosition x))
                                      BlinkyDirection <= 4'b0001;

                                 // if PacMan to the Right and Up and Up is closer go Left
                                 else if((ghost_y_reg > PacManPosition_y) &&
                                      (PacManPosition x > ghost x reg) &&
                                      ((ghost y reg - PacManPosition y) < (PacManPosition x -
ghost_x_reg)))

                                           BlinkyDirection <= 4'b0001;

                                 else
                                      BlinkyDirection <= BlinkyDirection;

                            end
              4'b1010: begin // can go Down or Right

                                 // if PacMan directly below go Down
                                 if((ghost x reg == PacManPosition x) && (PacManPosition y >
ghost_y_reg))
                                      BlinkyDirection <= 4'b1000;

                                 // if PacMan directly to the Right go Right
                                 else     if((ghost y reg     ==     PacManPosition y)     &&
(PacManPosition x > ghost x reg))
                                      BlinkyDirection <= 4'b0010;
```

```
                              // if PacMan directly above go Right
                              else if((ghost_x_reg == PacManPosition_x) && (ghost_y_reg > 
PacManPosition_y))
                                  BlinkyDirection <= 4'b0010;

                              // if PacMan directly to the Left go Down
                              else if((ghost_y_reg == PacManPosition_y) && (ghost_x_reg > 
PacManPosition_x))
                                  BlinkyDirection <= 4'b1000;

                              // if PacMan to the Right and Down and Down is closer go Down
                              else if((PacManPosition_y > ghost_y_reg) &&
                                  (PacManPosition_x > ghost_x_reg) &&
                                  ((PacManPosition_y - ghost_y_reg) < (PacManPosition_x - 
ghost_x_reg)))

                                      BlinkyDirection <= 4'b1000;

                              // if PacMan to the Right and Down and Right is closer go Right
                              else if((PacManPosition_y > ghost_y_reg) &&
                                  (PacManPosition_x > ghost_x_reg) &&
                                  ((PacManPosition_x - ghost_x_reg) < (PacManPosition_y - 
ghost_y_reg)))

                                      BlinkyDirection <= 4'b0010;

                              // if PacMan to the Left and Up and Left is closer go Down
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (ghost_x_reg > PacManPosition_x) &&
                                  ((ghost_x_reg  -  PacManPosition_x)  <  (ghost_y_reg  - 
PacManPosition_y)))

                                      BlinkyDirection <= 4'b1000;

                              // if PacMan Down and to the Left go Down
                              else  if((PacManPosition_y  >  ghost_y_reg)  &&  (ghost_x_reg > 
PacManPosition_x))
                                  BlinkyDirection <= 4'b1000;

                              // if Pacman up and to the Right go Right
                              else if((ghost_y_reg > PacManPosition_y) && (PacManPosition_x > 
ghost_x_reg))
                                  BlinkyDirection <= 4'b0010;

                              // if PacMan to the Left and Up and Up is closer go Right
                              else if((ghost_y_reg > PacManPosition_y) &&
                                  (ghost_x_reg > PacManPosition_x) &&
                                  ((ghost_y_reg  -  PacManPosition_y)  <  (ghost_x_reg  - 
PacManPosition_x)))

                                      BlinkyDirection <= 4'b0010;

                              else
                                  BlinkyDirection <= BlinkyDirection;

                          end
              4'b1011: begin // can go Left, Right, or Down

                          // if PacMan directly below go Down
                              if((ghost_x_reg == PacManPosition_x) && (PacManPosition_y > 
ghost_y_reg))
                                  BlinkyDirection <= 4'b1000;

                              // if PacMan directly to the Right go Right
                              else     if((ghost_y_reg     ==     PacManPosition_y)     && 
(PacManPosition_x > ghost_x_reg))
                                  BlinkyDirection <= 4'b0010;

                              // if PacMan directly to the Left go Left
```

```
                                  else if((ghost y reg == PacManPosition y) && (ghost x reg >
PacManPosition_x))
                                       BlinkyDirection <= 4'b0001;

                                  // if PacMan directly above go Right
                                  else if((ghost x reg == PacManPosition x) && (ghost y reg >
PacManPosition y))
                                       BlinkyDirection <= 4'b0010;

                                  // if PacMan above and to the Right go Right
                                  else    if((ghost_y_reg    >    PacManPosition_y)        &&
(PacManPosition x > ghost x reg))
                                       BlinkyDirection <= 4'b0010;

                                  // if PacMan above and to the Left go Left
                                  else if((ghost y reg > PacManPosition y) && (ghost x reg >
PacManPosition_x))
                                       BlinkyDirection <= 4'b0001;

                                  // if PacMan to the Right and Down and Down is closer go Down
                                  else if((PacManPosition y > ghost y reg) &&
                                       (PacManPosition x > ghost x reg) &&
                                       ((PacManPosition_y - ghost_y_reg) < (PacManPosition_x -
ghost x reg)))

                                            BlinkyDirection <= 4'b1000;

                                  // if PacMan to the Right and Down and Right is closer go Right
                                  else if((PacManPosition_y > ghost_y_reg) &&
                                       (PacManPosition x > ghost x reg) &&
                                       ((PacManPosition x - ghost x reg) < (PacManPosition y -
ghost_y_reg)))

                                            BlinkyDirection <= 4'b0010;

                                  // if PacMan to the Left and Down and Down is closer go Down
                                  else if((PacManPosition y > ghost y reg) &&
                                       (ghost_x_reg > PacManPosition_x) &&
                                       ((PacManPosition y  -  ghost y reg)  <  (ghost x reg  -
PacManPosition x)))

                                            BlinkyDirection <= 4'b1000;

                                  // if PacMan to the Left and Down and Left is closer go Left
                                  else if((PacManPosition y > ghost y reg) &&
                                       (ghost x reg - PacManPosition x) &&
                                       ((ghost_x_reg - PacManPosition_x) < (PacManPosition_y -
ghost y reg)))

                                            BlinkyDirection <= 4'b0001;

                                  else
                                       BlinkyDirection <= BlinkyDirection;

                             end
              4'b1100: begin // can go Down or Up

                                  // if PacMan directly to the Right go Up
                                  if((ghost_y_reg == PacManPosition_y) && (PacManPosition_x >
ghost x reg))
                                       BlinkyDirection <= 4'b0100;

                                  // if PacMan directly to the Left go Down
                                  else if((ghost y reg == PacManPosition y) && (ghost x reg >
PacManPosition_x))
                                       BlinkyDirection <= 4'b1000;

                                  // if PacMan below go Down
                                  else if(PacManPosition y > ghost y reg)
```

```
                                     BlinkyDirection <= 4'b1000;

                                 // if PacMan above go Up
                                 else if(ghost y reg > PacManPosition y)
                                     BlinkyDirection <= 4'b0100;

                                 else
                                     BlinkyDirection <= BlinkyDirection;

                            end
              4'b1101: begin // can go Left, Down, or Up

                            // if PacMan directly below go Down
                            if((ghost_x_reg  ==  PacManPosition_x)  &&  (PacManPosition_y  >
ghost y reg))
                                 BlinkyDirection <= 4'b1000;

                            // if PacMan directly above go Up
                            else  if((ghost x reg  ==  PacManPosition x)  &&  (ghost y reg  >
PacManPosition_y))
                                 BlinkyDirection <= 4'b0100;

                            // if PacMan directly to the Left go Left
                            else  if((ghost y reg  ==  PacManPosition y)  &&  (ghost x reg  >
PacManPosition x))
                                     BlinkyDirection <= 4'b0001;

                            // if PacMan directly to the Right go Down
                            else if((ghost_y_reg == PacManPosition_y) && (PacManPosition_x >
ghost x reg))
                                     BlinkyDirection <= 4'b1000;

                            // if PacMan above and to the Right go Up
                            else if((ghost y reg > PacManPosition y) && (PacManPosition x >
ghost_x_reg))
                                     BlinkyDirection <= 4'b0100;

                            // if PacMan below and to the Right go Down
                            else if((PacManPosition y > ghost y reg) && (PacManPosition x >
ghost x reg))
                                     BlinkyDirection <= 4'b1000;

                            // if PacMan to the Left and Down and Down is closer go Down
                            else if((PacManPosition_y > ghost_y_reg) &&
                                     (ghost x reg > PacManPosition x) &&
                                     ((PacManPosition y  -  ghost y reg)  <  (ghost x reg  -
PacManPosition_x)))

                                          BlinkyDirection <= 4'b1000;

                            // if PacMan to the Left and Down and Left is closer go Left
                            else if((PacManPosition y > ghost y reg) &&
                                     (ghost_x_reg > PacManPosition_x) &&
                                     ((ghost x reg - PacManPosition x) < (PacManPosition y -
ghost y reg)))

                                          BlinkyDirection <= 4'b0001;

                            // if PacMan to the Left and Up and Up is closer go Up
                            else if((ghost y reg > PacManPosition y) &&
                                 (ghost x reg > PacManPosition x) &&
                                 ((ghost_y_reg   -   PacManPosition_y)   <   (ghost_x_reg   -
PacManPosition x)))

                                          BlinkyDirection <= 4'b0100;

                            // if PacMan to the Left and Up and Left is closer go Left
                            else if((ghost_y_reg > PacManPosition_y) &&
                                     (ghost x reg > PacManPosition x) &&
                                        ((ghost_x_reg  -  PacManPosition_x)  <  (ghost_y_reg  -
PacManPosition_y)))

                                             BlinkyDirection <= 4'b0001;

                              else
                                        BlinkyDirection <= BlinkyDirection;

                              end
               4'b1110: begin // can go Right, Down, or Up

                              // if PacMan directly below go Down
                              if((ghost_x_reg  ==  PacManPosition_x)  &&  (PacManPosition_y  >
ghost_y_reg))
                                   BlinkyDirection <= 4'b1000;

                              // if PacMan directly above go Up
                              else  if((ghost_x_reg  ==  PacManPosition_x)  &&  (ghost_y_reg  >
PacManPosition_y))
                                   BlinkyDirection <= 4'b0100;

                              // if PacMan directly to the Right go Right
                              else if((ghost_y_reg == PacManPosition_y) && (PacManPosition_x >
ghost_x_reg))
                                        BlinkyDirection <= 4'b0010;

                              // if PacMan directly to the Left go Down
                              else  if((ghost_y_reg  ==  PacManPosition_y)  &&  (ghost_x_reg  >
PacManPosition_x))
                                        BlinkyDirection <= 4'b1000;

                              // if PacMan above and to the Left go Up
                              else  if((ghost_y_reg  >  PacManPosition_y)  &&  (ghost_x_reg  >
PacManPosition_x))
                                        BlinkyDirection <= 4'b0100;

                              // if PacMan below and to the Left go Down
                              else  if((PacManPosition_y  -  ghost_y_reg)  &&  (ghost_x_reg  >
PacManPosition_x))
                                        BlinkyDirection <= 4'b1000;

                              // if PacMan to the Right and Down and Down is closer go Down
                              else if((PacManPosition_y > ghost_y_reg) &&
                                        (PacManPosition_x > ghost_x_reg) &&
                                        ((PacManPosition_y - ghost_y_reg) < (PacManPosition_x -
ghost_x_reg)))

                                             BlinkyDirection <= 4'b1000;

                              // if PacMan to the Right and Down and Right is closer go Right
                              else if((PacManPosition_y > ghost_y_reg) &&
                                        (PacManPosition_x > ghost_x_reg) &&
                                        ((PacManPosition_x - ghost_x_reg) < (PacManPosition_y -
ghost_y_reg)))

                                             BlinkyDirection <= 4'b0010;

                              // if PacMan to the Right and Up and Up is closer go Up
                              else if((ghost_y_reg > PacManPosition_y) &&
                                   (PacManPosition_x > ghost_x_reg) &&
                                   ((ghost_y_reg  -  PacManPosition_y)  <  (PacManPosition_x  -
ghost_x_reg)))

                                             BlinkyDirection <= 4'b0100;

                              // if PacMan to the Right and Up and Right is closer go Right
                              else if((ghost_y_reg > PacManPosition_y) &&
                                        (PacManPosition_x > ghost_x_reg) &&
                                        ((PacManPosition_x  -  ghost_x_reg)  <  (ghost_y_reg  -
PacManPosition y)))

                                             BlinkyDirection <= 4'b0010;

                              else
                                        BlinkyDirection <= BlinkyDirection;

                              end
               4'b1111: begin // can go Right, Left, Down, or Up

                                   // if PacMan directly below go Down
                                   if((ghost x reg == PacManPosition x) && (PacManPosition y >
ghost y reg))
                                        BlinkyDirection <= 4'b1000;

                                   // if PacMan directly above go Up
                                   else if((ghost_x_reg == PacManPosition_x) && (ghost_y_reg >
PacManPosition y))
                                        BlinkyDirection <= 4'b0100;

                                   // if PacMan directly to the Right go Right
                                   else    if((ghost y reg    ==    PacManPosition y)    &&
(PacManPosition_x > ghost_x_reg))
                                        BlinkyDirection <= 4'b0010;

                                   // if PacMan directly to the Left go Left
                                   else if((ghost y reg == PacManPosition y) && (ghost x reg >
PacManPosition x))
                                        BlinkyDirection <= 4'b0001;

                                   // if PacMan to the Left and Up and Up is closer go Up
                                   else if((ghost_y_reg > PacManPosition_y) &&
                                        (ghost x reg > PacManPosition x) &&
                                        ((ghost y reg  -  PacManPosition y)  <  (ghost x reg  -
PacManPosition_x)))

                                             BlinkyDirection <= 4'b0100;

                                   // if PacMan to the Left and Up and Left is closer go Left
                                   else if((ghost y reg > PacManPosition y) &&
                                             (ghost_x_reg > PacManPosition_x) &&
                                             ((ghost x reg - PacManPosition x) < (ghost y reg -
PacManPosition y)))

                                             BlinkyDirection <= 4'b0001;

                                   // if PacMan to the Right and Up and Up is closer go Up
                                   else if((ghost y reg > PacManPosition y) &&
                                        (PacManPosition x > ghost x reg) &&
                                        ((ghost_y_reg - PacManPosition_y) < (PacManPosition_x -
ghost x reg)))

                                             BlinkyDirection <= 4'b0100;

                                   // if PacMan to the Right and Up and Right is closer go Right
                                   else if((ghost_y_reg > PacManPosition_y) &&
                                        (PacManPosition x > ghost x reg) &&
                                        ((PacManPosition x  -  ghost x reg)  <  (ghost y reg  -
PacManPosition_y)))

                                             BlinkyDirection <= 4'b0010;

                                   // if PacMan to the Right and Down and Down is closer go Down
                                   else if((PacManPosition y > ghost y reg) &&
                                        (PacManPosition_x > ghost_x_reg) &&
                                        ((PacManPosition y - ghost y reg) < (PacManPosition x -
ghost x reg)))

                                             BlinkyDirection <= 4'b1000;
```

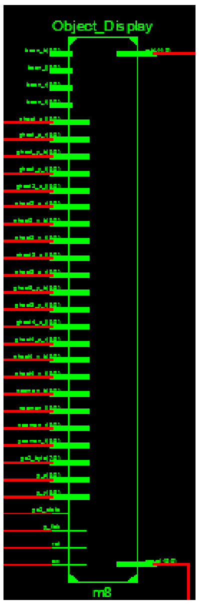
```
                                 // if PacMan to the Right and Down and Right is closer go Right
                                 else if((PacManPosition y > ghost y reg) &&
                                      (PacManPosition x > ghost x reg) &&
                                      ((PacManPosition_x - ghost_x_reg) < (PacManPosition_y -
ghost y reg)))

                                          BlinkyDirection <= 4'b0010;

                                 // if PacMan to the Left and Down and Down is closer go Down
                                 else if((PacManPosition_y > ghost_y_reg) &&
                                      (ghost x reg > PacManPosition x) &&
                                      ((PacManPosition y  -  ghost y reg)  <  (ghost x reg  -
PacManPosition_x)))

                                          BlinkyDirection <= 4'b1000;

                                 // if PacMan to the Left and Down and Left is closer go Left
                                 else if((PacManPosition y > ghost y reg) &&
                                      (ghost_x_reg > PacManPosition_x) &&
                                      ((ghost x reg - PacManPosition x) < (PacManPosition y -
ghost y reg)))

                                          BlinkyDirection <= 4'b0001;

                                 else
                                     BlinkyDirection <= BlinkyDirection;


                            end

         endcase
     end
     wire [3:0] valid;
     assign valid=BlinkyDirection;
     always @(posedge clk)
   begin
       case (BlinkyDirection)
          4'b0100:
             begin
                //block_y = block_y - 1;
                ghost x reg = ghost x;
               ghost y reg = ghost y - valid[2];
             end
          4'b0001:
             begin
                //block_x = block_x - 1;
                ghost x reg = ghost x - valid[0];
                ghost y reg = ghost y;
             end
          4'b1000:
             begin
                //block_y = block_y + 1;
                ghost x reg = ghost x;
                ghost y reg = ghost y + valid[3];
             end
          4'b0010:
             begin
                //block_x = block_x + 1;
                ghost x reg = ghost x + valid[1];
                ghost y reg = ghost y;
             end
          default:
             begin
                ghost_x_reg = ghost_x;
                ghost y reg = ghost y;
             end
       endcase
       if (rst) //reset;
```

```
        begin
           ghost_x = 2;
           ghost_y = 2;
           ghost_x_reg = 2;
           ghost_y_reg = 2;
           end
        else
        begin
           ghost_x = ghost_x_reg;
           ghost_y = ghost_y_reg;
        end
      end

endmodule
```

E．Object_Display 模块

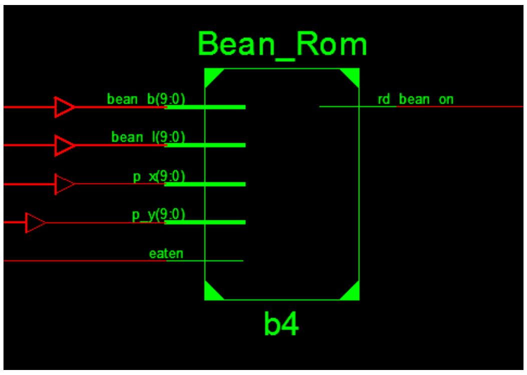
1.输入和输出

主要功能：这是一个综合性较强的模块，几乎涵盖了所有 VGA 上需要显示的元素，包括豆人，豆子，墙体，ghost。



图表 5 Object_Display 模块输入输出端口示意图

1.1Beam_Rom



图表 6 Beam 显示

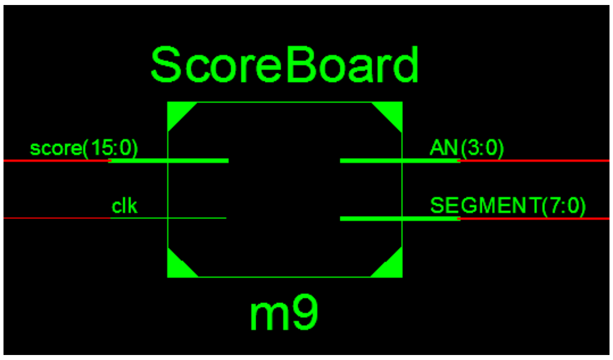
1.2 其余具体模块见工程文件

2.核心代码

```
module Bean Rom(
          input wire [9:0] p_x, p_y,
          input wire [9:0] bean b, bean l,//bean area
          input wire eaten,  //eaten or not signal
          output wire rd_bean_on
   );

     wire sq_bean_on;
     wire [11:0] bean rgb;
     wire [3:0] bean addr, bean col;
     wire bean_bit;

     assign                                sq bean on                                     =
(bean_l<p_x&&p_x<(bean_l+10))&&(bean_b<p_y&&p_y<(bean_b+10));//bean area
     assign bean addr = p y - bean b;//bean rom address
     assign bean col = p x - bean l;//bean rom colume
     assign bean_bit = bean_data[bean_col];//bean bit
     assign rd bean on = sq bean on&&bean bit&&eaten;//bean bit on

     reg [9:0] bean_data;
     always @*
          case (bean addr)
              4'h0: bean_data = 10'b0000000000;//
              4'h1: bean data = 10'b0000000000;//
              4'h2: bean data = 10'b0000110000;//    **
              4'h3: bean_data = 10'b0001111000;//   ****
              4'h4: bean data = 10'b0011111100;//  ******
              4'h5: bean_data = 10'b0011111100;//  ******
              4'h6: bean_data = 10'b0001111000;//   ****
              4'h7: bean data = 10'b0000110000;//    **
              4'h8: bean data = 10'b0000000000;//
              4'h9: bean_data = 10'b0000000000;//
          endcase
endmodule
```

F．Scoreboard 模块

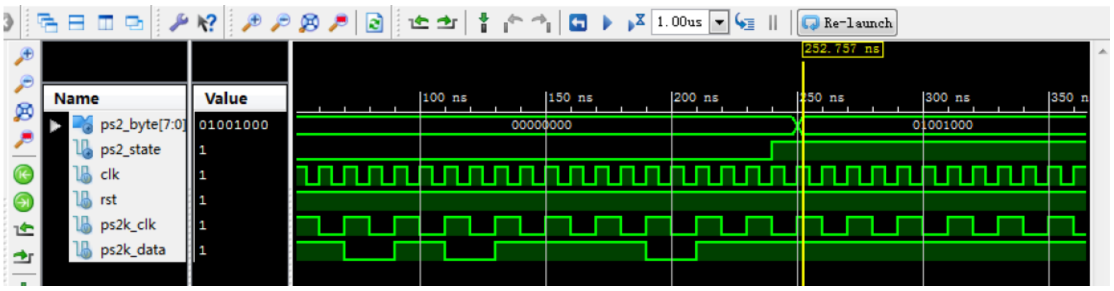
主要功能 :在其他模块通过豆子是否被吃掉会传出一个 eaten 信号，转化为 score 的值传入此模块，模块在七段数码管上显示得分。



图表 7 scoreboard 模块输入输出端口示意图

3.3 仿真与调试

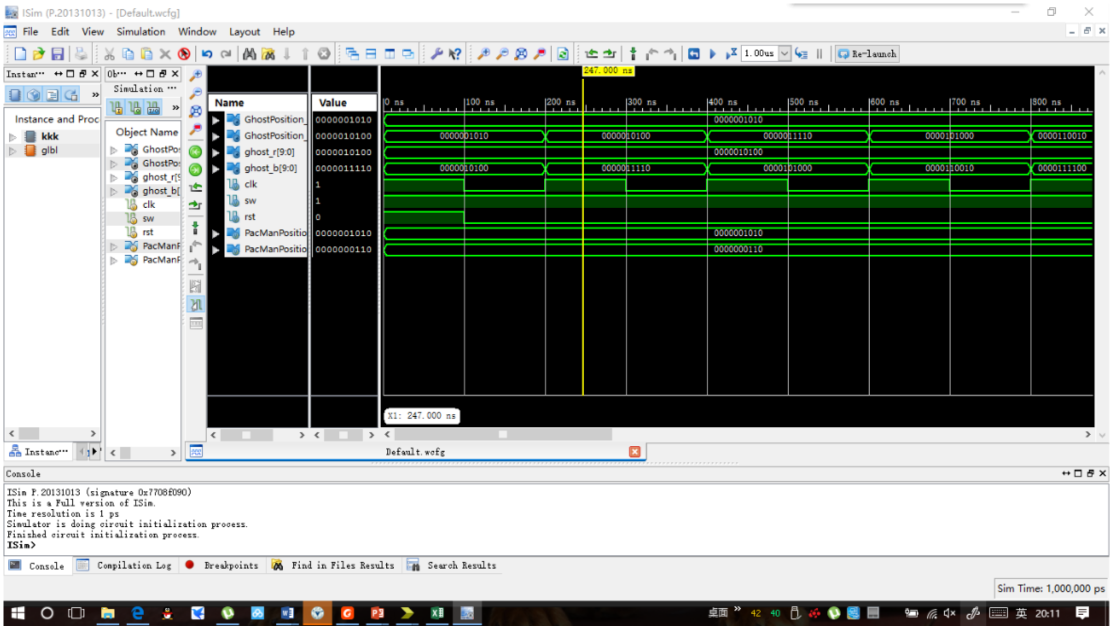
3.3.1 PS2 仿真



模拟波形分析：

该波形中最下面的 ps2k_data 是键盘扫描码的信号，这里模拟了键盘按键时通码的输入(方向键 Up 的通码是 E075，这里发送了 75)，ps2k_clk 是键盘自带时钟的信号，该信号只有在键盘有按键时会产生波形，下降沿有效，rst 是重置信号，这里赋值 1，clk 是 SWORD 板的时钟信号，上升沿有效。前半部分键盘通码发送时表示键盘有无按键按下的 ps2_state 信号为 0，输出的信号 ps2_byte 也为 0，当通码输入完毕后，模块内通过将通码转换为相应键 ASCII Code 的后八位输出，ps2_state 变为 1，ps2_byte 输出 01001000（方向键 Up 的 ASCII Code 为 E048，模块中取后八位 48 输出）。

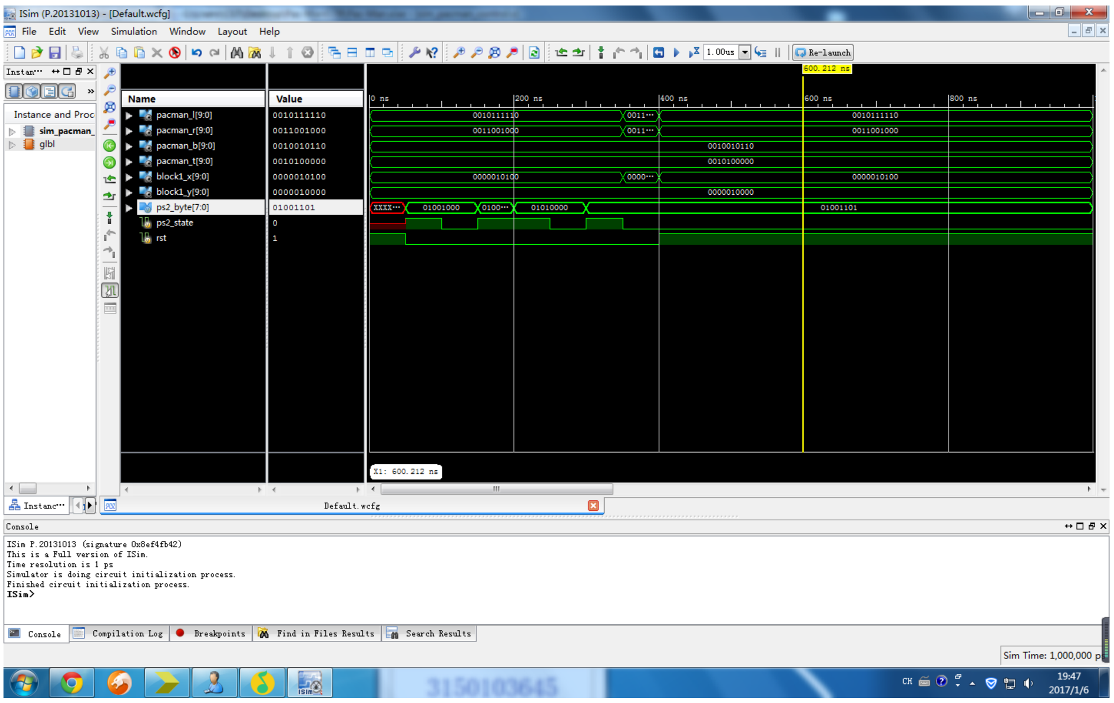
3.3.2 ghost 仿真



```
Name                 Value
GhostPosition_       0000001010
GhostPosition_       0000010100
ghost_r[9:0]         0000010100
ghost_b[9:0]         0000011110
clk                  1
sw                   1
rst                  0
PacManPositio        0000001010
PacManPositio        0000000110
```

```
ISim P.20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

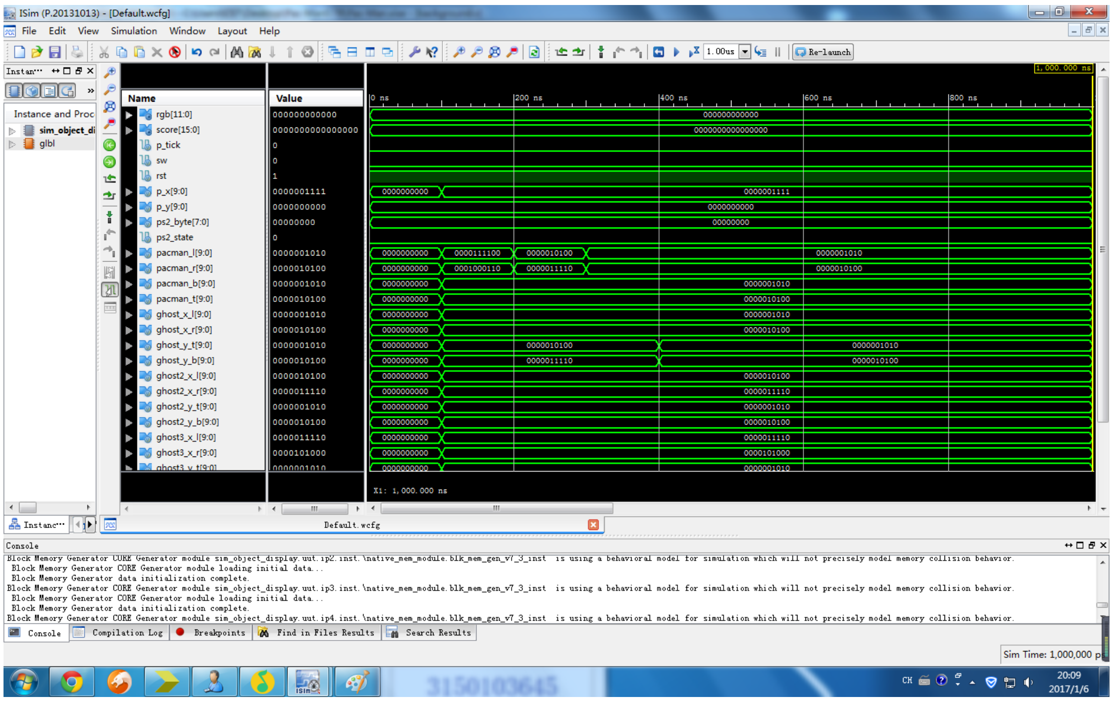
3.3.3 Pacman_control 仿真



```
Name                 Value
pacman_l[9:0]        0010111110
pacman_r[9:0]        0011001000
pacman_b[9:0]        0010010110
pacman_t[9:0]        0010100000
block1_x[9:0]        0000010100
block1_y[9:0]        0000010000
ps2_byte[7:0]        01001101
ps2_state            0
rst                  1
```

```
ISim P.20131013 (signature 0x8ef4fb42)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

3.3.4 Object_control 模块仿真



Name	Value
rgb[11:0]	000000000000
score[15:0]	0000000000000000
p_tick	0
sw	0
rst	1
p_x[9:0]	0000001111
p_y[9:0]	0000000000
ps2_byte[7:0]	00000000
ps2_state	0
pacman_l[9:0]	0000001010
pacman_r[9:0]	0000010100
pacman_b[9:0]	0000001010
pacman_t[9:0]	0000010100
ghost_x_l[9:0]	0000001010
ghost_x_r[9:0]	0000010100
ghost_y_t[9:0]	0000001010
ghost_y_b[9:0]	0000010100
ghost2_x_l[9:0]	0000010100
ghost2_x_r[9:0]	0000011110
ghost2_y_t[9:0]	0000001010
ghost2_y_b[9:0]	0000010100
ghost3_x_l[9:0]	0000011110
ghost3_x_r[9:0]	0000101000
ghost3_y_t[9:0]	0000001010

```
Block Memory Generator CORE Generator module sim_object_display.uut.ip2.inst.\native_mem_module.blk_mem_gen_v7_3_inst  is using a behavioral model for simulation which will not precisely model memory collision behavior.
 Block Memory Generator CORE Generator module loading initial data...
 Block Memory Generator data initialization complete.
Block Memory Generator CORE Generator module sim_object_display.uut.ip3.inst.\native_mem_module.blk_mem_gen_v7_3_inst  is using a behavioral model for simulation which will not precisely model memory collision behavior.
 Block Memory Generator CORE Generator module loading initial data...
 Block Memory Generator data initialization complete.
Block Memory Generator CORE Generator module sim_object_display.uut.ip4.inst.\native_mem_module.blk_mem_gen_v7_3_inst  is using a behavioral model for simulation which will not precisely model memory collision behavior.
```

Sim Time: 1,000,000 ps

四、系统测试验证与结果分析

A.功能测试

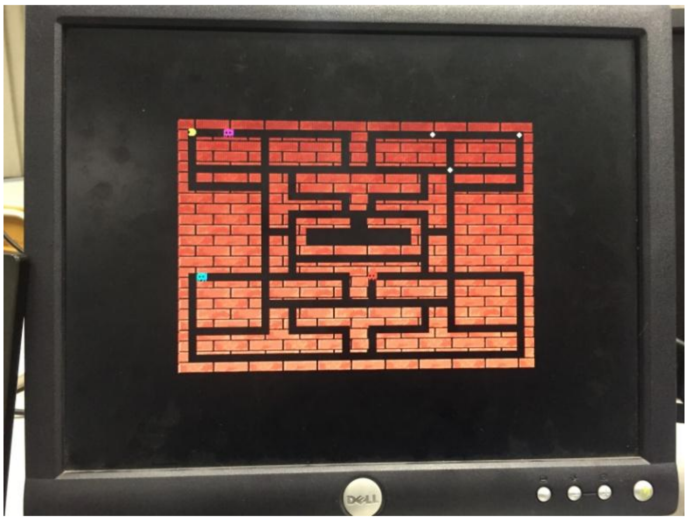
首先呈现开始界面。按动游戏开始开关，显示砖墙地图，界面上有四个颜色不同的怪物，屏幕中央有黄色豆人，地图上均匀分布着十六颗豆子，通过 ps2 键盘可以控制豆人上下左右移动，但不能穿墙。当豆人吃到豆子时，七段数码管上得分加一，当豆子吃完时，呈现游戏成功界面，当遇到怪物时 ，出现游戏失败界面，都可以按下 reset 重新开始游戏。另外，还有一个开关控制键盘按键的锁定。

B．系统演示

主要参照游戏视频，以下是一些特殊情况的界面。

1. 初始化界面

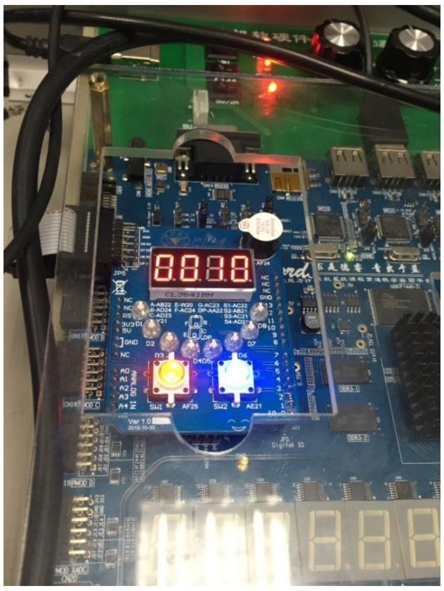
- 2. 开始游戏



- 3. 游戏失败

- 4. 游戏胜利

- 5. 计分显示



五、结论与展望

本次工程运用了我们在数字逻辑实验中学习过的绝大部分内容，并且还自学了 VGA 模块和 PS2 键盘模块。做出来的效果基本符合我们的预想。虽然可能由于期末的原因，有一些功能不能展开扩展，但是我们对这次工程基本上是满意的。

在这个过程中，我们以小组为单位，充分体会到团队意识的重要性，在遇到一些难题的时候，我们会一起讨论，来寻求更好的解决方法。我们也意识到，我们生活中遇到的一些看似简单的小游戏背后，是需要很多复杂的基础来支撑的。

我们也希望在接下来相关科目的学习中，我们能够在此基础上，更加深入地了解硬件的构造组成，能够收获更多。

六、成员分工

1. 分工明细：

蒋晨恺（组长）：VGA 显示、Pacman 控制、显示模块

王田园：ps2 键盘、设计报告撰写

余南龙：ps2 键盘扫描读入、计分模块

丁昊：VGA 显示、Ghost 控制、走动判断模块

2. 贡献比例：

每人 25%