浙江大学

本科实验报告

课程名称：数字逻辑设计

姓　　名：金艺轲，龙昱锦，肖思勃，盛　铭

学　　院：计算机科学与技术学院

指导教师：杨莹春

2024 年 6 月 26 日

浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 双人对战五子棋游戏设计

学生姓名： 金艺轲 专业： 工业设计 学号： 3230103159

学生姓名： 肖思勃 专业： 计算机科学与技术 学号： 3220105728

学生姓名： 龙昱锦 专业： 计算机科学与技术 学号： 3230105719

学生姓名： 盛 铭 专业： 自动化（控制） 学号： 3220104048

指导老师： 杨莹春 助教： 邱日宏 实验地点： 东 4-511 实验日期： 2024 年 6 月 26 日

一、游戏内容简介

五子棋的玩法大家都很熟悉，本课程设计实现了一个简单的双人对战五子棋游戏，其主要游戏规则如下：游戏棋盘上有 10 条竖线和 10 条横线，相互交叉形成一个 10*10 的棋盘，玩家双方可以在棋盘上轮流下棋，每当任意一个玩家在任意横、竖、斜线上连成连续的五颗棋子，则判定该玩家胜利，该局游戏结束，进入胜利界面，同时将胜利的玩家比分加一。这时，玩家可以通过控制 rst 开关，进入新的一轮对战。

在本游戏中，主要采用 PS2 键盘的上下左右按键控制游标的位置，通过按下 Enter 按键确认落子，通过 rst 开关清除板子上的所有棋子并开始新的一局，双方玩家的比分通过数码管显示。

二、设计原理

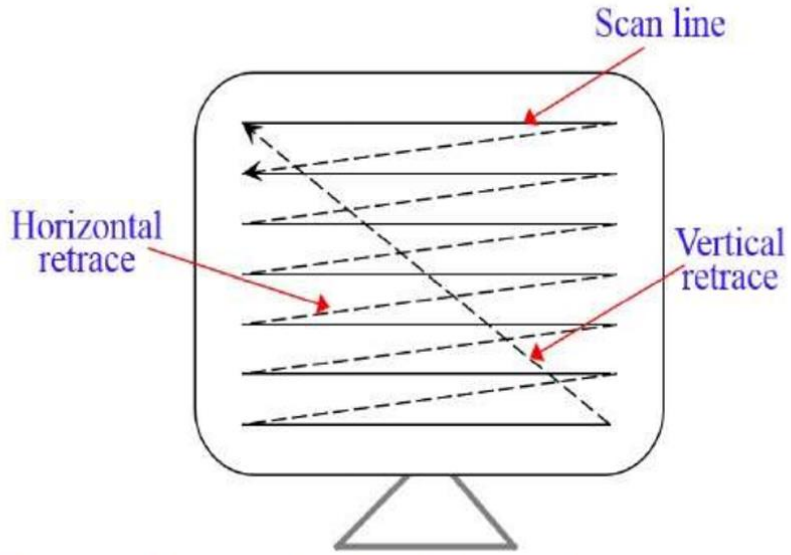
1、VGA 显示

本设计利用了实验板中的 VGA 显示屏，通过对于 VGA 时序控制信号以及输出颜色的控制，实现了显示屏信号的分时输出。VGA 是 IBM 在 1987 年随推出的一种视频传输标准，具有分辨率高、显示速率快、颜色丰富等优点，在彩色显示器领域得到了广泛的应用。VGA 在任何时刻都必须工作在某一显示模式之下，其显示模式分为字符显示模式和图形显示模式。

一般所使用的 VGA 显示器均为标准五输入类型，如下表所示：

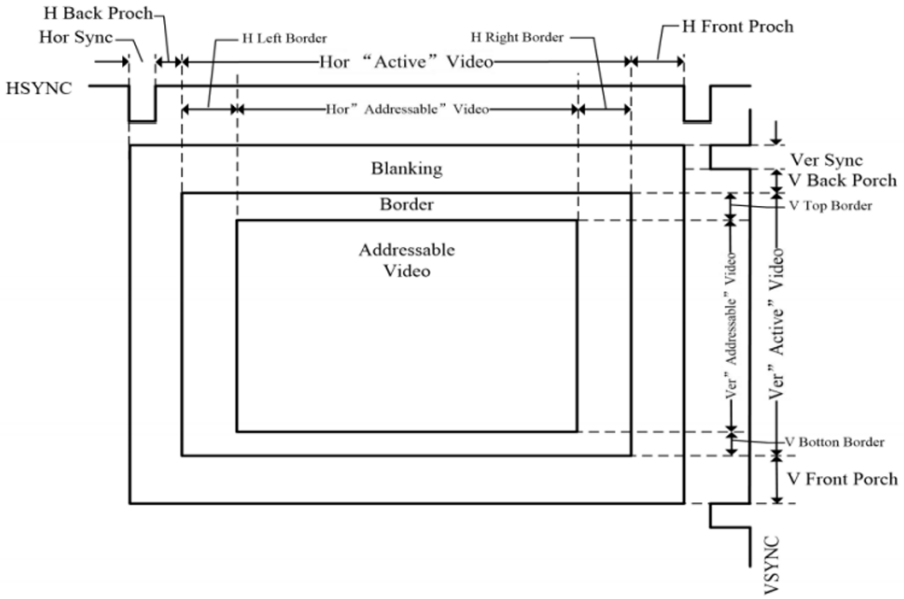
信号线	定义
HS	列同步信号（3.3V 电平）
VS	行同步信号（3.3V 电平）
R	红基色（0~0.714V 模拟信号）
G	绿基色（0~0.714V 模拟信号）
B	蓝基色（0~0.714V 模拟信号）

通过对行列同步信号进行控制，遍历输出各个像素点的颜色信息，能够输出完整图像。信号线中 HS 与 VS 用于控制显示器的显示频率，R、G 与 B 则用于控制显示器当前像素的颜色。VGA 显示器扫描方式从屏幕左上角一点开始，从左向右逐点扫描，每当扫描完一行时，电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，并开始下一帧。完成一行扫描的时间称为水平扫描时间， 其倒数称为行频率；完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率， 即屏幕的刷新频率，常见的有 60Hz，75Hz 等等，但标准的 VGA 显示的场频 60Hz。其扫描示意图如下图所示：



Raster scanning terminology for a single output frame.

总的来说，VGA 的时序主要包括行时序与场时序两个部分。其中，行时序主要包括行同步(Hor Sync) 、行消隐(Hor Back Porch)、行视频有效 (Hor Active Video) 和行前肩(Hor Front Porch) 这四个参数；而场时序主要包括场同步(Ver Sync) 、场消隐(Ver Back Porch) 、场视频有效(Ver Active Video)和场前肩(Ver Front Porch) 这四个参数。

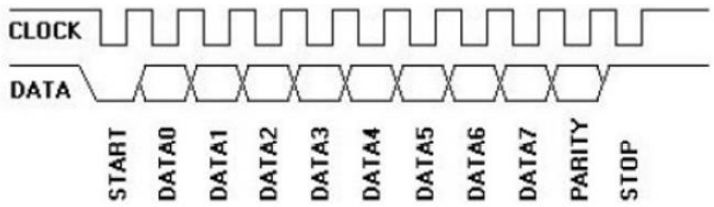


2、PS/2 键盘输入

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

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

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



三、关键模块设计思路

1、落棋的判定

在顶层模块中，设有 flag，[99:0]R，[99:0]U 三个变量。flag 用于存储当前玩家，如果为玩家 A，则 flag 为 0；如果为玩家 B，则 flag 为 1。R[99]对应棋盘上的 100 个点，对于位置为(i,j)的点，其对应 R[10*i+j]（数组 U 同理）。R 为 0 时，说明该位置没有棋子，R 为 1 时说明该位置有棋子；U 为 0 且 R 为 1 时，说明该位置被玩家 A 下棋，U 为 1 且 R 为 1 时说明该位置被 B 玩家下棋。

在收到落棋信号后，如果该位置已经有棋子，则无视命令；若无棋子，则放置棋子，完成落棋（注：c，d 代表游标所在位置的横竖坐标）：

```
1.   if(R[d*10+c]==0)
2.   begin
3.      R[d*10+c]=1;
4.       U[d*10+c]=flag[0];
5.       flag[0]=~flag[0];
```

```
    end
```

2、输赢判定

在输赢判断模块中，以 vga_clk 为时钟信号的上升沿为触发，在遍历棋盘（a，b 从(0,0)到(9,9)依次遍历）过程中，如果发现一个位置右侧、下侧、左下侧、右下侧存在连续 5 颗颜色相同的棋子，则返回将该棋子的颜色记录 winner 变量中，并设定 over=1，该轮游戏结束，进入胜利界面：

```
always@(posedge vga_clk)
begin
  if(rst)begin
  over[1:0]<=2'b00;
  end
  else begin
  if(a<=5&&U[b*10+a]==1&&U[b*10+a+1]==1&&U[b*10+a+2]==1&&U[b*10+a+3]==1&&U[b*10+a+4]==1)begin
    over[0]<=1;
    winner<= 1;
     end
    else if(b<=5&&U[b*10+a]==1&&U[b*10+a+10]==1&&U[b*10+a+20]==1&&U[b*10+a+30]==1&&U[b*10+a+40]==1)begin
    over[0]<=1;
    winner<= 1;
    end
    else if(a<=5&&b<=5&&U[b*10+a]==1&&U[b*10+a+11]==1&&U[b*10+a+22]==1&&U[b*10+a+33]==1&&U[b*10+a+44]==1)begin
    over[0]<=1;
    winner<= 1;
    end
    else if(b<=5&&a>=4&&U[b*10+a]==1&&U[b*10+a+9]==1&&U[b*10+a+18]==1&&U[b*10+a+27]==1&&U[b*10+a+36]==1) begin
    over[0]<=1;
    winner<= 1;
    end
  if(a<=5&&U[b*10+a]==0&&U[b*10+a+1]==0&&U[b*10+a+2]==0&&U[b*10+a+3]==0&&U[b*10+a+4]==0&&R[b*10+a]==1&&R[b*10+a+1]==1&&R[b*10+a+2]==1&&R[b*10+a+3]==1&&R[b*10+a+4]==1)begin //heng
    over[1]<=1;
    winner<= 0;
    end
    else if(b<=5&&U[b*10+a]==0&&U[b*10+a+10]==0&&U[b*10+a+20]==0&&U[b*10+a+30]==0&&U[b*10+a+40]==0&&(a<=10&&R[b*10+a]==1&&R[b*10+a+10]==1&&R[b*10+a+20]==1&&R[b*10+a+30]==1&&R[b*10+a+40]==1)) begin
    over[1]<=1;
    winner<= 0;
```

```
    end
    else
if(a<=5&&b<=5&&U[b*10+a]==0&&U[b*10+a+11]==0&&U[b*10+a+22]==0&&U[b*10+a+33]==0&&U[b*10+a+44]==0&&R[b*10+a]==1&&R[b*10+a+11]==1&&R[b*10+a+22]==1&&R[b*10+a+33]==1&&R[b*10+a+44]==1)
begin
    over[1]<=1;
    winner<= 0;
    end
    else
if(b<=5&&a>=4&&U[b*10+a]==0&&U[b*10+a+9]==0&&U[b*10+a+18]==0&&U[b*10+a+27]==0&&U[b*10+a+36]==0&&R[b*10+a]==1&&R[b*10+a+9]==1&&R[b*10+a+18]==1&&R[b*10+a+27]==1&&R[b*10+a+36]==1)
begin
    over[1]<=1;
    winner<= 0;
    end
  end
end
```

3、PS2 键盘输入

在本次大程中，为了方便操作，采用了 PS2 键盘输入控制游标和落棋操作。

PS2 模块定义如下：

```
module ps2(
        input clk,
        input rst,
        input ps2_clk,
        input ps2_data,
        output [9:0] data_out,
        output ready
    );

    reg ps2_clk_sign0, ps2_clk_sign1, ps2_clk_sign2, ps2_clk_sign3;
    wire negedge_ps2_clk;
    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            ps2_clk_sign0 <= 1'b0;
            ps2_clk_sign1 <= 1'b0;
            ps2_clk_sign2 <= 1'b0;
            ps2_clk_sign3 <= 1'b0;
        end
        else begin
            ps2_clk_sign0 <= ps2_clk;
            ps2_clk_sign1 <= ps2_clk_sign0;
            ps2_clk_sign2 <= ps2_clk_sign1;
            ps2_clk_sign3 <= ps2_clk_sign2;
        end
    end

```

```
    assign negedge_ps2_clk = !ps2_clk_sign0 & !ps2_clk_sign1 & ps2_clk_sign2 & ps2_clk_sign3;
    reg [3:0] cnt;
    always @(posedge clk or posedge rst) begin
        if(rst)
            cnt <= 4'd0;
        else if(cnt == 4'd11)
            cnt <= 4'd0;
        else if(negedge_ps2_clk)
            cnt <= cnt + 1'b1;
    end

    reg negedge_ps2_clk_shift;
    always @ (posedge clk) begin
        negedge_ps2_clk_shift <= negedge_ps2_clk;
    end

    reg [7:0] data_in;
    always @ (posedge clk or posedge rst) begin
        if(rst)
            data_in <= 8'd0;
        else if(negedge_ps2_clk_shift) begin
            case(cnt)
                4'd2 : data_in[0] <= ps2_data;
                4'd3 : data_in[1] <= ps2_data;
                4'd4 : data_in[2] <= ps2_data;
                4'd5 : data_in[3] <= ps2_data;
                4'd6 : data_in[4] <= ps2_data;
                4'd7 : data_in[5] <= ps2_data;
                4'd8 : data_in[6] <= ps2_data;
                4'd9 : data_in[7] <= ps2_data;
                default : data_in <= data_in;
            endcase
        end else
            data_in <= data_in;
    end

    reg key_break, key_done, key_expand;
    reg [9:0] data;
    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            key_break <= 1'b0;
            data <= 10'd0;
            key_done <= 1'b0;
            key_expand <= 1'b0;
        end
        else if(cnt == 4'd11) begin
            if(data_in == 8'hE0) begin
                key_expand <= 1'b1;
```

```
                key_break <= 1'b0;
            end
            else if(data_in == 8'hF0) begin
                key_break <= 1'b1;
                key_expand <= 1'b0;
            end
            else begin
                data <= {key_expand, key_break, data_in};
                key_done <= 1'b1;
                key_expand <= 1'b1;
                key_break <= 1'b0;
            end
        end
        else begin
            data <= data;
            key_done <= 1'b0;
            key_expand <= key_expand;
            key_break <= key_break;
        end
    end

    assign data_out = data;
    assign ready = key_done;

endmodule
```

在顶层模块中，我们对 PS2 模块的引用如下：

```
ps2 k1 (.clk(clk), .rst((1'b0)), .ps2_clk(ps2_clk), .ps2_data(ps2_data), .data_out(keyb), .ready (keybR));
```

keyb[9]变量中，我们可以判断是否有按键被按下，在数组 ps2_data[7:0]中，可以得到被按下按键的编号。我们已知 PS/2 键盘按键的 8 个数据位与按键的对应如下图所示：

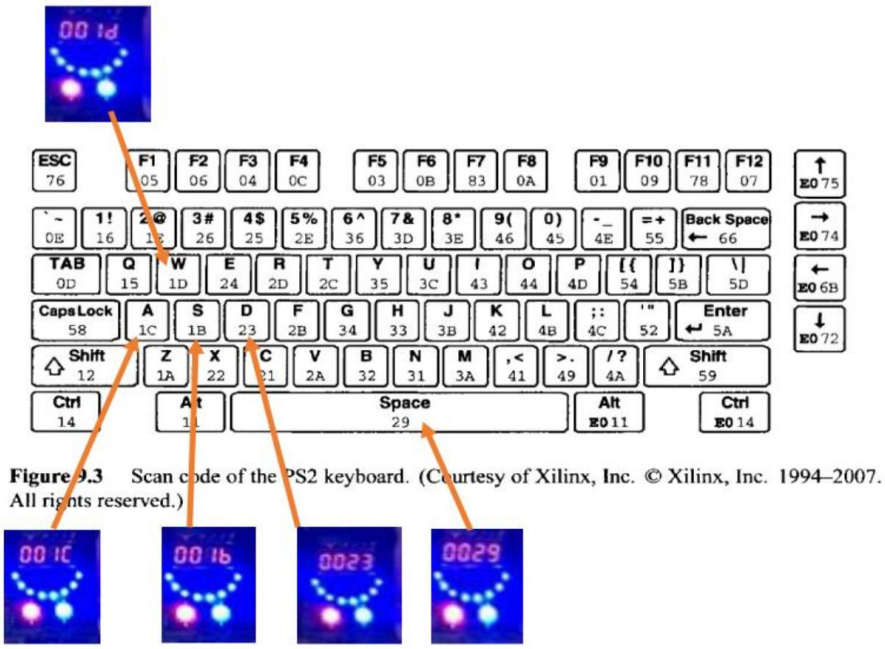


Figure 9.3 Scan code of the PS2 keyboard. (Courtesy of Xilinx, Inc. © Xilinx, Inc. 1994–2007. All rights reserved.)

因此，我们需要的上、下、左、右按键以及 Enter 键对应的数据位依次为：75H, 72H, 6BH, 74H, 59H。

我们在以下模块中对按键的行为进行描述，这一目标通过 case 语句来实现。

```
always@(posedge clk_50ms)
    if(keyb[9]==1)begin
    case(keyb[7:0])
         8'b01101011:begin
   if (ball_x_pos== 10'd240)
          ball_x_pos<=10'd600;
   else
         ball_x_pos<=ball_x_pos-10'd40;
                    end
         8'b01110100:begin
   if (ball_x_pos==10'd600)
         ball_x_pos<=10'd240;
   else
          ball_x_pos<= ball_x_pos+10'd40;
   end
         8'b01110101: begin
    if (ball_y_pos== 10'd90)
         ball_y_pos<=10'd450;
    else
         ball_y_pos<=ball_y_pos-10'd40;
    end
       8'b01110010: begin
    if(ball_y_pos== 10'd450)
    ball_y_pos<=10'd90;
    else
    ball_y_pos<=ball_y_pos+10'd40;
    end
       8'b01011010: begin
   if(R[d*10+c]==0)
   begin
      R[d*10+c]=1;
       U[d*10+c]=flag[0];
       flag[0]=~flag[0];
   end
   end
         default: rightbound<=10'd220;
     endcase
     end
```

4、VGA 显示

VGA 显示相关参数定义如下：

```
parameter
hsync_end = 10'd95,
hdat_begin = 10'd143,
```

```
hdat_end   = 10'd783,
hpixel_end = 10'd799,
vsync_end  = 10'd1,
vdat_begin = 10'd34,
vdat_end   = 10'd514,
vline_end  = 10'd524;
parameter ball_r=20;
```

```
parameter  WIDTH = 40,    //矩形长
           HEIGHT = 40,  //矩形宽
          //显示屏上总的显色区域
           DISV_TOP = 10'd470,
           DISV_DOWN =10'd70,
           DISH_LEFT = 10'd220,
           DISH_RIGHT = 10'd620;
          //变色区域右边界=左边界+格子宽度
           reg [9:0] ball_y_pos = 10'd90 ;
            reg [9:0] ball_x_pos = 10'd240 ;
            reg [9:0] rightbound = DISH_LEFT + WIDTH ;
```

vga_clk 定义如下：

```
always@(posedge clk)
begin
     if(cnt_clk == 1)
       begin
          vga_clk <= ~vga_clk;
          cnt_clk <= 0;
       end
      else
          cnt_clk <= cnt_clk + 1;
end
```

行扫描代码如下：

```
always@(posedge vga_clk)
begin
     if(hcount_ov)
        hcount <= 10'd0;
     else
        hcount <= hcount + 10'd1;
end
assign hcount_ov = (hcount == hpixel_end);
```

场扫描代码如下：

```
always@(posedge vga_clk)
begin
     if(hcount_ov)
     begin
        if(vcount_ov)
           vcount <= 10'd0;
        else
```

```
            vcount <= vcount + 10'd1;
        end
end
assign vcount_ov = (vcount == vline_end);
```

数据、同步信号输入定义如下：

```
assign dat_act = ((hcount >= hdat_begin) && (hcount < hdat_end)) && ((vcount > vdat_begin) && (vcount<vdat_end));
assign hsync   = (hcount > hsync_end);
assign vsync   = (vcount > vsync_end);
assign disp_rgb = (dat_act)?data:3'h000;
```

以下模块定义了棋子的绘制，当某位置 R 为 1 时，说明该位置上有棋子，以 ball_r 为半径画圆，根据 U 的不同数值，用不同颜色进行绘制，代表不同玩家：

```
always@(posedge vga_clk)
begin
      if ( (hcount - ball_x_pos)*(hcount - ball_x_pos) + (vcount- ball_y_pos)*(vcount - ball_y_pos) <= (ball_r * ball_r))
        x_dat<= 12'h0ff;
    else if (x[b*10+a]&&R[b*10+a]&&U[b*10+a])
    begin
            x_dat<= 12'hfff;
            y_dat <= 12'hf0f;
            end
    else if (x[b*10+a]&&R[b*10+a])
    begin
            x_dat<= 12'hfff;
            y_dat <= 12'hff0;

            end
     else
     begin
                x_dat<= 12'hfff;
               y_dat <= 12'hfff;
            end
end
```

在以下模块中，绘制了显示界面的竖线：

```
always@(posedge vga_clk)
begin
    if(hcount<=200||hcount>=640)
 v_dat <= 12'h000;//hei
    else if(hcount ==240||hcount ==280||hcount ==320||hcount ==360||hcount ==400||hcount ==440||hcount ==480||hcount ==520||hcount==560||hcount==600)
      v_dat <= 12'h000;//hei
   else
     v_dat <= 12'hfff;//bai
end
```

在以下模块中，绘制了显示界面的横线：

```
always@(posedge vga_clk)
begin
    if(vcount<=50||vcount>=490)
 h_dat <= 12'h000;//hei
    else if(vcount ==90||vcount ==130||vcount ==170||vcount==210 ||vcount==250||vcount ==290||vcount ==330||vcount==370 ||vcount==410||vcount==450)
      h_dat <= 12'h000;
   else
     h_dat <= 12'hfff;
end
```

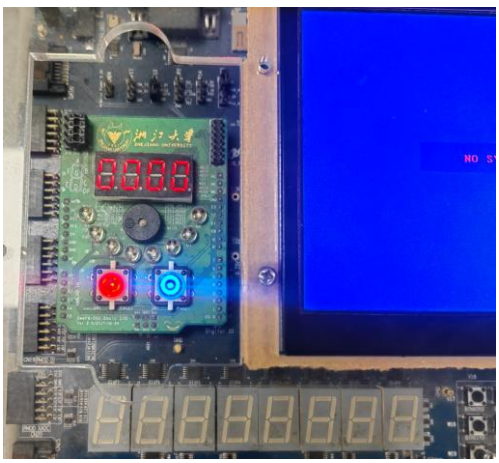
在以下模块中，绘制了 5 个与胜利玩家颜色相同的颜色的圆与一条横穿它的黑线，为胜利界面：

```
always@(posedge vga_clk)
begin
    if((hcount<=540&&hcount>=220)&&vcount<=282&&vcount>=278)
     z_dat<=12'h000;
   else if((hcount-320)*(hcount-320)+(vcount-280)*(vcount-280)<=400)
     z_dat<=winner?12'hf0f:12'hff0;
   else if((hcount-380)*(hcount-380)+(vcount-280)*(vcount-280)<=400)
     z_dat<=winner?12'hf0f:12'hff0;
   else if((hcount-440)*(hcount-440)+(vcount-280)*(vcount-280)<=400)
     z_dat<=winner?12'hf0f:12'hff0;
   else if((hcount-500)*(hcount-500)+(vcount-280)*(vcount-280)<=400)
     z_dat<=winner?12'hf0f:12'hff0;
   else if((hcount-260)*(hcount-260)+(vcount-280)*(vcount-280)<=400)
     z_dat<=winner?12'hf0f:12'hff0;
   else
     z_dat <= 12'hfff;
end
```

四、实验结果与现象

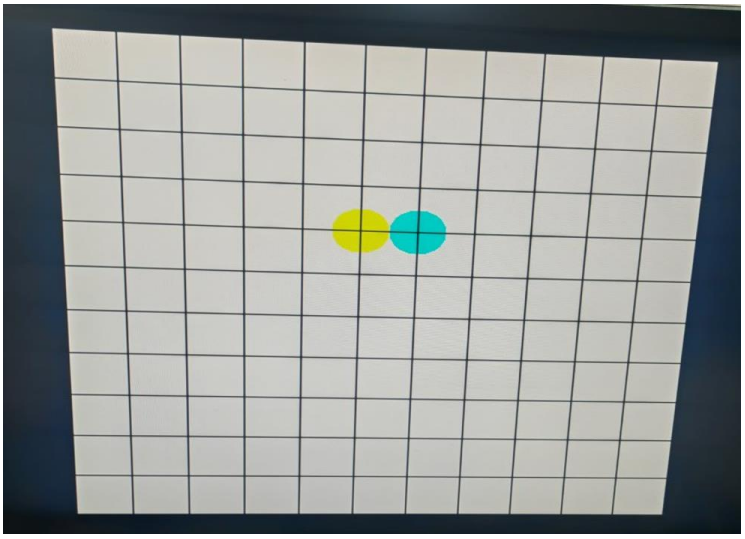
由于本项目中时序电路仿真并不方便，因此在实现 VGA 显示和 PS2 输入后，我们直接通过键盘上实操，观察现象（比如观察是否在连成 5 颗棋子时正确判断，结束游戏），来判断代码核心逻辑是否有误，因此仿真部分在本报告中不详细展示。以下展示一次实际游戏过程：

初始状态下，双方比分初始化为 0，显示 00:00：

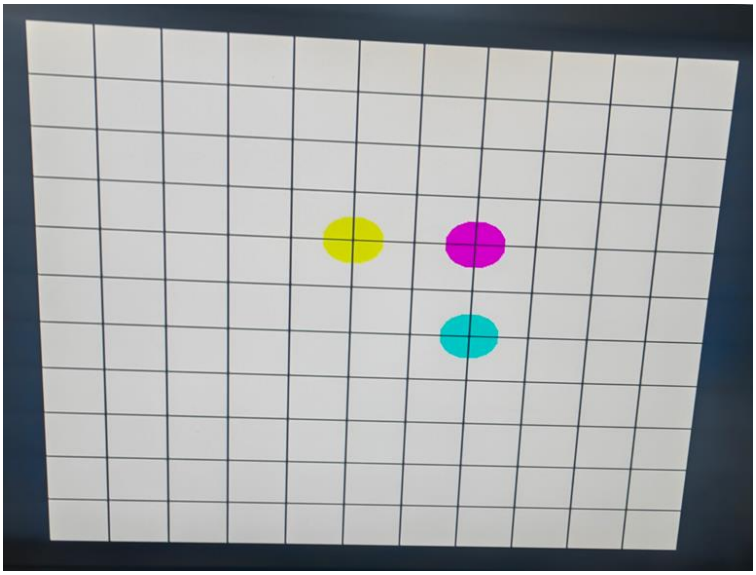


此时，游标初始化在左上角（1,1）位置处：

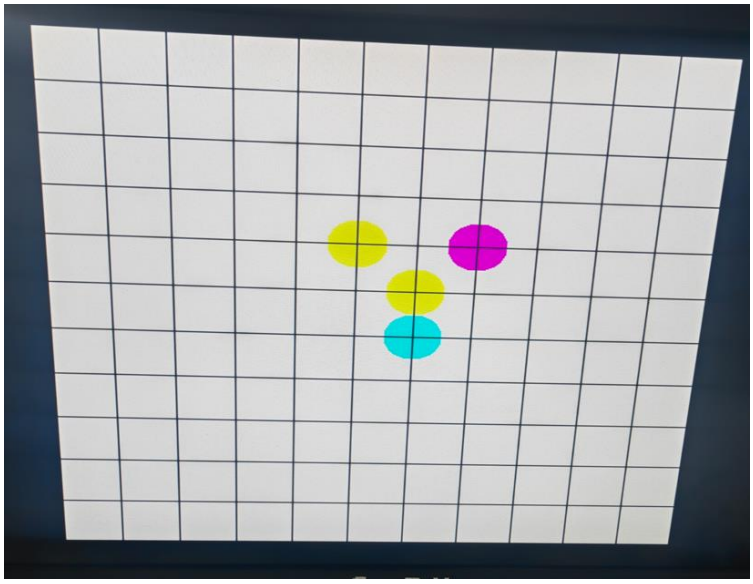
通过键盘上面的上下左右按键，可以移动游标，按击 enter 键，成功落下一颗黄棋子：



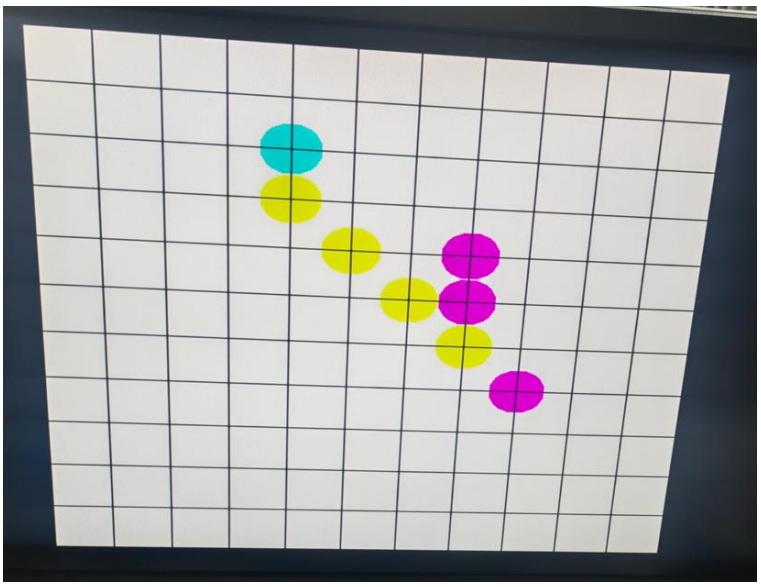
这时，玩家已经默认地切换到紫色一方，此时移动游标，再次下棋，发现为紫色棋子：

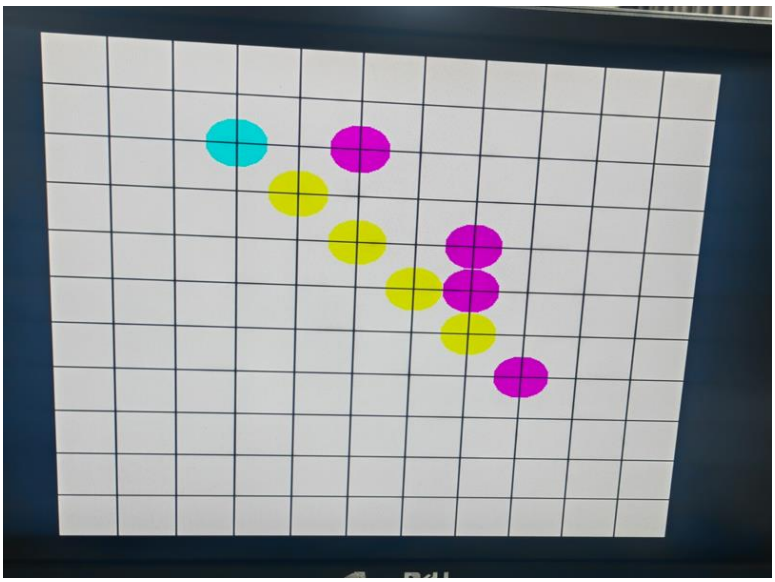


下完后，棋子切换为黄色一方，此时移动游标，再次下棋，发现为又为黄色棋子：

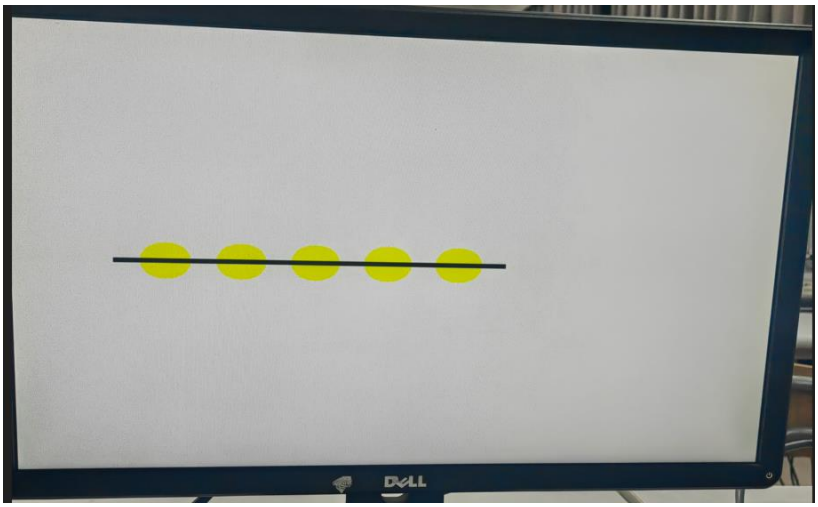


连续下下多颗棋子，如下图：

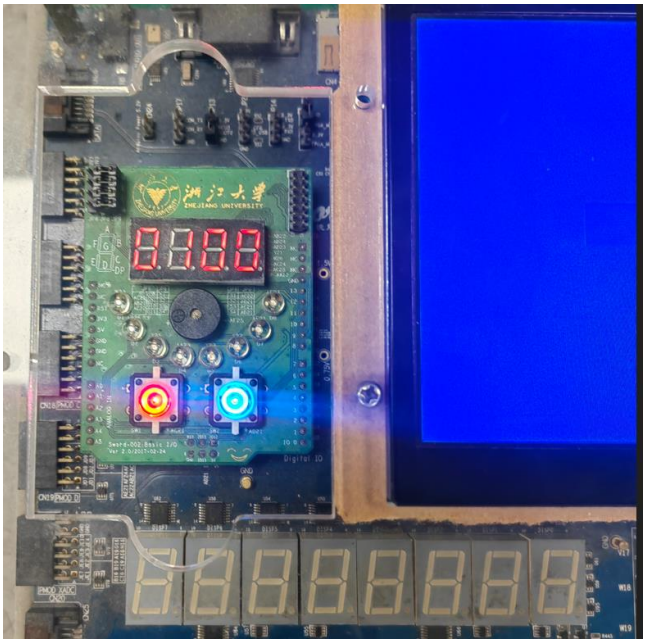




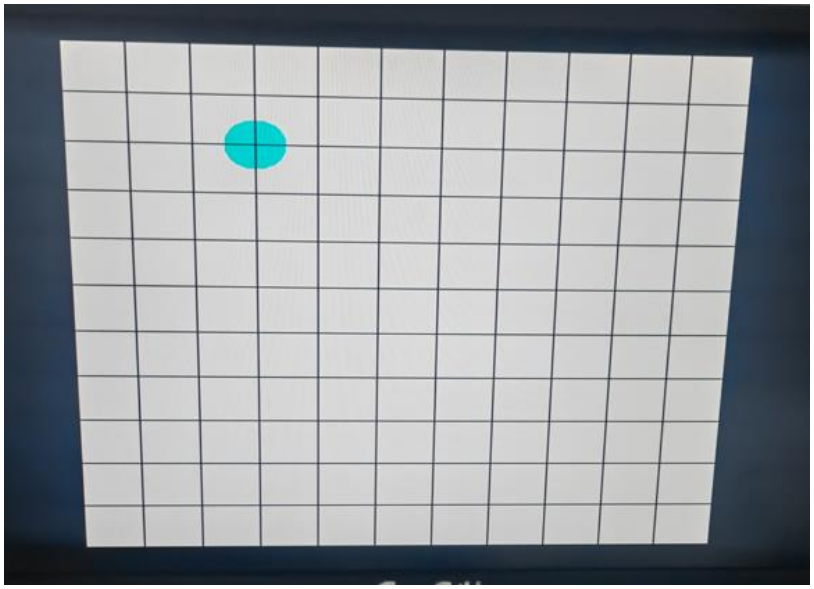
上图中按下 Enter 按键后，成功在游标位置落下一颗黄色棋子，这时候斜着已经连成了五颗棋子，系统判定黄色一方胜利，进入如下胜利界面（黄色）：



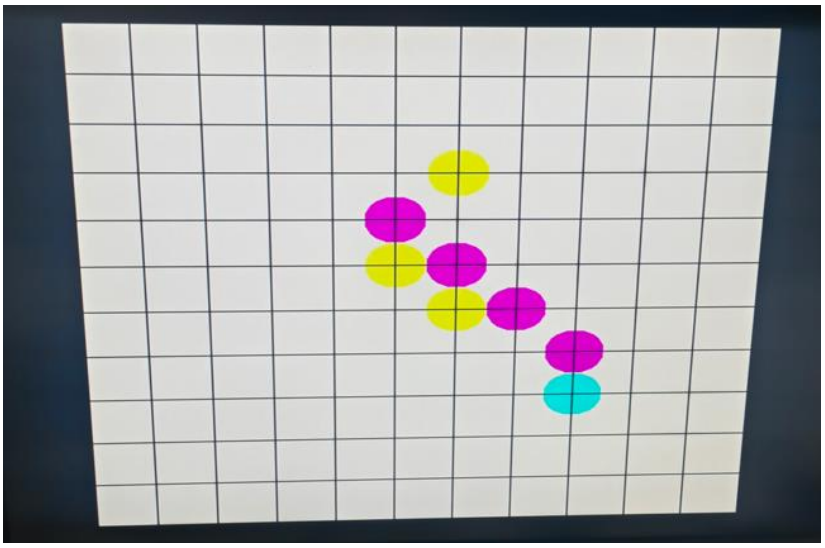
同时，黄色一方的比分加一，此时的比分变为 00:01：



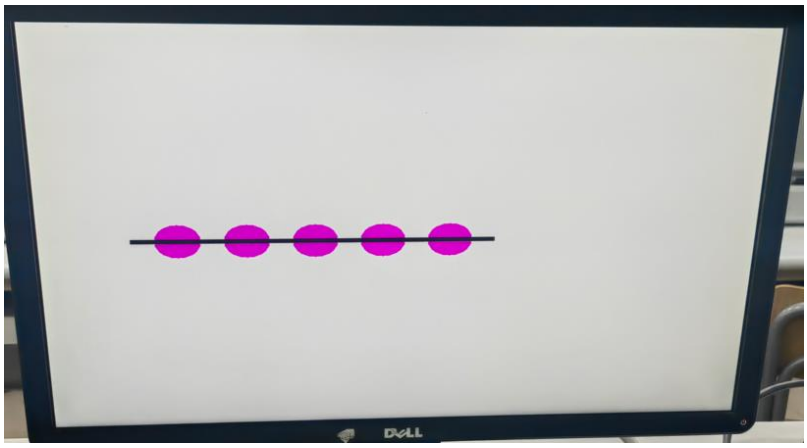
来回拨动一次 rst 开关，重新进入棋盘界面，且棋盘上所有棋子清空：



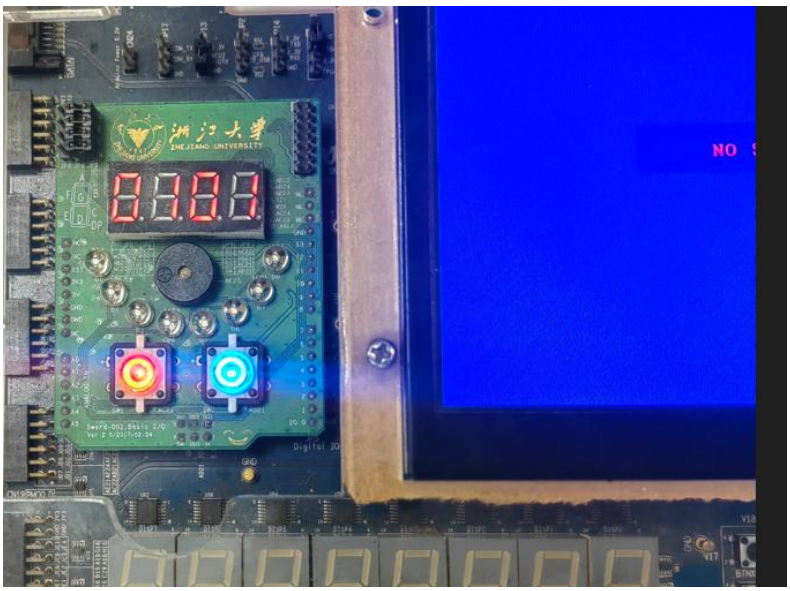
这时，连着下棋，过程略去，到以下界面：



这时，接着下棋，紫色连成 5 颗棋子后，进入紫色一方胜利界面：



同时，紫色一方的比分加一，此时的比分变为 01:01：



以上即为我们小组最终实现的双人对战五子棋游戏的简单展示，具体的细节可以看我们小组拍摄的视频。

五、小组分工

金艺轲（25%）	小组内各个成员都共同参与了 VGA 显示，PS2 键盘连接，五子棋逻辑模块的设计。
龙昱锦（25%）	
肖思勃（25%）	
盛铭（25%）	