计算机组成原理 实验报告

姓名：廖洲洲　学号：PB17081504　实验日期：2019-4-26

一、实验题目：

Lab4 存储器与显示控制器

二、实验目的：

利用简单双端口存储器 VRAM 存储 256*256 个像素的颜色信息，实现控制画笔在 800*600 分辨率的显示屏上进行涂画。

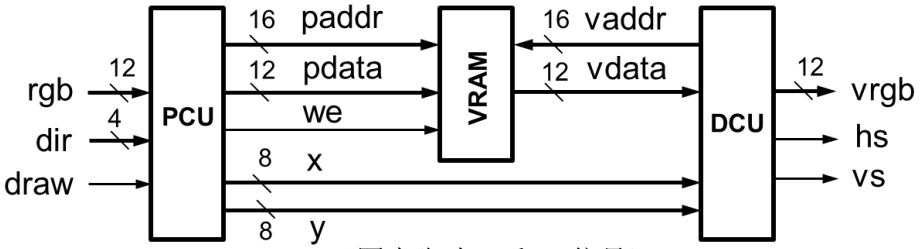
三、实验平台：

Vivado 2016.2

四、实验过程：

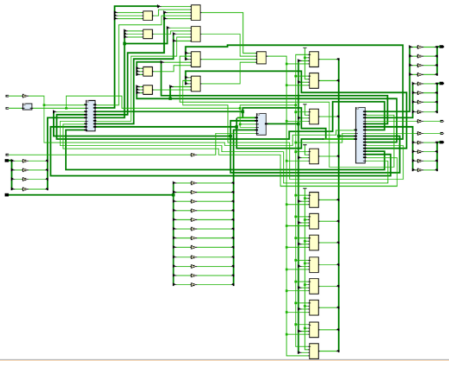
- 整体设计

（1）设计思路图：PCU 从外部读入写控制信号、颜色信息、画笔移动信号，然后将这些数据转换为在 VRAM 中相应的地址，将颜色存储在 VRAM 中，于是实现了写操作。DCU 模块从 VRAM 中读出相应的像素信息，将他们显示在显示屏上，于是实现了显示功能。

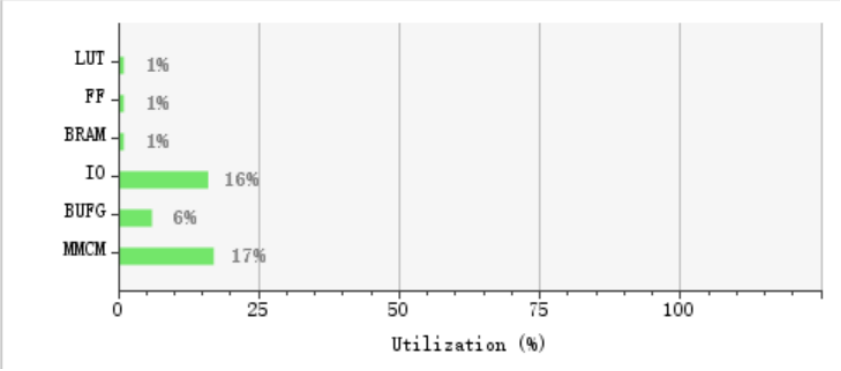


（图中省略clk和rst信号）

（2）原理图：



（3）资源使用情况



- DCU 显示模块设计

（1）设计思路：通过读端口取出 VRAM 信息并在显示屏显示，显示模式为 800x600 分辨率，刷新频率 72h，像素时钟频率 50mhz，同时在屏幕上显示十字光变。

（2）核心代码：行、场计数的实现和坐标的计算

```
always@(posedge clk_50mhz or posedge rst) begin
    if(rst) begin
          h_cnt<=0;
          v_cnt<=0;
    end
    else begin
          if(h_cnt==(H_CNT_MAX-1)) begin
                h_cnt<=0;
                if(v_cnt==(V_CNT_MAX-1)) v_cnt<=0;
                else v_cnt<=v_cnt+1;
          end
          else h_cnt<=h_cnt+1;
    end
end
//HS
assign HS=(h_cnt<H_HS)? 1'b0: 1'b1;
//VS
assign VS=(v_cnt<V_HS)? 1'b0: 1'b1;
assign disp_valid=((h_cnt>=456)&(h_cnt<712)&
(v_cnt>=201)&(v_cnt<457)) ? 1'b1:1'b0;
//display strips, one color per 32 lines, begin from v_cnt=31
assign {R,G,B}=disp_valid ? rgb:12'b0000_0000_0000;
assign  x=disp_valid ? h_cnt-456:0;
assign  y=disp_valid ? v_cnt-201:0;
```

- PCU 控制模块的设计

（1） 设计思路：从外部读入 4 位 dir 用于控制上下左右，当某一位按键被按下，对相应的横纵坐标进行加减，在根据 draw 使能信号决定是否在 RAM 进行写操作。进行了写操作即在存储器中存下了当

前点的颜色，相等于在显示器上进行了画操作。光标的实现是通过判断显示坐标与写坐标的差值来实现的，当显示坐标在以写坐标为中心的一个十字上时，让传给 DCU 显示模块中的数据不再是 RAM 中的值，而是 0（黑色），于是实现了光标的显示。

（2） 核心代码：

光标的移动，这种控制方式不但能使得光标上下左右移动，也能使得光标进行斜向移动。

```
if(dir[0] && ypos<255)        ypos<=ypos + 1;
if(dir[1] && xpos<255)        xpos<=xpos + 1;
if(dir[2] && ypos>=1)        ypos<=ypos - 1;
if(dir[3] && xpos>=1)        xpos<=xpos - 1;
```

十字光标的显示：

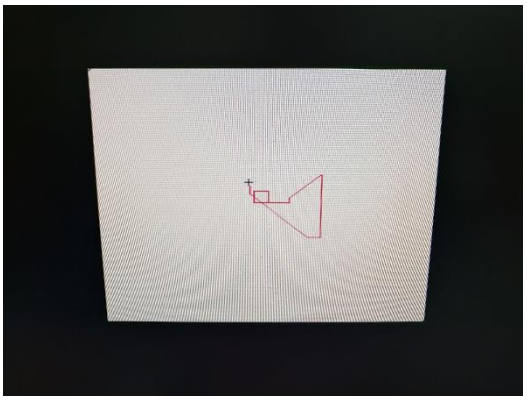
```
    if( ((x-xpos)>0 &(x-xpos)<9) | ((xpos-x)>0 &(xpos-x)<9) |
((y-ypos)>0&(y-ypos)<9) | ((ypos-y)>0 &(ypos-y)<9) ) begin
         rgb=12'b0000_0000_0000;
          end
    else
         rgb=ram_data;
```

- RAM 的实现：直接调用 IP 核，采用的是 Block memory 的简单双端口模式,数据宽度 12 位，深度 65536.

五、实验结果：

- 单独对 VGA 的调试，判断 VGA 模块的显示是否正确，通过多次的调试修改，使得 VGA 正确输出我想要的条纹

- 最后整个项目完成，可以通过按键在显示器上进行读写



六、心得体会：

- 通过本次实验，我学习并明白了 VGA 的显示原理，并通过这个显示原理设计并完成了显示的应用。
- 第一次调用了 RAM 的 IP 核，发现 Block Memory 比 Distributed Memory 更加适用于大数据存储，采用 Distributed Memory 来存储大数据可能会使得 IP 核的生成时间过长。

七、完整代码

```
module top_test(clk,rst,R,G,B,HS,VS,draw,wrgb,dir);
input clk,rst;
output [3:0] R;
output [3:0] G;
output [3:0] B;
output HS,VS;
input draw;
input [11:0] wrgb;
input [3:0] dir;
reg [11:0] rgb;
wire [7:0] x,y;//读
wire [7:0] xpos,ypos;//写
wire clk_50mhz;//时钟分频
wire [11:0] h_cnt,v_cnt;
//pcu_test pcu(rgb,x,y);
clk_wiz_0 clk1(.clk_in1(clk),.clk_out1(clk_50mhz));
vga_1 test1(clk_50mhz,rst,R,G,B,HS,VS,rgb,x,y,h_cnt,v_cnt);
```

```
wire [11:0] ram_data;
blk_mem_gen_0 blk(.addra({xpos,ypos}),.clka(clk_50mhz),.dina(wrgb),.ena(draw),
                  .addrb({x,y}),.clkb(clk_50mhz),.doutb(ram_data));
pcu_test pcu(clk_50mhz,rst,dir,xpos,ypos,h_cnt,y_cnt);
//光标的生成
always@(posedge clk_50mhz) begin
    if( ((x-xpos)>0 &(x-xpos)<9) | ((xpos-x)>0 &(xpos-x)<9) | ((y-ypos)>0&(y-
ypos)<9) | ((ypos-y)>0 &(ypos-y)<9) ) begin
         rgb=12'b0000_0000_0000;
          end
    else
         rgb=ram_data;
    end

endmodule


module vga_1(clk_50mhz,rst,R,G,B,HS,VS,rgb,x,y,h_cnt,v_cnt);
input clk_50mhz,rst;
output [3:0] R;
output [3:0] G;
output [3:0] B;
output HS,VS;
input [11:0] rgb;
//reg [11:0] rgb;
output [7:0] x,y;
output reg[11:0] h_cnt,v_cnt;
//reg [5:0] x,y;
//行参数
parameter H_CNT_MAX=1040;
parameter H_HS=120;
parameter H_BP=64;
parameter H_DISP=800;
parameter H_FP=56;
parameter H_Left=H_HS+H_BP;
parameter H_Right=H_HS+H_BP+H_DISP;
//列参数
parameter V_CNT_MAX=666;
parameter V_HS=6;
parameter V_BP=23;
parameter V_DISP=600;
parameter V_FP=37;
parameter V_Left=V_HS+V_BP;
parameter V_Right=V_HS+V_BP+V_DISP;


wire disp_valid;

//分频得到25MHz时钟
//  wire clk_50mhz;//时钟分频
//  clk_wiz_0 clk1(.clk_in1(clk),.clk_out1(clk_50mhz));
//行列扫描
```

```
always@(posedge clk_50mhz or posedge rst) begin
      if(rst) begin
            h_cnt<=0;
            v_cnt<=0;
      end
      else begin
            if(h_cnt==(H_CNT_MAX-1)) begin
                  h_cnt<=0;
                  if(v_cnt==(V_CNT_MAX-1)) v_cnt<=0;
                  else v_cnt<=v_cnt+1;
            end
            else h_cnt<=h_cnt+1;
      end
end
//HS
assign HS=(h_cnt<H_HS)? 1'b0: 1'b1;
//VS
assign VS=(v_cnt<V_HS)? 1'b0: 1'b1;
//valid region for display
//assign                          disp_valid=((h_cnt>=H_Left)&(h_cnt<H_Right)&
(v_cnt>=V_Left)&(v_cnt<V_Right)) ? 1'b1:1'b0;
//assign  disp_valid=((h_cnt>=272)&(h_cnt<528)&  (v_cnt>=172)&(v_cnt<428))  ?
1'b1:1'b0;
assign  disp_valid=((h_cnt>=456)&(h_cnt<712)&  (v_cnt>=201)&(v_cnt<457))  ?
1'b1:1'b0;
//display strips, one color per 32 lines, begin from v_cnt=31
assign {R,G,B}=disp_valid ? rgb:12'b0000_0000_0000;
assign  x=disp_valid ? h_cnt-456:0;
assign  y=disp_valid ? v_cnt-201:0;
//assign R=rgb[11:8];
//assign G=rgb[7:4];
//assign B=rgb[3:0];
//always@(posedge clk_50mhz) begin
//    R=4'b0000; G=4'b0000; B=4'b0000;   //black
//    if(disp_valid) begin
//          case(v_cnt[6:5])              //[6:5]=32lines
//          0: begin R=4'b1111; G=4'b0000; B=4'b0000; end    //red
//          1: begin R=4'b0000; G=4'b1111; B=4'b0000; end    //green
//          2: begin R=4'b0000; G=4'b0000; B=4'b1111; end    //blue
//          3: begin R=4'b1111; G=4'b1111; B=4'b1111; end    //white
//          default: begin R=4'b0000; G=4'b0000; B=4'b0000; end    //black
//          endcase
//    if(disp_valid) begin
//        x=h_cnt-552;
//        y=v_cnt[5:0];
       //y=v_cnt-297;
//        end
//    else    begin
//        x=0;
//        y=6'b111111;
//        end
//        if(y<16) begin
```

```
//           rgb=12'b1111_0000_0000;
//            end
//         else if(y<32)
//           rgb=12'b0000_1111_0000;
//         else if(y<48)
 //           rgb=12'b0000_0000_1111;
 //        else
 //           rgb=12'b1111_1111_1111;

//        R=rgb[11:8];
 //       G=rgb[7:4];
 //       B=rgb[3:0];

  // end



//always@(negedge clk_50mhz) begin
//    if(y<16) begin
//        rgb=12'b1111_0000_0000;
//        end
//     else if(y<32)
//        rgb=12'b0000_1111_0000;
//     else if(y<48)
//        rgb=12'b0000_0000_1111;
//     else
//        rgb=12'b1111_1111_1111;
 //    end
 //       if(disp_valid)
//           case(y[3:2])               //[6:5]=32lines
//         0: begin rgb=12'b1111_0000_1111; end    //red
//         1: begin rgb=12'b0000_1111_0000; end    //green
//         2: begin rgb=12'b0000_0000_1111; end    //blue
 //        3: begin  rgb=12'b1111_1111_1111;  end    //white
//         default: begin  rgb=12'b0000_1111_1111;  end    //black
//         endcase
//         else
//            rgb=12'b0000_0000_0000;
//         end
endmodule

module pcu_test(clk,rst,dir,xpos,ypos,h_cnt,v_cnt);
input clk,rst;
input [3:0] dir;
output reg [7:0] xpos,ypos;
input [11:0] h_cnt,v_cnt;
//output [3:0] R;
//output [3:0] G;
//output [3:0] B;
//output HS,VS;
//output reg [11:0] rgb;
```

```
//input [5:0] x,y;
//wire clk_50mhz;//时钟分频
//clk_wiz_0 clk1(.clk_in1(clk),.clk_out1(clk_50mhz));
//vga_test test1(clk_50mhz,rst,R,G,B,HS,VS,rgb,x,y);

//always@(*) begin
//    if(y<16) begin
//        rgb=12'b0000_1111_0000;
//        end
//    else if(y<32)
//        rgb=12'b0000_0000_0000;
 //   else if(y<48)
//        rgb=12'b0000_0000_1111;
//    else
//        rgb=12'b1111_1111_1111;
//     end

    always@(posedge clk,posedge rst ) begin
        if(rst) begin
            xpos <= 128;
            ypos <= 128;
//            cnt0<=0;
//            cnt1<=0;
//            cnt2<=0;
//            cnt3<=0;
//            we <= 0;
            end
        else if(h_cnt==0&v_cnt==0) begin
            if(dir[0] && ypos<255)       ypos<=ypos + 1;
            if(dir[1] && xpos<255)       xpos<=xpos + 1;
            if(dir[2] && ypos>=1)       ypos<=ypos - 1;
            if(dir[3] && xpos>=1)       xpos<=xpos - 1;
            end
        end
//        else if(dir[0]) begin
//            cnt0 <= cnt0+1;
//            if(cnt0==50000) begin
//                cnt0<=0;
//                if(ypos>=0&ypos<31)
//                    ypos<=ypos+1;
//                else
//                    ypos<=ypos;
  //              end
//            end
//        else if(dir[1]) begin
//            cnt1 <= cnt1+1;
//        if(cnt1==50000) begin
//            cnt1<=0;
//            if(xpos>=0&xpos<31)
//                xpos<=xpos+1;
//            else
```

```
//                xpos<=xpos;
 //           end
//        end
//        else if(dir[2]) begin
//            cnt2 <= cnt2+1;
//        if(cnt2==50000) begin
//            cnt2<=0;
//            if(ypos>=1&ypos<32)
 //               ypos<=ypos-1;
 //           else
 //               ypos<=ypos;
//            end
 //       end
//        else if(dir[3]) begin
 //            cnt3 = cnt3+1;
 //       if(cnt3==50000) begin
 //           cnt3=0;
 //           if(xpos>=1&xpos<32)
 //               xpos=xpos-1;
 //           else
 //               xpos<=xpos;
 //           end
//        end
 //       else  ;

 //   end


endmodule
```