实验报告

实验名称：Lab4_存储器与控制显示器

学生姓名：田宏宇 学号：PB17111573 得分：

实验日期： 2019 年 4 月 19 日

一、实验目的

1.学习例化并使用存储器IP（distributed memory）

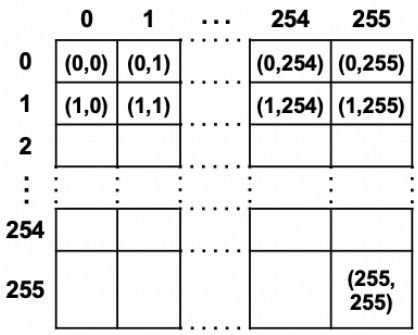
2.掌握VGA显示接口信号和显示原理

二、实验内容

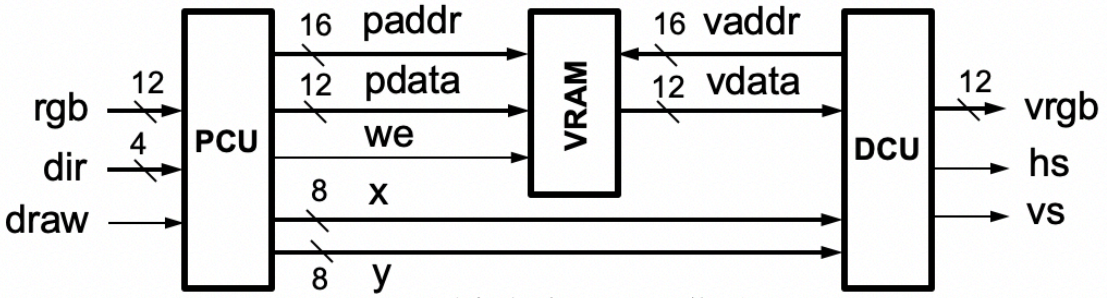
1. 控制画笔在800x600分辨率的显示器上随意涂画，画笔的颜色12位(红r绿g蓝b各4位)，绘画区域位于屏幕正中部，大小为256x256

- 画笔位置(x, y): x = y = 0 ~ 255, 复位时 (128, 128)
- 移动画笔(dir): 上/下/左/右按钮
- 画笔颜色(rgb): 12位开关设置
- 绘画状态(draw): 1-是, 0-否；

处于绘画状态时，移动画笔同时绘制颜色，否则仅移动画笔



- **VRAM**：视频存储器，存储**256x256**个像素的颜色信息，采用简单双端口存储器实现
  - **paddr, pdata, we:** 地址、数据、写使能，用于绘画的同步写端口
  - **vaddr, vdata:** 地址、数据，用于显示的异步读端口



（图中省略clk和rst信号）

- **PCU**：**Paint Control Unit**，绘画控制单元，修改**VRAM**中像素信息
  - 通过**12**个拨动开关设置像素颜色 **(rgb)**
  - 通过上/下/左/右**(dir)**按钮开关，移动画笔位置**(x, y)**
    - 直角移动：单一按钮按下一次，**x**或**y**增加或减小**1**
    - 对角移动：两按钮同时按下一次，**x**和**y**同时加或减**1**
    - 连续移动：按钮按下超过**t**秒后，等效为**s**速率的连续点击，直至松开（调试时确定合适的**t**和**s**取值）
  - 绘画 **(draw=1)** 时，依据 **rgb** 和 **(x, y)**，通过写端口 **(paddr, pdata, we)** 修改**VRAM**像素信息

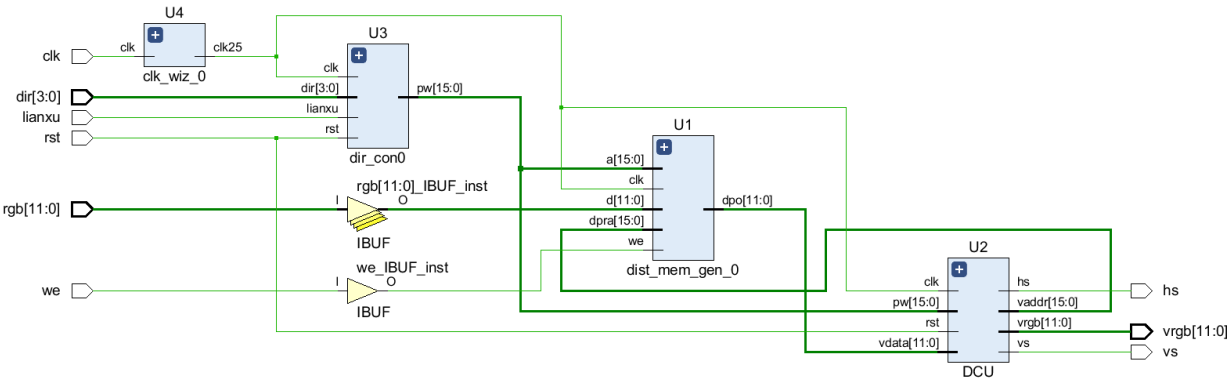
- **DCU：Display Control Unit，显示控制单元，显示VRAM中像素信息**
  - 通过读端口(vaddr, vdata)取出VRAM信息并显示
  - vrgb, hs, vs：显示器接口信号

  显示模式：分辨率800x600，刷新频率72Hz，像素时钟频率50MHz

  VRAM中的1个像素对应显示屏上1个像素
  - 在屏幕上显示十字光标，指示画笔当前位置 (x, y)

三、实验设计

主要功能分为四个模块（包括时钟），原理图如下：



U1：例化的distributed memory

```
dist_mem_gen_0 U1 (
                    .a(pwi),        // input wire [15 : 0] a
                    .d(rgb),        // input wire [11 : 0] d
                    .dpra(dp),  // input wire [15 : 0] dpra
                    .clk(clk25),    // input wire clk
                    .we(we),      // input wire we
                    .dpo(data)    // output wire [11 : 0] dpo
                  );
```

U2：DCU-显示控制单元

```
DCU U2 (
                    .vdata(data),
                    .vaddr(dp),
                    .vrgb(vrgb),
                    .hs(hs),
                    .vs(vs),
                    .clk(clk25),
                    .rst(rst),
                    .pw(pwi)
                  );
```

U3：通过方向控制存储器写地址

```
dir_con0 U3(
                  .dir(dir),
                  .clk(clk25),
                  .rst(rst),
                  .pw(pwi),
                  .lianxu(lianxu)
                       );
```

其中lianxu是控制画笔连续移动的选择信号。

U4: 时钟IP-将板载输入100Mhz转化为25Mhz

```
clk_wiz_0 U4 (
                  .clk(clk),
                  .clk25(clk25));
```

四、实验过程

1.为实现连续移动的功能，我添加了一个输入变量lianxu，引脚分配给SW14通过板上开关控制；

当lianxu=0时，画笔为单点移动，这时要将dir的电平信号转化为时钟沿信号

```
              //up
              always @ ( posedge clk  )
              up <= { up[1:0], dir[0]} ;
              assign pos_dir[0] = up[1] && ( ~up[2] );
               //down
              always @ ( posedge clk  )
              down <= { down[1:0], dir[1]} ;
              assign pos_dir[1] = down[1] && ( ~down[2] );
              //left
              always @ ( posedge clk  )
              left <= { left[1:0], dir[2]} ;
              assign pos_dir[2] = left[1] && ( ~left[2] );
              //right
```

```
always @ ( posedge clk  )
right <= { right[1:0], dir[3]} ;
assign pos_dir[3] = right[1] && ( ~right[2] );
```

当lianxu=1时，画笔为连续移动，每次时钟沿到来时判断dir来对画笔进行移动

```
always@(posedge clk1hz or posedge rst)
    begin
        if(rst)
          pw2<=16'd32895;//32768+127
        else
        case (dir)
        4'b0001:
        if(pw>=16'd256)//up
          pw2<=pw-16'd256;
        4'b0010:
        if(pw<(16'd65280))//down
          pw2<=pw+16'd256;
        4'b0100:
        if(pw%16'd256!=0)//left
          pw2<=pw-16'd1;
        4'b1000:
        if(pw%16'd256!=16'd255)//right
          pw2<=pw+16'd1;
        default: pw2<=pw;
        endcase
    end
```

2.实现画笔为十字

利用条件x将画笔位置pw以及其左上左下右上右下显示为黑色，形成”x“形状的画笔。

x条件

```
assign x=(pw==vaddr || pw-255==vaddr || pw-257==vaddr || pw+255==vaddr ||
pw+257==vaddr)?1:0;
```

3.第一次到显示器试验时，显示输入信号频率不符无法输出，后来听同学的建议，将各个定时参数改为分辨率为640x480的标准。

分辨率	像元频率（兆赫兹）	每行像元数	行同步（消隐）脉冲开始时钟周期	行同步（消隐）脉冲结束时钟周期	每行时钟周期数	每帧图象行数	帧同步脉冲开始扫描行数	帧同步脉冲结束扫描行数	每帧扫描行数
640x480@60	25.2	640	656	752	800	480	490	492	525

为方便赋值和扫描时各条件判断，将各参数设置为常量

```
parameter
                    C_H_SYNC_PULSE      =   96  ,
                    C_H_BACK_PORCH      =   48  ,
                    C_H_ACTIVE_TIME     =   640 ,
                    C_H_FRONT_PORCH     =   16  ,
                    C_H_LINE_PERIOD     =   800 ;


parameter
                  C_V_SYNC_PULSE      =   2   ,
                  C_V_BACK_PORCH      =   33  ,
                  C_V_ACTIVE_TIME     =   480 ,
                  C_V_FRONT_PORCH     =   10  ,
                  C_V_FRAME_PERIOD    =   525 ;
```

4.改变参数后再次测试，发现在画笔连续移动和非连续移动切换后，会出现画出的线颜色不对和画出平行线等错误，经过分析，我觉得是对pw进行操作时，以时钟沿控制数据pw1，pw2，pw的改变会有一定的门延迟存在，导致数据写入有错误，因此我将原本reg型pw时钟沿赋值改为wire型以assign方式赋值，解决了问题。

改后wire型

```
            wire [15:0]pw11;
            assign pw11=pw1;
            wire [15:0]pw22;
            assign pw22=pw2;
            assign pw=(lianxu)?pw22:pw11;
```

改前reg型

```
          always@(posedge clk or posedge rst)
            begin
            if(rst)
              pw<=32768+127;
            else
             case(lianxu)
             1:pw<=pw2;
             0:pw<=pw1;
             default:pw<=pw;
             endcase
            end
```

5.原本打算用1Hz的时钟控制画笔连续，即1画格每秒，而后发现画笔移动太慢，将时钟改为10Hz

```
always@(posedge clk)
              begin
                //if(count==25'd24999999)     //1Hz
                if(count==25'd2499999)        //10Hz
                    begin
                  count<=0;
                  clk1hz<=~clk1hz;
                  end
                else count<=count+1;
              end
```

五、实验总结

在这次实验中，我实现了利用VGA扫描技术将存储器中的256*256个数据显示到显示器上，即实现彩色画板的功能，加深了我对功能模块化和参数化设计的理解，提高了我的verilog编程能力，并掌握了VGA显示原理。

附：代码

```
      module vga(
                  input [11:0]rgb,
                  input [3:0]dir,
                  input we,
                  input clk,
                  input rst,
                  input lianxu,
                  output [11:0]vrgb,
                  output hs,
                  output vs
                );
                wire [15 : 0] dp;
                wire [11 : 0] data;
                wire [15 : 0] pwi;
                wire clk25;

                dist_mem_gen_0 U1 (
                    .a(pwi),        // input wire [15 : 0] a
                    .d(rgb),        // input wire [11 : 0] d
                    .dpra(dp),  // input wire [15 : 0] dpra
                    .clk(clk25),    // input wire clk
                    .we(we),      // input wire we
                    .dpo(data)    // output wire [11 : 0] dpo
                  );
                  DCU U2 (
                    .vdata(data),
```

```
            .vaddr(dp),
            .vrgb(vrgb),
            .hs(hs),
            .vs(vs),
            .clk(clk25),
            .rst(rst),
            .pw(pwi)
          );
          dir_con0 U3(
          .dir(dir),
              .clk(clk25),
              .rst(rst),
                .pw(pwi),
          .lianxu(lianxu)
                );
         clk_wiz_0 U4 (
          .clk(clk),
          .clk25(clk25)
         );

endmodule

module DCU(
       input [11:0]vdata,
       output reg [15:0]vaddr,
       output reg [11:0]vrgb,
       output hs,
       output vs,
       input clk,
       input rst,
       input [15:0] pw
    );
    wire draw;
    wire active;
    wire x;

    parameter
        C_H_SYNC_PULSE      =   96  ,
        C_H_BACK_PORCH      =   48  ,
        C_H_ACTIVE_TIME     =   640 ,
        C_H_FRONT_PORCH     =   16  ,
        C_H_LINE_PERIOD     =   800 ;

    parameter
        C_V_SYNC_PULSE      =   2   ,
        C_V_BACK_PORCH      =   33  ,
        C_V_ACTIVE_TIME     =   480 ,
        C_V_FRONT_PORCH     =   10  ,
        C_V_FRAME_PERIOD    =   525 ;
```

```
                parameter UP_BOUND = C_V_SYNC_PULSE + C_V_BACK_PORCH ;
                parameter DOWN_BOUND = C_V_FRAME_PERIOD - C_V_FRONT_PORCH;
                parameter LEFT_BOUND = C_H_SYNC_PULSE + C_H_BACK_PORCH;
                parameter RIGHT_BOUND = C_H_LINE_PERIOD - C_H_FRONT_PORCH;
                parameter UP = 100;
                parameter DOWN = 356;
                parameter LEFT = 300;
                parameter RIGHT = 556;

                  wire pclk;
                  assign pclk=clk;
                  reg [12:0] hcount, vcount;
    assign draw=(vcount>=UP && vcount<DOWN &&hcount>=LEFT && hcount<RIGHT)?
1:0;
    assign active=(vcount>=UP_BOUND && vcount<DOWN_BOUND&&hcount>=LEFT_BOUND
&&
                   hcount<RIGHT_BOUND)?1:0;
    assign x=(pw==vaddr || pw-255==vaddr || pw-257==vaddr || pw+255==vaddr ||
pw+257==vaddr)?1:0;
    assign vs = (vcount < C_V_SYNC_PULSE )? 0 : 1;
    assign hs = (hcount < C_H_SYNC_PULSE )? 0 : 1;
                  always @ (posedge pclk or posedge rst)
                  begin
                    if (rst)
                    begin
                      vcount <= 0;
                      vaddr <= 0;
                    end
                    else if (hcount == C_H_LINE_PERIOD -1)
                    begin
                      hcount <= 0;
                      if (vcount == C_V_FRAME_PERIOD -1)
                        begin
                               vcount <= 0;
                               vaddr <= 0;
                           end
                      else
                        vcount <= vcount+1;
                    end
                    else
                    begin
                      hcount <= hcount+1;
                      vcount <= vcount;
                      if(draw)
                      vaddr<=vaddr+1;
                    end
                  end

                  always @ ()
```

```
            begin
              if(active)
              begin
              if(draw)
                   if(x)
                       vrgb<=0;
                   else
                    vrgb<=vdata;
              else vrgb<=0;
              end
              else vrgb<=0;
            end
endmodule


module dir_con0(
      input [3:0]dir,
      input clk,
      input rst,
      output  [15:0]pw,
      input lianxu
    );
      reg [2:0] up;
      reg [2:0] down;
      reg [2:0] left;
      reg [2:0] right;
      wire [3:0]pos_dir;
      reg clk1hz;
      reg [24:0] count;
      reg [15:0]pw1;
      reg [15:0]pw2;
      always@(posedge clk)
      begin
        //if(count==25'd24999999)     //1Hz
        if(count==25'd2499999)        //10Hz
            begin
          count<=0;
          clk1hz<=~clk1hz;
          end
        else count<=count+1;
      end
      //up
      always @ ( posedge clk  )
      up <= { up[1:0], dir[0]} ;
      assign pos_dir[0] = up[1] && ( ~up[2] );
       //down
      always @ ( posedge clk  )
      down <= { down[1:0], dir[1]} ;
      assign pos_dir[1] = down[1] && ( ~down[2] );
```

```
  //left
  always @ ( posedge clk  )
  left <= { left[1:0], dir[2]} ;
  assign pos_dir[2] = left[1] && ( ~left[2] );
  //right
  always @ ( posedge clk  )
  right <= { right[1:0], dir[3]} ;
  assign pos_dir[3] = right[1] && ( ~right[2] );


always@(posedge clk or posedge rst)
begin
    if(rst)
    begin
      pw1<=16'd32895;
    end
    else
    case(pos_dir)
    4'b0001:
    if(pw>=16'd256)//up
      pw1<=pw-16'd256;
    4'b0010:
    if(pw<(16'd65280))//down
      pw1<=pw+16'd256;
    4'b0100:
    if(pw%16'd256!=0)//left
      pw1<=pw-16'd1;
    4'b1000:
    if(pw%16'd256!=16'd255)//right
      pw1<=pw+16'd1;
    default: pw1<=pw;
    endcase
end

always@(posedge clk1hz or posedge rst)
begin
    if(rst)
      pw2<=16'd32895;//32768+127
    else
    case (dir)
    4'b0001:
    if(pw>=16'd256)//up
      pw2<=pw-16'd256;
    4'b0010:
    if(pw<(16'd65280))//down
      pw2<=pw+16'd256;
    4'b0100:
    if(pw%16'd256!=0)//left
      pw2<=pw-16'd1;
```

```
        4'b1000:
        if(pw%16'd256!=16'd255)//right
          pw2<=pw+16'd1;
        default: pw2<=pw;
        endcase
    end
    wire [15:0]pw11;
    assign pw11=pw1;
    wire [15:0]pw22;
    assign pw22=pw2;
    assign pw=(lianxu)?pw22:pw11;
    /*always@(posedge clk or posedge rst)
    begin
    if(rst)
      pw<=32768+127;
    else
     case(lianxu)
     1:pw<=pw2;
     0:pw<=pw1;
     default:pw<=pw;
     endcase
    end*/
    endmodule
```