实验九

VGA 接口控制器

实验报告　　日期：2020 年 11 月 20 日

姓名： 朱嘉琦

学号： 191220185

班级： 数电实验一班

邮箱： 1477194584@qq.com

实验九报告—— VGA 接口控制器实现

191220185 朱嘉琦

一、实验目的

VGA 接口是 IBM 制定的一种视频数据的传输标准，是电脑显示器最典型的接口。本实验的目的是学习 VGA 接口原理，学习 VGA 接口控制器的设计方法。

二、实验原理

VGA 接口的外观和引脚功能

图 9-1: VGA 接口形状示意图

VGA（Video Graphics Array）接口，即视频图形阵列。VGA 接口最初是用于连接 CRT 显示器的接口，CRT 显示器因为设计制造上的原因，只能接受模拟信号输入，这就需要显卡能输出模拟信号，VGA 接口就是显卡上输出模拟信号的接口，在传统的 CRT 显示器中，使用的都是 VGA 接口。VGA 接口是 15 针/孔的梯形插头，分成 3 排，每排 5 个，如图9-1所示：

VGA 接口的接口信号主要有 5 个：R（Red）、G（Green）、B（Blue）、HS（Horizontal Synchronization）和VS（Vertical Synchronization），即红、绿、蓝、水平同步和垂直同步（也称行同步和帧同步）。

VGA 的工作原理

图像的显示是以像素（点）为单位，显示器的分辨率是指屏幕每行有多少个像素及每帧有多少行，标准的 VGA 分辨率是 640×480，也有更高的分辨率， 如 1024×768、1280×1024、1920×1200 等。从人眼的视觉效果考虑，屏幕刷新的频率（每秒钟显示的帧数）应该大于 24，这样屏幕看起来才不会闪烁，VGA 显示器一般的刷新频率是 60HZ。 每一帧图像的显示都是从屏幕的左上角开始一行一行进行的，行同步信号是一个负脉冲，行同步信号有效后，由 RGB 端送出当前行显示的各像素点的 RGB 电压值，当一帧显示结束后，由帧同步信号送出一个负脉冲，重新开始从屏幕的左上端开始显示下一帧图像，如图 9-2所示。

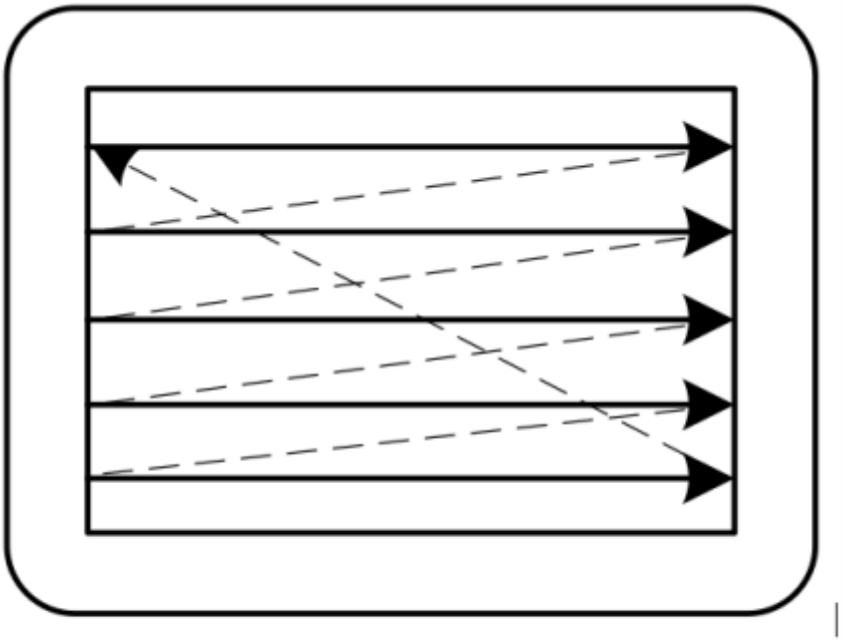


图 9-2: 显示器扫描示意图

在标准的 640 × 480 的 VGA 上有效地显示一行信号需要 96+48+640+16=800 个像素点的时间，其中行同步负脉冲宽度为 96 个像素点时间，行消隐后沿需要 48 个像素点时间，然后每行显示 640 个像素点，最后行消隐前沿需要 16 个像素点的时间。所以一行中显示像素的时间为 640 个像素点时间，一行消隐时间为 160 个像素点时间。

在标准的 640 × 480 的 VGA 上有效显示一帧图像需要 2+33+480+10=525 行 时间，其中场同步负脉冲宽度为 2 个行显示时间，场消隐后沿需要 33 个行显示 时间，然后每场显示 480 行，场消隐前沿需要 10 个行显示时间，一帧显示时间 为 525 行显示时间，一帧消隐时间为 45 行显示时间。

因此，在 640 × 480 的 VGA 上的一幅图像需要 525 × 800 =420k 个像素点的时间。而每秒扫描 60 帧共需要约 25M 个像素点的时间。

DE10-Standard 开发板上的 VGA 接口

DE10-Standard 开发板上使用了一块 VGA DAC ADV7123 芯片来实现 VGA 功能。该芯片完成 FPGA 数字信号到 VGA 模拟信号的转换，具体连接方式如图 9-4所示。 开发板和 ADV7123 芯片之间的接口引脚包括 3 组 8bit 的颜色信号 VGA_R[7:0], VGA_G[7:0], VGA_B[7:0]，行同步信号 VGA_HS，帧同步信号 VGA_VS，VGA 时钟信号 VGA_CLK，VGA 同步（低有效）VGA_SYNC_N, 和 VGA 消隐信号（低有效）VGA_BLANK_N。

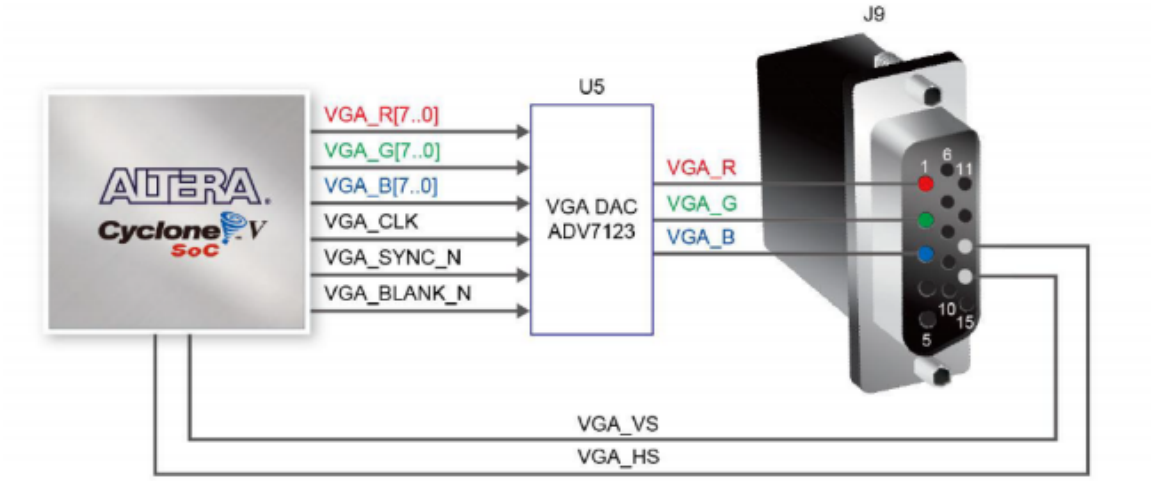


图 9-4: DE10-Standard 的 VGA 连接示意图

三、实验环境/器材

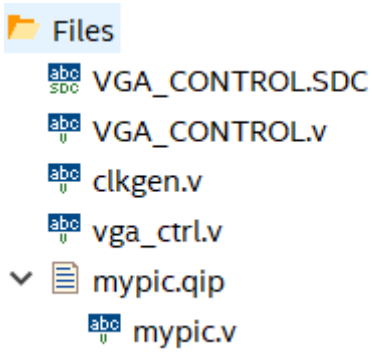
实验环境是Quarters 17.1 Lite，实验器材是DE10 开发板

四、程序代码或流程图

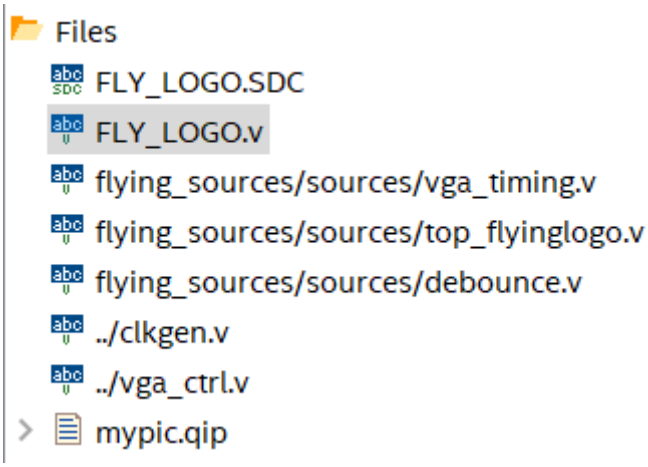
1. 工程文件

本次实验涉及到多个模块，而且有两个工程文件

VGA_CONTROL



FLY_LOGO



2. VGA_CONTROL代码部分

1） 分频器clkgen.v

```
module clkgen(
    input clkin,
    input rst,
    input clken,
    output reg clkout
    );
    parameter clk_freq=1000;
    parameter countlimit=50000000/2/clk_freq; // count

    reg[31:0] clkcount;
    always @ (posedge clkin)
        if(rst)
            begin
                clkcount=0;
                clkout=1'b0;
            end
        else
            begin
                if(clken)
                    begin
                        clkcount=clkcount+1;
                        if(clkcount>=countlimit)
                            begin
                                clkcount=32'd0;
                                clkout=~clkout;
                            end
                        else
                            clkout=clkout;
                    end
                else
                    begin
                        clkcount=clkcount;
                        clkout=clkout;
                    end
            end
endmodule
```

2) VGA控制信号 vga_ctrl

```
module vga_ctrl(
    input pclk, //25MHz时钟
    input reset, //置位
    input [11:0] vga_data, // 上层模块提供的VGA颜色数据
    output [9:0] h_addr, // 提供给上层模块的当前扫描像素点坐标
    output [9:0] v_addr,
    output hsync, // 行同步和列同步信号
    output vsync,
    output valid, //消隐
    output [3:0] vga_r, // RGB
    output [3:0] vga_g,
    output [3:0] vga_b
    );

    //640x480
    parameter h_frontporch = 96;
    parameter h_active = 144;
    parameter h_backporch = 784;
    parameter h_total = 800;

    parameter v_frontporch = 2;
    parameter v_active = 35;
    parameter v_backporch = 515;
    parameter v_total = 525;

    // 像素计数值
    reg [9:0] x_cnt;
    reg [9:0] y_cnt;
    wire h_valid;
    wire v_valid;

    always @(posedge reset or posedge pclk) // 行像素计数
        if (reset == 1'b1)
            x_cnt <= 1;
        else
            begin
                if (x_cnt == h_total)
                    x_cnt <= 1;
                else
                    x_cnt <= x_cnt + 10'd1;
            end

    always @(posedge pclk) // 列像素计数
    if (reset == 1'b1)
        y_cnt <= 1;
    else
        begin
            if (y_cnt == v_total & x_cnt == h_total)
                y_cnt <= 1;
            else if (x_cnt == h_total)
                y_cnt <= y_cnt + 10'd1;
        end

    // 生成同步信号
    assign hsync = (x_cnt > h_frontporch);
    assign vsync = (y_cnt > v_frontporch);
    // 生成消隐信号
    assign h_valid = (x_cnt > h_active) & (x_cnt <= h_backporch);
```

```
    assign v_valid = (y_cnt > v_active) & (y_cnt <= v_backporch);
    assign valid = h_valid & v_valid;
    // 计算当前有效像素坐标
    assign h_addr = h_valid ? (x_cnt - 10'd145) : {10{1'b0}};
    assign v_addr = v_valid ? (y_cnt - 10'd36) : {10{1'b0}};
    // 设置输出的颜色值
    assign vga_r = vga_data[11:8];
    assign vga_g = vga_data[7:4];
    assign vga_b = vga_data[3:0];

endmodule
```

3) VGA_CONTROL.v

```
//=======================================================
//  This code is generated by Terasic System Builder
//=======================================================

module VGA_CONTROL(

    ////////////// CLOCK //////////
    input                       CLOCK2_50,
    input                       CLOCK3_50,
    input                       CLOCK4_50,
    input                       CLOCK_50,

    ////////////// KEY //////////
    input             [3:0]     KEY,

    ////////////// SW //////////
    input             [9:0]     SW,

    ////////////// LED //////////
    output            [9:0]     LEDR,

    ////////////// Seg7 //////////
    output            [6:0]     HEX0,
    output            [6:0]     HEX1,
    output            [6:0]     HEX2,
    output            [6:0]     HEX3,
    output            [6:0]     HEX4,
    output            [6:0]     HEX5,

    ////////////// VGA //////////
    output                      VGA_BLANK_N,
    output            [7:0]     VGA_B,
    output                      VGA_CLK,
    output            [7:0]     VGA_G,
    output                      VGA_HS,
    output            [7:0]     VGA_R,
    output                      VGA_SYNC_N,
    output                      VGA_VS
);
```

```
//=============================================================
//  REG/WIRE declarations
//=============================================================
    assign VGA_SYNC_N=0;
    wire[9:0] h_addr;
    wire[9:0] v_addr;
    wire[11:0] vga_data;
//=============================================================
//  Structural coding
//=============================================================

    clkgen #(25000000) my_vgaclk(CLOCK_50,1'b0,1'b1,VGA_CLK);

    mypic r1(
    .clock(VGA_CLK),
    .data(),
    .rdaddress({h_addr[9:0],v_addr[8:0]}),
    .rden(1'b1),
    .wraddress(),
    .wren(1'b0),
    .q(vga_data));

    vga_ctrl(
    .pclk(VGA_CLK), //25MHz时钟
    .reset(SW[0]), //置位
    .vga_data(vga_data), // 上层模块提供的VGA颜色数据
    .h_addr(h_addr), // 提供给上层模块的当前扫描像素点坐标
    .v_addr(v_addr),
    .hsync(VGA_HS), // 行同步和列同步信号
    .vsync(VGA_VS),
    .valid(VGA_BLANK_N), //消隐
    .vga_r(VGA_R[7:4]), // RGB
    .vga_g(VGA_G[7:4]),
    .vga_b(VGA_B[7:4])
    );



endmodule
```

3. FLY_LOGO代码部分

1） top_flying.v

```
`timescale 1 ns / 1 ns


module top_flyinglogo(clk, rst, hsync, vsync, vga_r, vga_g, vga_b, pclk, valid);
   input           clk;
   input           rst;

   output          hsync;
   output          vsync;
   output [3:0]    vga_r;
   output [3:0]    vga_g;
```

```
    output [3:0]    vga_b;


    output            pclk;
    output            valid;
    wire [9:0]      h_cnt;
    wire [9:0]      v_cnt;
    reg [11:0]       vga_data;

    reg [18:0]      rom_addr;
    wire [11:0]      douta;

    wire            logo_area;
    reg [9:0]       logo_x;
    reg [9:0]       logo_y;
    parameter [9:0] logo_length = 200;
    parameter [9:0] logo_hight  = 200;

    reg [7:0]       speed_cnt;
    wire            speed_ctrl;

    reg [3:0]       flag_edge;

     clkgen #(25000000) my_vgaclk(clk,1'b0,1'b1,pclk);


     mypic r1(
     .clock(pclk),
     .data(),
     .rdaddress({h_cnt[9:0],v_cnt[8:0]}),
     .rden(1'b1),
     .wraddress(),
     .wren(1'b0),
     .q(douta));



     vga_ctrl(
     .pclk(pclk), //25MHz时钟
     .reset(1'b0), //置位
     .vga_data(douta), // 上层模块提供的VGA颜色数据
     .h_addr(h_cnt), // 提供给上层模块的当前扫描像素点坐标
     .v_addr(v_cnt),
     .hsync(hsync), // 行同步和列同步信号
     .vsync(vsync),
     .valid(valid), //消隐
     .vga_r(vga_r), // RGB
     .vga_g(vga_g),
     .vga_b(vga_b)
     );



    assign logo_area = ((v_cnt >= logo_y) & (v_cnt <= logo_y + logo_hight - 1) &
(h_cnt >= logo_x) & (h_cnt <= logo_x + logo_length - 1)) ? 1'b1 :
                       1'b0;
```

```
always @(posedge pclk)
begin: logo_display
   if (rst == 1'b1)
      vga_data <= 12'b000000000000;
   else
   begin
      if (valid == 1'b1)
      begin
         if (logo_area == 1'b1)
         begin
            rom_addr <= rom_addr + 19'b0000000000000000001;
            vga_data <= douta;
         end
         else
         begin
            rom_addr <= rom_addr;
            vga_data <= 12'b000000000000;
         end
      end
      else
      begin
         vga_data <= 12'b111111111111;
         if (v_cnt == 0)
            rom_addr <= 19'b0000000000000000000;
      end
   end
end

assign vga_r = vga_data[11:8];
assign vga_g = vga_data[7:4];
assign vga_b = vga_data[3:0];


always @(posedge pclk)
begin: speed_control
   if (rst == 1'b1)
      speed_cnt <= 8'h00;
   else
   begin
      if ((v_cnt[5] == 1'b1) & (h_cnt == 1))
         speed_cnt <= speed_cnt + 8'h01;
   end
end


debounce u3(.sig_in(speed_cnt[5]), .clk(pclk), .sig_out(speed_ctrl));


always @(posedge pclk)
begin: logo_move

   reg [1:0]       flag_add_sub;

   if (rst == 1'b1)
   begin
      flag_add_sub = 2'b01;

      logo_x <= 10'b0110101110;
```

```
            logo_y <= 10'b0000110010;
        end
        else
        begin

            if (speed_ctrl == 1'b1)
            begin
                if (logo_x == 1)
                begin
                    if (logo_y == 1)
                    begin
                        flag_edge <= 4'h1;
                        flag_add_sub = 2'b00;
                    end
                    else if (logo_y == 480 - logo_hight)
                    begin
                        flag_edge <= 4'h2;
                        flag_add_sub = 2'b01;
                    end
                    else
                    begin
                        flag_edge <= 4'h3;
                        flag_add_sub[1] = (~flag_add_sub[1]);
                    end
                end

                else if (logo_x == 640 - logo_length)
                begin
                    if (logo_y == 1)
                    begin
                        flag_edge <= 4'h4;
                        flag_add_sub = 2'b10;
                    end
                    else if (logo_y == 480 - logo_hight)
                    begin
                        flag_edge <= 4'h5;
                        flag_add_sub = 2'b11;
                    end
                    else
                    begin
                        flag_edge <= 4'h6;
                        flag_add_sub[1] = (~flag_add_sub[1]);
                    end
                end

                else if (logo_y == 1)
                begin
                    flag_edge <= 4'h7;
                    flag_add_sub[0] = (~flag_add_sub[0]);
                end
                else if (logo_y == 480 - logo_hight)
                begin
                    flag_edge <= 4'h8;
                    flag_add_sub[0] = (~flag_add_sub[0]);
                end
                else
                begin
                    flag_edge <= 4'h9;
```

```
            flag_add_sub = flag_add_sub;
        end

        case (flag_add_sub)
            2'b00 :
                begin
                    logo_x <= logo_x + 10'b0000000001;
                    logo_y <= logo_y + 10'b0000000001;
                end
            2'b01 :
                begin
                    logo_x <= logo_x + 10'b0000000001;
                    logo_y <= logo_y - 10'b0000000001;
                end
            2'b10 :
                begin
                    logo_x <= logo_x - 10'b0000000001;
                    logo_y <= logo_y + 10'b0000000001;
                end
            2'b11 :
                begin
                    logo_x <= logo_x - 10'b0000000001;
                    logo_y <= logo_y - 10'b0000000001;
                end
            default :
                begin
                    logo_x <= logo_x + 10'b0000000001;
                    logo_y <= logo_y + 10'b0000000001;
                end
        endcase
      end

    end
  end

endmodule
```

2) FLY_LOGO.v

```
//=======================================================
//  This code is generated by Terasic System Builder
//=======================================================

module FLY_LOGO(

    ////////////// CLOCK //////////
    input                       CLOCK2_50,
    input                       CLOCK3_50,
    input                       CLOCK4_50,
    input                       CLOCK_50,

    ////////////// KEY //////////
    input           [3:0]       KEY,

    ////////////// SW //////////
    input           [9:0]       SW,
```

```
    ////////////// LED //////////
    output           [9:0]      LEDR,

    ////////////// Seg7 //////////
    output           [6:0]      HEX0,
    output           [6:0]      HEX1,
    output           [6:0]      HEX2,
    output           [6:0]      HEX3,
    output           [6:0]      HEX4,
    output           [6:0]      HEX5,

    ////////////// VGA //////////
    output                      VGA_BLANK_N,
    output           [7:0]      VGA_B,
    output                      VGA_CLK,
    output           [7:0]      VGA_G,
    output                      VGA_HS,
    output           [7:0]      VGA_R,
    output                      VGA_SYNC_N,
    output                      VGA_VS
);



//=======================================================
//  REG/WIRE declarations
//=======================================================

assign VGA_SYNC_N=0;


//=======================================================
//  Structural coding
//=======================================================
top_flyinglogo(.clk(CLOCK_50), .rst(1'b0), .hsync(VGA_HS), .vsync(VGA_VS),
.vga_r(VGA_R[7:4]), .vga_g(VGA_G[7:4]), .vga_b(VGA_B[7:4]), .pclk(VGA_CLK),
.valid(VGA_BLANK_N));


endmodule
```

五、实验步骤

- 根据实验讲义复习 VGA 接口控制器的相关内容。
- 了解如何利用verilog语言设计 VGA 接口控制器。

VGA控制器代码：

表 9-2: VGA 参考代码

```
module vga_ctrl(
    input            pclk,      //25MHz时钟
    input            reset,     //置位
    input  [23:0]    vga_data,  //上层模块提供的VGA颜色数据
    output [9:0]     h_addr,    //提供给上层模块的当前扫描像素点坐标
    output [9:0]     v_addr,
    output           hsync,     //行同步和列同步信号
    output           vsync,
    output           valid,     //消隐信号
    output [7:0]     vga_r,     //红绿蓝颜色信号
    output [7:0]     vga_g,
    output [7:0]     vga_b
    );

   //640x480分辨率下的VGA参数设置
   parameter    h_frontporch = 96;
   parameter    h_active = 144;
   parameter    h_backporch = 784;
   parameter    h_total = 800;

   parameter    v_frontporch = 2;
   parameter    v_active = 35;
   parameter    v_backporch = 515;
   parameter    v_total = 525;

   //像素计数值
   reg [9:0]    x_cnt;
   reg [9:0]    y_cnt;
   wire         h_valid;
   wire         v_valid;

   always @(posedge reset or posedge pclk) //行像素计数
      if (reset == 1'b1)
```

```
            x_cnt <= 1;
        else
        begin
            if (x_cnt == h_total)
                x_cnt <= 1;
            else
                x_cnt <= x_cnt + 10'd1;
        end

    always @(posedge pclk)  //列像素计数
        if (reset == 1'b1)
            y_cnt <= 1;
        else
        begin
            if (y_cnt == v_total & x_cnt == h_total)
                y_cnt <= 1;
            else if (x_cnt == h_total)
                y_cnt <= y_cnt + 10'd1;
        end
    //生成同步信号
    assign hsync = (x_cnt > h_frontporch);
    assign vsync = (y_cnt > v_frontporch);
    //生成消隐信号
    assign h_valid = (x_cnt > h_active) & (x_cnt <= h_backporch);
    assign v_valid = (y_cnt > v_active) & (y_cnt <= v_backporch);
    assign valid = h_valid & v_valid;
    //计算当前有效像素坐标
    assign h_addr = h_valid ? (x_cnt - 10'd145) : {10{1'b0}};
    assign v_addr = v_valid ? (y_cnt - 10'd36) : {10{1'b0}};
    //设置输出的颜色值
    assign vga_r = vga_data[23:16];
    assign vga_g = vga_data[15:8];
    assign vga_b = vga_data[7:0];
endmodule
```

- 利用clkgen实现分频器实现25MHz的时钟周期。
- 先在代码中实现颜色条纹的显示。
- 再利用matlab中的脚本实现图片到mif文件的转换

```
function img2 = img2mif(imgfile,outfile,mysize)

img=imread(imgfile);
img=img(mysize(1):mysize(2),mysize(3):mysize(4),:);

height = mysize(2)-mysize(1)+1;
width = mysize(4)-mysize(3)+1;
s = fopen(outfile,'wb');
fprintf(s,'%s\n','--VGA Memory Map');
fprintf(s,'---Height: %d,Width: %d\n\n',height,width);
fprintf(s,'%s\n','WIDTH=12;');
fprintf(s,'DEPTH=%d;\n',512*width);
```

```
fprintf(s,'%s\n','ADDRESS_RADIX=HEX;');
fprintf(s,'%s\n','DATA_RADIX=HEX;');

fprintf(s,'%s\n','CONTENT');
fprintf(s,'%s\n','BEGIN');
cnt = 0;
img2 =img;
for r=1 :width
    for c=1:512
        cnt = cnt+1;
        if(c<=height)
            R = img(c,r,1);
            G = img(c,r,2);
            B = img(c,r,3);
        else
            R = 0;
            G = 0;
            B = 0;
        end
        fprintf(s,'%06X: %01X%01X%01X;\n',cnt-
1,bitand(R,240)/16,bitand(G,240)/16,bitand(B,240)/16);
    end
end
fprintf(s,'%s\n','END;');

fclose(s);
```

```
运行: img2mif("5.png", "5.mif", [1,512,1,640])

运行: img2mif("picture3.png", "mypicture33", [1,480,1,640])
```

- 进行分析与综合，检查是否有语法错误，最终编译通过。
- 进行编译，待编译通过后将二进制文件写入开发板 ，连接到显示器。
- 在开发板上进行硬件验证。
- 用相似的方法撰写FLY_LOGO.v并测试。

六、测试方法

1. 由于VGA显示不方便用仿真测试，所以直接利用显示器测试。
2. 将sof文件导入开发板中实际测试电路，观察显示器是否符合预期。
3. 在开发板上接入 VGA 显示器，观察条纹和图片是否如预期所示。

七、实验结果

条纹显示和图片显示均正常，但FLY_LOGO仍然存在问题。

八、思考

在实现fly_logo时我按同样的方式转换图片文件，但mif文件就存在问题，而且在显示器上应该反弹的图片能够正常反弹，但只能显示左半部分的图片，可能是在控制输出的时候在大于小图片宽度的区域都置为了0，所以显示黑色。

九、实验中遇到的问题及解决方法

在直接利用课件中给的vga_ctrl.v实现的控制器下发现显示器一直没有显示，后来把屏幕亮度调到最大后发现是有显示的，但是图片非常的暗，所以考虑到可能是RGB的值出了问题，于是重新复习课件，发现每个像素点不能用3个8bit表示，而要改为3个4bit，而且在分配引脚的时候需要分配到RGB引脚的高四位，修改后可以正常显示。

在做拓展功能的时候遇到了问题，因为拓展功能要求的图片比显示全屏要小（我准备的图片是200*200的），所以不能使用之前的matlab脚本生成，修改后的matlab脚本为

```
function img2 = img2mif(imgfile,outfile,mysize)

img=imread(imgfile);
img=img(mysize(1):mysize(2),mysize(3):mysize(4),:);

height = mysize(2)-mysize(1)+1;
width = mysize(4)-mysize(3)+1;
s = fopen(outfile,'wb');
fprintf(s,'%s\n','--VGA Memory Map');
fprintf(s,'---Height: %d,Width: %d\n\n',height,width);
fprintf(s,'%s\n','WIDTH=12;');
fprintf(s,'DEPTH=%d;\n',512*width);
fprintf(s,'%s\n','ADDRESS_RADIX=HEX;');
fprintf(s,'%s\n','DATA_RADIX=HEX;');

fprintf(s,'%s\n','CONTENT');
fprintf(s,'%s\n','BEGIN');
cnt = 0;
img2 =img;
for r=1 :width
    for c=1:512
        cnt = cnt+1;
        if(c<=height)
            R = img(c,r,1);
            G = img(c,r,2);
            B = img(c,r,3);
        else
            R = 0;
            G = 0;
            B = 0;
        end
        fprintf(s,'%06X: %01X%01X%01X;\n',cnt-
1,bitand(R,240)/16,bitand(G,240)/16,bitand(B,240)/16);
    end
end
fprintf(s,'%s\n','END;');

fclose(s);
```

这样可以生成将小图片放在屏幕的左上角，其他均为黑色的mif文件，但由于时间关系，没能按时完成拓展功能，只验收了基础功能。

十、 意见与建议

无