



南京大學

## 数电实验八： VGA 接口控制器实现

课程名称： 数字逻辑与计算机组成实验

姓名： 孙文博

学号： 201830210

班级： 数电一班

邮箱： [201830210@smail.nju.edu.cn](mailto:201830210@smail.nju.edu.cn)

实验时间： 2022.5.11 - 2022.5.19

## 一、实验目的

1. 学习 VGA 接口原理；
2. 学习 VGA 接口控制器的设计方法；

## 二、实验环境

设计\编译环境：Quartus (Quartus Prime 17.1) Lite Edition

开发平台：DE10-Standard

FPGA 芯片：Cyclone II 5CSXFC6D6

VGA 显示器

## 三、实验原理

### 1. VGA 接口及其工作原理

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。

我们使用的 VGA 显示器的分辨率是  $640 \times 480$  的，因此如果想屏幕看起来不闪烁需要刷新频率大于 24，需要将 VGA 频率改为 25MHZ 的，这是通过一个分频模块实现的，如下所示：

表 8-1: 通用时钟生成代码

```
1 module clkgen(  
2     input clkin,  
3     input rst,  
4     input clken,  
5     output reg clkout  
6 );  
7 parameter clk_freq=1000;  
8 parameter countlimit=50000000/2/clk_freq; // 自动计算计数次数  
9  
10 reg[31:0] clkcount;  
11 always @ (posedge clkin)  
12     if(rst)  
13         begin  
14             clkcount=0;  
15             clkout=1'b0;  
16         end  
17     else  
18         begin  
19             if(clken)  
20                 begin  
21                     clkcount=clkcount+1;  
22                     if(clkcount>=countlimit)  
23                         begin  
24                             clkcount=32'd0;  
25                             clkout=~clkout;  
26                         end  
27                     else  
28                         clkout=clkout;  
29                 end  
30             else  
31                 begin  
32                     clkcount=clkcount;  
33                     clkout=clkout;  
34                 end  
35             end  
36 endmodule
```

此时传入参数 2500 0000 就得到了 25MHZ 的同步时钟, 通过计算得知: 在 25MHZ 的时钟周期下总时长为 16.8 毫秒, 所以对应大概约每秒 60 帧大于 24 帧, 人眼感受不到。

每一帧的图像都是从屏幕的左上方开始一行一行进行的, 并且行同步信号是一个负脉冲, 当其有效的时候, RGB 端就会送出当前行显示的各像素点的 RGB 电压值, 每一帧结束后都会由帧同步信号送出一个负脉冲, 重新从屏幕的左上方开始进行扫描。

从实验手册中我们了解到 RGB 端并不是所有时间都在传送像素信息, 会因为电子束从一行尾到下一行头需要时间而出现行消隐时间,

这时 RGB 送出的电压值为 0(黑色)。并且行消隐时间以像素为单位。有效显示一行需要 800 个像素点的时间，其中每行显示 640 个像素点，其中行消隐时间为 160 个像素点。有效显示一帧图像需要 525 行时间，一帧的消隐时间为 40 行。

对于手册中给出的的 VGA 控制信号模块：

表 8-2: VGA 参考代码

```
1 module vga_ctrl(  
2     input          pclk,      //25MHz 时钟  
3     input          reset,    //复位  
4     input  [23:0]   vga_data, //上层模块提供的VGA颜色数据  
5     output [9:0]    h_addr,   //提供给上层模块的当前扫描像素点坐标  
6     output [9:0]    v_addr,  
7     output         hsync,    //行同步和列同步信号  
8     output         vsync,  
9     output         valid,    //消隐信号  
10    output [7:0]    vga_r,    //红绿蓝颜色信号  
11    output [7:0]    vga_g,  
12    output [7:0]    vga_b  
13 );  
14  
15 //640x480分辨率下的VGA参数设置  
16 parameter h_frontporch = 96;  
17 parameter h_active = 144;  
18 parameter h_backporch = 784;  
19 parameter h_total = 800;  
20  
21 parameter v_frontporch = 2;  
22 parameter v_active = 35;  
23 parameter v_backporch = 515;  
24 parameter v_total = 525;  
25  
26 //像素计数值  
27 reg [9:0] x_cnt;  
28 reg [9:0] y_cnt;  
29 wire      h_valid;  
30 wire      v_valid;  
31  
32 always @(posedge reset or posedge pclk) //行像素计数  
33     if (reset == 1'b1)
```

我们需要在顶层模块里面传入该模块需要的参数，然后利用其

中的行和列的扫描坐标，在顶层文件里面修改传入的 `vga_data`，然后在该模块里面用改变的 `vga_data` 来改变相应的 `vga_r`、`vga_g` 以及 `vga_b`，从而实现对屏幕颜色的改变。

## 四、实验过程

### 1. 设计思路

本次实验我们需要分多个模块来实现，每个模块对应不同的功能，最后通过顶层模块实现对他们的调用。

首先我们需要设计一个顶层实体，该实体里面的参数是 VGA 接口需要的参数如下：

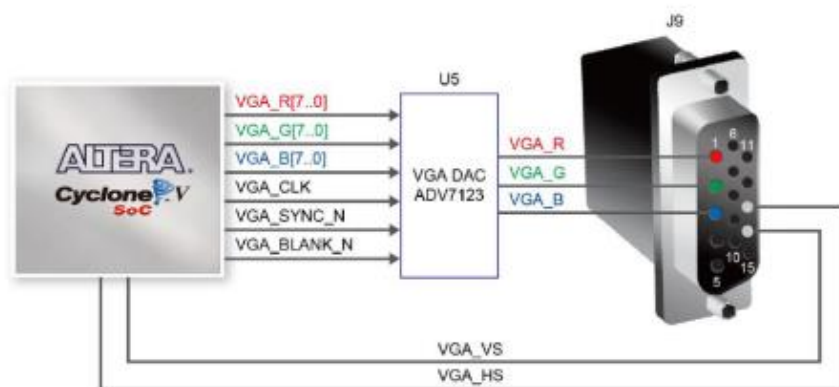


图 8-4: DE10-Standard 的 VGA 连接示意图

然后用实验手册给出的上的 `clkgen` 模块，实例化出来一个 25MHZ 的同步时钟，再用这个 25MHZ 的同步时钟去实例化实验手册中的另一个模块 `VGA_CTRL`，并且根据手册中的 VGA 参考代码模块实例化出来的提供给上层用的像素点的坐标来进行对整个屏幕的颜色控制。下面依次介绍各功能的实现。

## 2. 设计代码

### 顶层模块 lab8.v:

```

1 module lab8(
2     clk_in,
3     clk_en,
4     hsync,
5     vsync,
6     valid,
7     vga_sync_n,
8     vga_r,
9     vga_g,
10    vga_b,
11    VGA_CLK
12);
13
14    input clk_in;
15    input clk_en;
16    output hsync;
17    output vsync;
18    output valid;
19    output vga_sync_n;
20    output [7:0] vga_r;
21    output [7:0] vga_g;
22    output [7:0] vga_b;
23    output VGA_CLK;
24
25    wire [11:0] vga_data;
26    //wire [2:0] vga_data;
27    wire [9:0] h_addr;
28    wire [9:0] v_addr;
29    wire [18:0] addr;
30
31    assign vga_sync_n = 0;
32    assign addr = v_addr+h_addr*512;
33    //assign addr = h_addr+v_addr*640;
34
35    clkgen #(25000000) vgaClks(clk_in,1'b0,1'b1,VGA_CLK);
36    roms romx(.address(addr), .clock(VGA_CLK), .q(vga_data));
37
38
39    vga_ctrl VGA(
40        .pclk(VGA_CLK), //25MHz 时钟
41        .reset(1'b0), //复位
42        .vga_data({vga_data}), //上层模块提供的VGA颜色数据
43        .h_addr(h_addr), //提供给上层模块的当前扫描像素点坐标
44        .v_addr(v_addr),
45        .hsync(hsync), //行同步和列同步信号
46        .vsync(vsync),
47        .valid(valid), //消隐信号
48        .vga_r(vga_r), //红绿蓝颜色信号
49        .vga_g(vga_g),
50        .vga_b(vga_b)
51    );
52
53 endmodule

```

正如设计思路中所述的, 用扫描的行坐标来进行对整个屏幕的划分, 先后调用 vgaClks、roms 和 vga\_ctrl 三个模块, 将开发板 VGA 所需要的接口放在顶层模块的引脚上。

### 存储器模块 roms.v:

```

1 module roms(
2     input [18:0] address,
3     input clock,
4     output reg [11:0] q
5 );
6
7     // 通过.mif文件初始化vga_data
8     (* ram_init_file = "my_picture.mif" *) reg [11:0] myrom[327679:0];
9     //(* ram_init_file = "tuozhan_picture.mif" *) reg [11:0] myrom[327679:0];
10    //(* ram_init_file = "my_test.mif" *) reg [15:0] myrom[327199:0];
11    //(* ram_init_file = "my_test1.mif" *) reg [2:0] myrom[327199:0];
12
13    always @(posedge clock) begin
14        q = myrom[address];
15    end
16
17 endmodule

```

按照实验手册的要求我们需要实现一个存储器用来存储一张静态图片的信息，这里用到的图片以.mif(存储器内存)格式保存，一张来自于课程网站的范例，另一张是通过图片软件将自己的某张.bmp 格式图片转化后的.mif 文件。

## 时钟模块 clkgen.v:

```
1 module clkgen(  
2     input clk_in,  
3     input rst,  
4     input clken,  
5     output reg clkout  
6 );  
7  
8 parameter clk_freq=1000;  
9 parameter countlimit=5000000/2/clk_freq; // 自动计算计数次数  
10 reg[31:0] clkcount;  
11 always @ (posedge clk_in)  
12 if(rst)  
13 begin  
14     clkcount=0;  
15     clkout=1'b0;  
16 end  
17 else  
18 begin  
19     if(clken)  
20     begin  
21         clkcount=clkcount+1;  
22         if(clkcount>=countlimit)  
23         begin  
24             clkcount=32'd0;  
25             clkout=~clkout;  
26         end  
27         else  
28             clkout=clkout;  
29         end  
30     else  
31     begin  
32         clkcount=clkcount;  
33         clkout=clkout;  
34     end  
35 end  
36 endmodule
```

## 控制模块 vgactrl.v:

```
1 module vga_ctrl(  
2     input pclk, //25MHz 时钟  
3     input reset, //复位  
4     input [11:0] vga_data, //上层模块提供的VGA颜色数据  
5     //input [2:0] vga_data, //上层模块提供的VGA颜色数据  
6     output [9:0] h_addr, //提供给上层模块的当前扫描像素点坐标  
7     output [9:0] v_addr,  
8     output hsync, //行同步和列同步信号  
9     output vsync,  
10    output valid, //消隐信号  
11    output [7:0] vga_r, //红绿蓝颜色信号  
12    output [7:0] vga_g,  
13    output [7:0] vga_b  
14 );  
15  
16 //640x480分辨率下的VGA参数设置  
17 parameter h_frontporch = 96;  
18 parameter h_active = 144;  
19 parameter h_backporch = 784;  
20 parameter h_total = 800;  
21  
22 parameter v_frontporch = 2;  
23 parameter v_active = 35;  
24 parameter v_backporch = 515;  
25 parameter v_total = 525;  
26  
27 //像素计数值  
28 reg [9:0] x_cnt;  
29 reg [9:0] y_cnt;  
30 wire h_valid;  
31 wire v_valid;  
32  
33 always @(posedge reset or posedge pclk) // 行像素计数  
34 if (reset == 1'b1)  
35     x_cnt <= 1;  
36 else  
37     begin  
38         if (x_cnt == h_total)
```

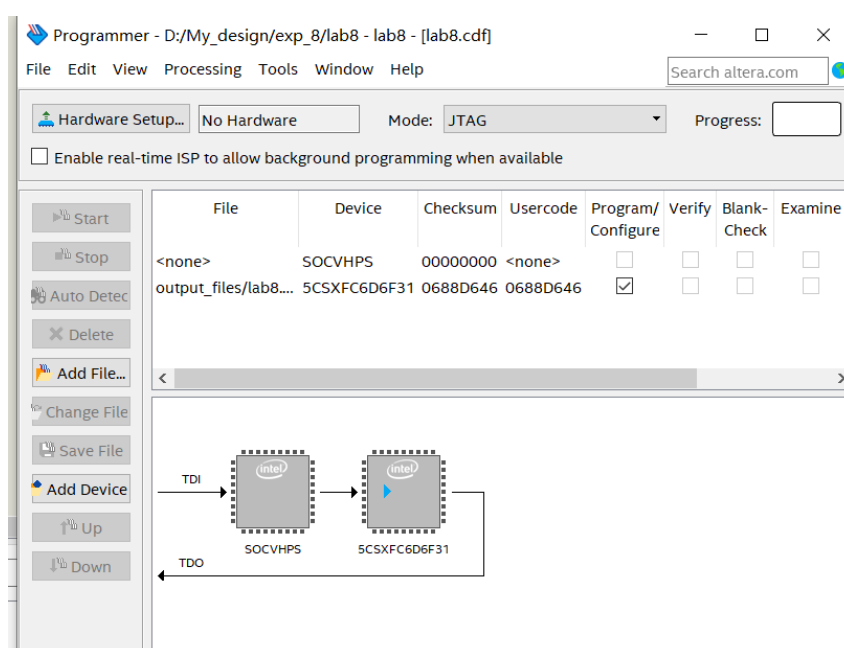
```

39         x_cnt <= 1;
40     else
41         x_cnt <= x_cnt + 10'd1;
42     end
43
44     always @(posedge clk) // 列像素计数
45     if (reset == 1'b1)
46         y_cnt <= 1;
47     else
48     begin
49         if (y_cnt == v_total & x_cnt == h_total)
50             y_cnt <= 1;
51         else if (x_cnt == h_total)
52             y_cnt <= y_cnt + 10'd1;
53         end
54         //生成同步信号
55         assign hsync = (x_cnt > h_frontporch);
56         assign vsync = (y_cnt > v_frontporch);
57         //生成消隐信号
58         assign h_valid = (x_cnt > h_active) & (x_cnt <= h_backporch);
59         assign v_valid = (y_cnt > v_active) & (y_cnt <= v_backporch);
60         assign valid = h_valid & v_valid;
61         //计算当前有效像素坐标
62         assign h_addr = h_valid ? (x_cnt - 10'd144) : {10{1'b0}};
63         assign v_addr = v_valid ? (y_cnt - 10'd35) : {10{1'b0}};
64         //设置输出的颜色值
65         assign vga_r = {vga_data[11:8], 1'b0, 1'b0, 1'b0, 1'b0};
66         assign vga_g = {vga_data[7:4], 1'b0, 1'b0, 1'b0, 1'b0};
67         assign vga_b = {vga_data[3:0], 1'b0, 1'b0, 1'b0, 1'b0};
68         // RGB 111
69         assign vga_r = {vga_data[2], 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0};
70         assign vga_g = {vga_data[1], 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0};
71         assign vga_b = {vga_data[0], 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0};
72     endmodule
73
74

```

接着将我们的四个模块编译运行，进行引脚分配，烧录到开发板上：

Node Name	Direction	Location	I/O Bank	VREF Group	Pin Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Analog Setting	IOB/VCCCT_Gx1	IO Pin Termination	Configured Refclk	Common Mode	Input Slew Rate	Differential Output
VGA_CLK	Output	PIN_AK21	4A	B4A_NO	PIN_AK21	2.5 V		12mA...auto	1 (default)								
clken	Input	PIN_AA30	5B	B5B_NO	PIN_AA30	2.5 V		12mA...auto									
hsync	Output	PIN_AK19	4A	B4A_NO	PIN_AK19	2.5 V		12mA...auto	1 (default)								
valid	Output	PIN_AK22	4A	B4A_NO	PIN_AK22	2.5 V		12mA...auto	1 (default)								
vga_b[7]	Output	PIN_AK16	4A	B4A_NO	PIN_AK16	2.5 V		12mA...auto	1 (default)								
vga_b[6]	Output	PIN_AJ16	4A	B4A_NO	PIN_AJ16	2.5 V		12mA...auto	1 (default)								
vga_b[5]	Output	PIN_AJ17	4A	B4A_NO	PIN_AJ17	2.5 V		12mA...auto	1 (default)								
vga_b[4]	Output	PIN_AH19	4A	B4A_NO	PIN_AH19	2.5 V		12mA...auto	1 (default)								
vga_b[3]	Output	PIN_AJ19	4A	B4A_NO	PIN_AJ19	2.5 V		12mA...auto	1 (default)								
vga_b[2]	Output	PIN_AH20	4A	B4A_NO	PIN_AH20	2.5 V		12mA...auto	1 (default)								
vga_b[1]	Output	PIN_AJ20	4A	B4A_NO	PIN_AJ20	2.5 V		12mA...auto	1 (default)								
vga_b[0]	Output	PIN_AJ21	4A	B4A_NO	PIN_AJ21	2.5 V		12mA...auto	1 (default)								
vga_g[7]	Output	PIN_AH23	4A	B4A_NO	PIN_AH23	2.5 V		12mA...auto	1 (default)								
vga_g[6]	Output	PIN_AK23	4A	B4A_NO	PIN_AK23	2.5 V		12mA...auto	1 (default)								
vga_g[5]	Output	PIN_AH24	4A	B4A_NO	PIN_AH24	2.5 V		12mA...auto	1 (default)								
vga_g[4]	Output	PIN_AJ24	4A	B4A_NO	PIN_AJ24	2.5 V		12mA...auto	1 (default)								
vga_g[3]	Output	PIN_AK24	4A	B4A_NO	PIN_AK24	2.5 V		12mA...auto	1 (default)								
vga_g[2]	Output	PIN_AH25	4A	B4A_NO	PIN_AH25	2.5 V		12mA...auto	1 (default)								
vga_g[1]	Output	PIN_AJ25	4A	B4A_NO	PIN_AJ25	2.5 V		12mA...auto	1 (default)								
vga_g[0]	Output	PIN_AK26	4A	B4A_NO	PIN_AK26	2.5 V		12mA...auto	1 (default)								
vga_r[7]	Output	PIN_AJ26	4A	B4A_NO	PIN_AJ26	2.5 V		12mA...auto	1 (default)								
vga_r[6]	Output	PIN_AK26	4A	B4A_NO	PIN_AK26	2.5 V		12mA...auto	1 (default)								
vga_r[5]	Output	PIN_AJ26	4A	B4A_NO	PIN_AJ26	2.5 V		12mA...auto	1 (default)								
vga_r[4]	Output	PIN_AH27	4A	B4A_NO	PIN_AH27	2.5 V		12mA...auto	1 (default)								
vga_r[3]	Output	PIN_AJ27	4A	B4A_NO	PIN_AJ27	2.5 V		12mA...auto	1 (default)								
vga_r[2]	Output	PIN_AK27	4A	B4A_NO	PIN_AK27	2.5 V		12mA...auto	1 (default)								





### 3. 仿真模型检测

本次实验的仿真模型较为复杂（需要调用多个模块）且并无实际意义（无法检测是否能在显示器上显示），因此仿真检测环节无需耽误时间。接下来直接上板调试并 debug，等待验收！

## 五、实验结果

### 1. 思考题

本次实验暂无思考题。

### 2. 上板验收

使用开发板通过 VGA 接口连接到显示器上，实现了自己单独的照片的显示（偏搞笑风格的一张图片）：



## 六、总结与反思

本次实验的时间其实挺紧的（键盘后两周），加上期中考试和其他实验，所以做的过程中常常因为着急而犯错，好在和另一位同学互帮互助修复了过程中的许多 bug，以及反复研读手册理解 VGA 原理之

后，实验还算顺利的完成了，最后图片出现在屏幕上的那一刻真的 excited!（上板的前两天一直对着黑屏 debug 的痛苦太深了）接下来的实验中我们将把前两次实验合并，实现键盘的交互和屏幕的显示，希望可以顺利完成吧！