2020-11-4

数字电路实验九 VGA 接口控制器实现

秦嘉余

191220088

1348288404@qq.com

目录

1.实验目的 1
2.实验原理 1
3.实验环境/器材 2
4. 显示不同颜色条纹 2
1. 设计思路 2
2. 实验代码 2
3.RTL 视图 4
4.引脚分配 4
5.实验结果 5
5. 图片显示 5
1. 设计思路 5
2. 实验代码 5
3.RTL 视图 5
4.引脚分配 6
5.实验结果 6
6. 拓展功能 6
1. 设计思路 6
2. 实验代码 6
3.RTL 视图 8
4.引脚分配 8
5.实验结果 8
7.实验实验中遇到的问题及解决办法 8

1.实验目的

本实验的目的是了解 VGA 的数据传输形式以及其相关信号，并在显示屏上显示相关内容

2.实验原理

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）。

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。
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。

3.实验环境/器材

本次实验的环境为 Quartus17.1 版本
本次实验的器材为 DE10 Standard 开发板

4. 显示不同颜色条纹

1. 设计思路

本次实验中，相关的 VGA 接口的代码设计以及给出，只需要理解就可以直接使用，我们需要写的内容仅有为：当横纵坐标改变时，对颜色进行赋值

2. 实验代码

给出的 vga_ctrl 模块代码

```
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
begin
always @(posedge pclk) // 列像 素 计数
begin
// 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
```

可变时钟模块

```
module clkgen( input clkin, input rst,input clken,output reg clkout );
    parameter clk_freq=1000;
    parameter countlimit=50000000/2/clk_freq; //自动计算计数次数
    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
```

颜色选择模块

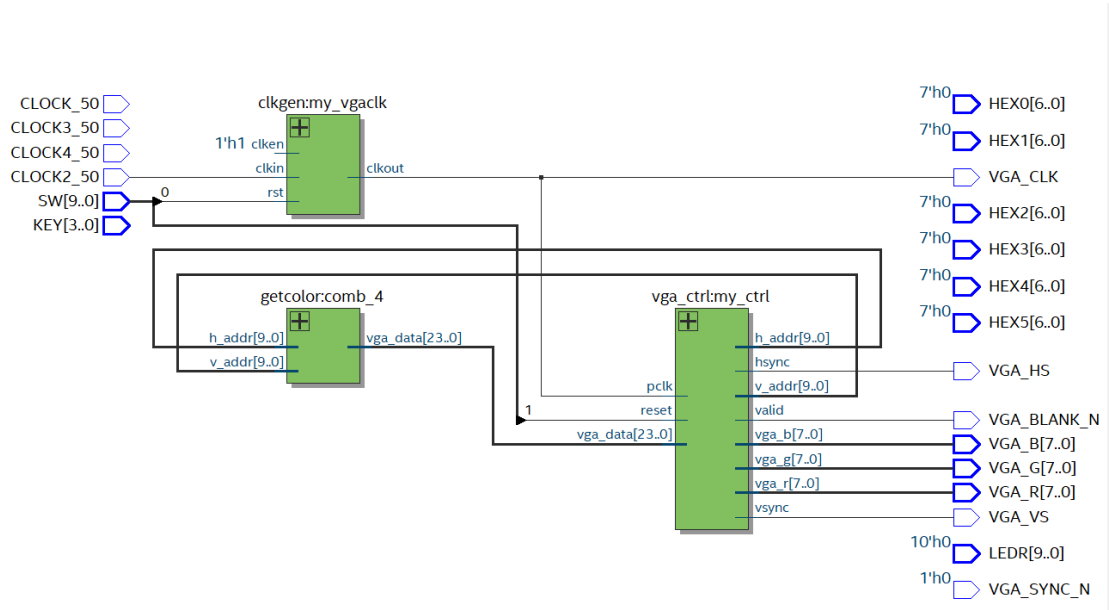
```
module getcolor(input [9:0] h_addr,input [9:0] v_addr,output reg [23:0] vga_data);

    initial begin
       vga_data=24'h000000;
    end


    always @(h_addr)
    begin
       if(h_addr>240 && h_addr<260)
          vga_data<=24'hFF3300;
       else if(h_addr>400 && h_addr<420)
          vga_data<=24'h0066FF;
       else
          vga_data<=24'h000000;
    end

endmodule
```

该模块为自己设计的模块，在条纹显示实验中较为简单，直接在横纵坐标一定范围内输入颜色信号即可

主模块调用如下：

```
 wire [9:0] h_addr,v_addr;
 wire [23:0] vga_data;
   assign VGA_SYNC_N=0;


//=======================================================
//  Structural coding
//=======================================================
   clkgen #(25000000) my_vgaclk(CLOCK2_50,SW[0],1'b1,VGA_CLK);

   vga_ctrl my_ctrl(VGA_CLK,SW[1],vga_data,h_addr,v_addr,VGA_HS,VGA_VS,VGA_BLANK_N,VGA_R,VGA_G,VGA_B);
   getcolor(h_addr,v_addr,vga_data);

endmodule
```

3.RTL 视图



4.引脚分配

该实验为 system 软件自动分配的引脚，所以不再展示、

5.实验结果

本实验已在开发板上验收

5. 图片显示

1. 设计思路

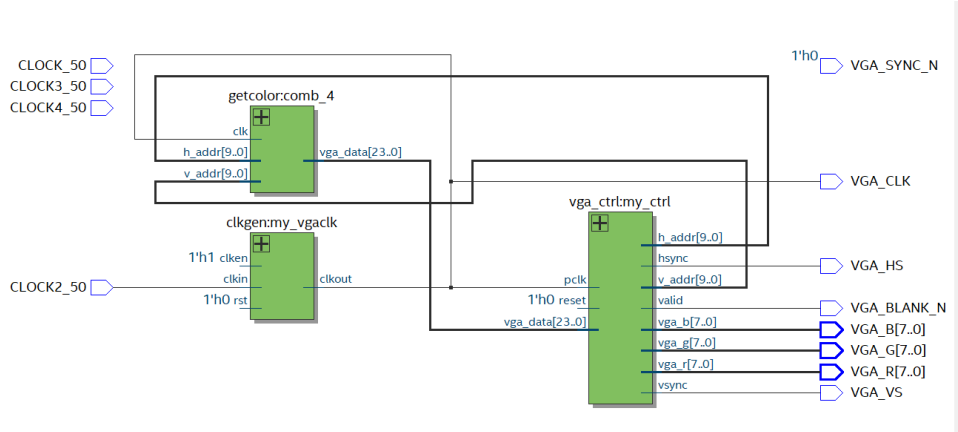
本次实验中，相关的 VGA 接口的代码设计以及给出，只需要理解就可以直接使用，我们需要写的内容仅有为：当横纵坐标改变时，对颜色进行赋值。此时由于图片的颜色内容复杂，所以需要建立 rom，通过 rom 赋值

2. 实验代码

VGA 相关模块与上述相同，只展示 getcolor 模块

```
module getcolor(input clk,input [9:0] h_addr,input [9:0] v_addr,output reg [23:0] vga_data);

   wire [18:0] addr;
   wire [11:0] data;

   initial begin
      vga_data=24'h000000;
   end

   assign addr={h_addr,v_addr[8:0]};
   myrom rom(addr,clk,data);

   always @(h_addr or v_addr)
   begin
      vga_data<={data[11:8],4'b0000,data[7:4],4'b0000,data[3:0],4'b0000};
   end

endmodule
```

3.RTL 视图



4.引脚分配

该实验为 system 软件自动分配的引脚，所以不再展示、

5.实验结果

本实验已在开发板上验收

6. 拓展功能

1. 设计思路

对于自定义图片，只需要生成相关的 mif 文件即可，与 5 相同，所以不再赘述。对于图片的移动，代码已给出，只需要略微修改即可

2. 实验代码

移动相关模块

```
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 [13:0]       rom_addr;
   wire [11:0]       douta;

   wire             logo_area;
   reg [9:0]        logo_x;
   reg [9:0]        logo_y;
   parameter [9:0] logo_length = 10'd100;
   parameter [9:0] logo_hight  = 10'd100;

   reg [7:0]        speed_cnt;
   wire             speed_ctrl;

   reg [3:0]        flag_edge;
```

debounce 模块

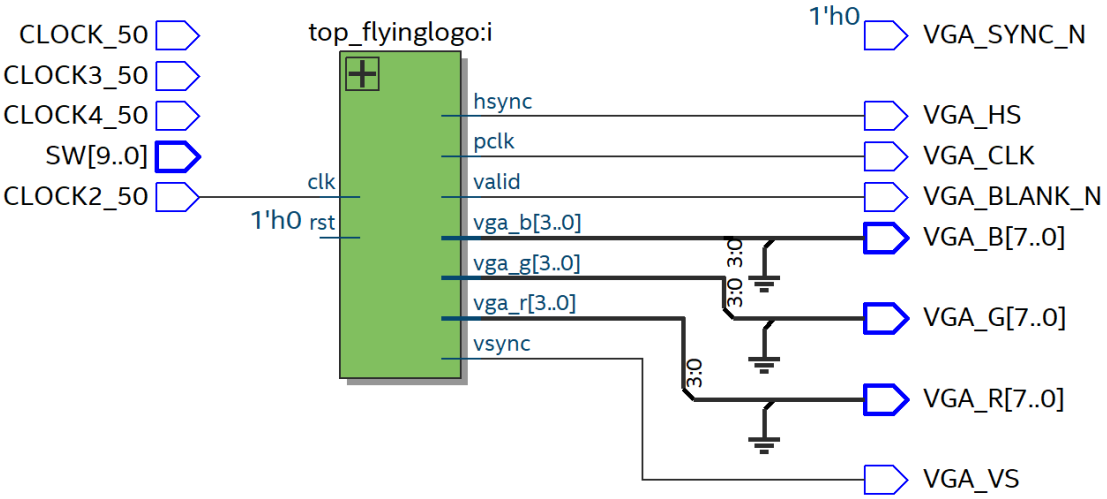
```
module debounce(
    input clk,
    input  sig_in,
    output sig_out
    );

   reg q1;
   reg q2;
   reg q3;

   always @ (posedge clk)
   begin
      q1 <= sig_in;
      q2 <= q1;
      q3 <= q2;
   end

   assign sig_out = q1 & q2 & (!q3);

endmodule
```

顶层模块

```
module move(

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

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

   ////////////// 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=1'b0;
   assign VGA_R[3:0]=4'b0000;
   assign VGA_G[3:0]=4'b0000;
   assign VGA_B[3:0]=4'b0000;

top_flyinglogo i(CLOCK2_50, 1'b0, VGA_HS, VGA_VS, VGA_R[7:4], VGA_G[7:4], VGA_B[7:4],VGA_CLK,VGA_BLANK
```

唯一的修改为将 top_flyinglogo 模块的 VGA_CLK, VGA_BLANK_N 变量传给顶层模块, 给 VGA 接口作为信号，否则显示器没有相关信号

3.RTL 视图



4.引脚分配

该实验为 system 软件自动分配的引脚，所以不再展示、

5.实验结果

本实验已在开发板上验收

7.实验实验中遇到的问题及解决办法

本实验比较简单，实验过程较为顺畅，一开始在用 IP 建立大容量 ROM 时，没有选择正确的 ROM 无法生成比较大 ROM 耗费了一些时间，后来改为正确的 IP 即完成了实验。最后调用 op_flyinglogo 模块时，没有修改导致 VGA_CLK，VGA_BLANK_N 变量没有传给顶层模块，显示器无信号，传给顶层后即解决了这个问题