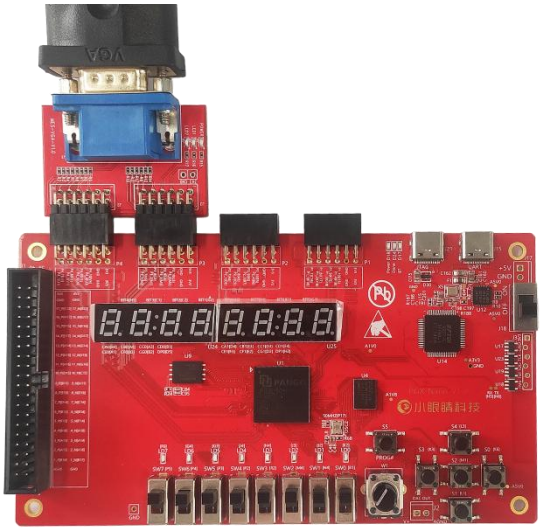
31. VGA 输出彩条实验说明

31.1. 实验简介

实验使用“小眼睛科技”公司的 FPGA 开发板以及 PMOD 转 VGA 转接板，通过 VGA 接口在显示屏屏幕上显示彩条，其中显示格式为 1024*768@60。



31.2. 实验目的

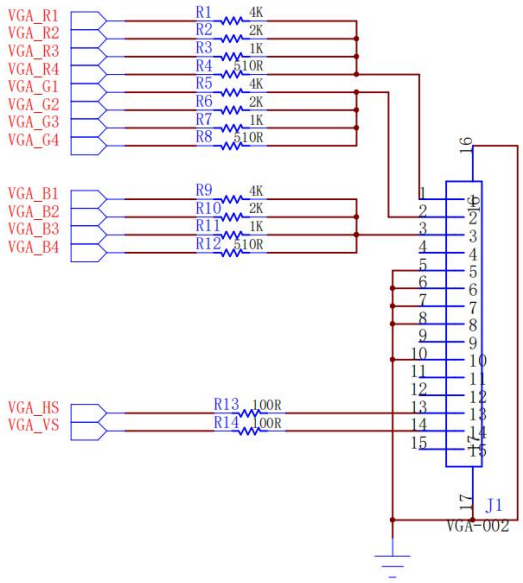
使用 VGA 线连接开发板上的 PMOD-VGA 转接板与显示器，在 VGA 屏幕上完成彩条显示实验。

31.3. 实验使用模块介绍

实验使用“小眼睛科技”公司的 PMOD 转 VGA 接口模块。模块共有两个 PMOD 2x6 针连接器（J2 与 J3）。

使用模块进行管脚绑定时，请仔细观察针脚位置进行管脚的绑定。

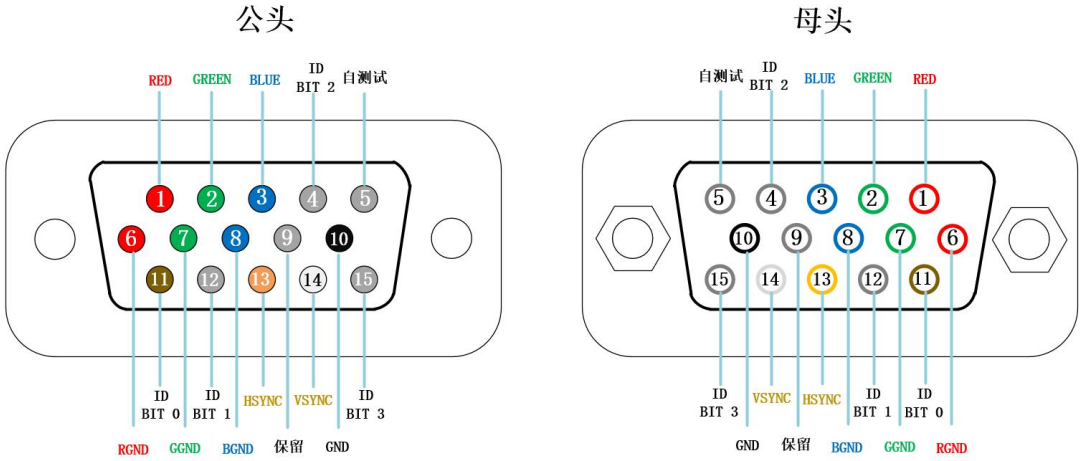
由于 VGA 输入信号为模拟量，FPGA 输出信号为数字量，因此模块使用权电阻网络将数字量转换为模拟量提供给 VGA。



31.4. 实验原理

31.4.1. VGA 接口

VGA（Video Graphics Array）**视频图形阵列**，于 1978 被 **IBM** 公司提出，是电脑显示器上应用广泛的视频接口。VGA 接口一共 15 针孔，一共三行，每行 5 孔，其中， 5 个 GND 信号，3 个 RGB 彩色分量信号，1 个行同步信号，1 个场同步信号，1 个电源信号。



15 针脚定义如下表所示：

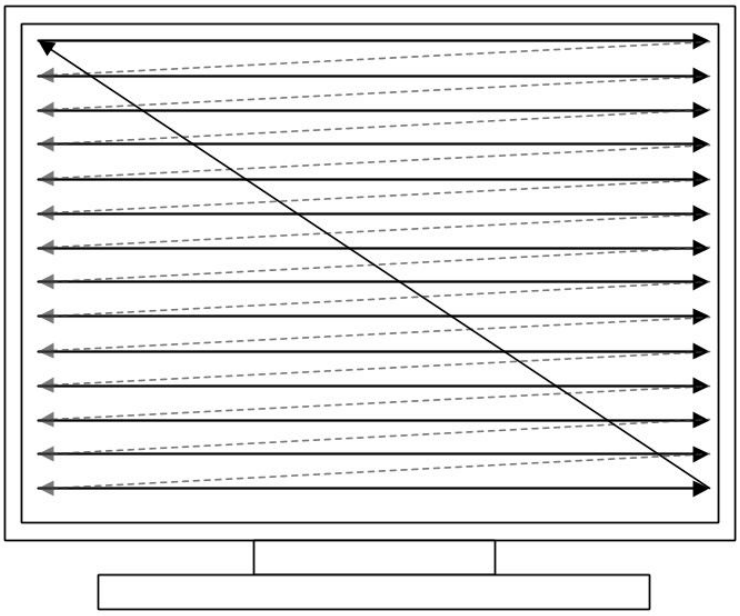
针脚	定义	针脚	定义
1	红基色	9	保留

2	绿基色	10	数字地
3	蓝基色	11	地址码
4	地址码	12	地址码
5	自测试	13	行同步
6	红地	14	场同步
7	绿地	15	地址码
8	蓝地		

31.4.2. VGA 显示

VGA 显示图像使用扫描的方式，从**第一行第一个像素点**开始扫描，扫描到**第一行最后一个像素点**后，开始扫描**第二行第一个像素点**，以此类推，直到扫描到**最后一行最后一个像素点**，即完成一帧图像的显示，返回第一行第一个像素点，开始第二帧图像的扫描。整体扫描路线为**Z 字形**。

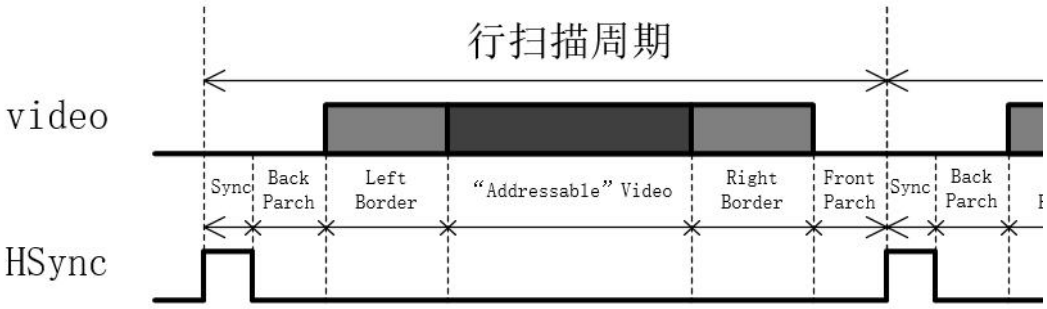
当扫描速度足够快时，由于人眼的视觉暂留特性，就可以观看到一个完整的画面；一个画面又称为一场或一帧，每秒钟刷新的帧数又称为帧率。



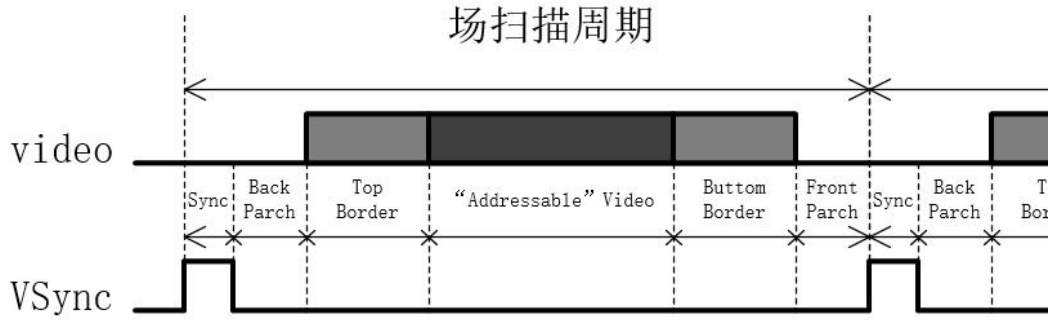
31.4.3. VGA 时序

VGA 显示时像素点的颜色信号源源不断的传输，此时行、场的切换需要行场同步信号控制。

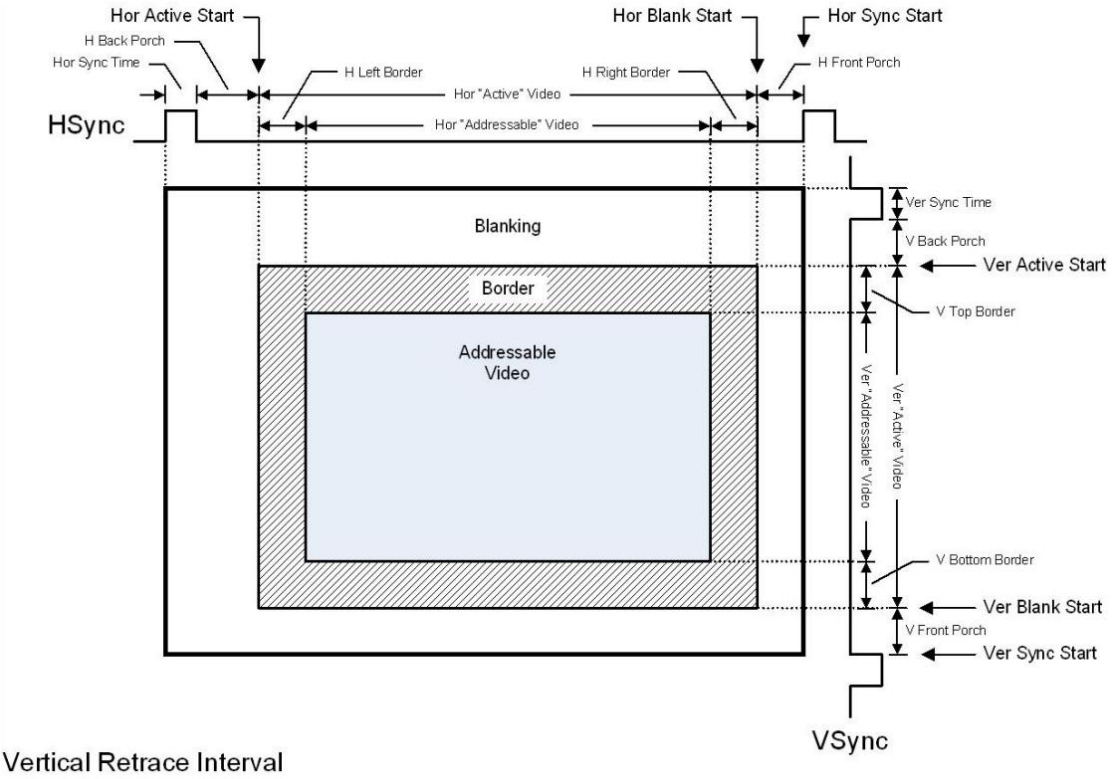
行同步信号用于标志一行的开始，同样也标志上一行的结束，行同步信号的时序如下图所示：



场同步信号用于标志一帧或一场画面的开始，同样也标志着上一场画面的结束，场同步信号的时序如下图所示：



整体的时序如下图所示：



上图中 Addressable 部分内容是在显示器中可看到的区域，Border 可理解为显示黑边或者显示边框，通常 Border（边框）显示黑色。行、场切换过程都是在用户感受不到的区域进行的，这个区域就是 Blanking 部分，称为消隐区间。同步信号上升沿表示新的一行/一场开始，Hsync 对应行，Vsync 对应场。

因此，VGA 刷新一帧的时间不仅与有效像素有关，还与消影、同步的时间有关。

本实验采用 1024*768@60 的视频规格，详细时序参数如下：

VESA MONITOR TIMING STANDARD

Adopted:	9/10/91 (VESA #901101A)
Resolution:	1024 x 768 at 60 Hz (non-interlaced)
EDID ID:	DMT ID: 10h; STD 2 Byte Code: (61, 40)h; CVT 3 Byte Code: n/a
BIOS Modes:	104h, 105h, 116h, 117h, & 118h (4, 8, 15, 16, & 24 bpp)
Method:	***** NOT CVT COMPLIANT *****

Detailed Timing Parameters

Timing Name	= **1024 x 768 @ 60Hz;**					
Hor Pixels	= **1024;**	// Pixels				
Ver Pixels	= **768;**	// Lines				
Hor Frequency	= 48.363;	// kHz	=	20.7 usec	/ line	
Ver Frequency	= 60.004;	// Hz	=	16.7 msec	/ frame	
Pixel Clock	= **65.000;**	// MHz	=	15.4 nsec	± 0.5%	
Character Width	= **8;**	// Pixels	=	123.1 nsec		
Scan Type	= **NONINTERLACED;**			// H Phase =	5.1 %	
Hor Sync Polarity	= **NEGATIVE;**	// HBlank	=	23.8% of HTotal		
Ver Sync Polarity	= **NEGATIVE;**	// VBlank	=	4.7% of VTotal		
Hor Total Time	= 20.677;	// (usec)	=	168 chars	=	1344 Pixels
Hor Addr Time	= 15.754;	// (usec)	=	128 chars	=	1024 Pixels
Hor Blank Start	= 15.754;	// (usec)	=	128 chars	=	1024 Pixels
Hor Blank Time	= 4.923;	// (usec)	=	40 chars	=	320 Pixels
Hor Sync Start	= 16.123;	// (usec)	=	131 chars	=	1048 Pixels
// H Right Border	= 0.000;	// (usec)	=	**0** chars	=	0 Pixels
// H Front Porch	= 0.369;	// (usec)	=	**3** chars	=	24 Pixels
Hor Sync Time	= 2.092;	// (usec)	=	**17** chars	=	136 Pixels
// H Back Porch	= 2.462;	// (usec)	=	**20** chars	=	160 Pixels
// H Left Border	= 0.000;	// (usec)	=	**0** chars	=	0 Pixels
Ver Total Time	= 16.666;	// (msec)	=	806 lines	HT – (1.06xHA)	
Ver Addr Time	= 15.880;	// (msec)	=	768 lines	= 3.98	
Ver Blank Start	= 15.880;	// (msec)	=	768 lines		
Ver Blank Time	= 0.786;	// (msec)	=	38 lines		
Ver Sync Start	= 15.942;	// (msec)	=	771 lines		
// V Bottom Border	= 0.000;	// (msec)	=	**0** lines		
// V Front Porch	= 0.062;	// (msec)	=	**3** lines		
Ver Sync Time	= 0.124;	// (msec)	=	**6** lines		
// V Back Porch	= 0.600;	// (msec)	=	**29** lines		
// V Top Border	= 0.000;	// (msec)	=	**0** lines		

31.5. 实验模块设计

实验采用 1024*768@60 的视频规格，VGA 时钟为：1344*806*60=65MHz；通过视频规格参数，在行同步位置拉高行同步信号，在场同步位置拉高场同步信号，在有效像素位置输入基色信号。

```
`timescale 1ns / 1ps
`define UD #1

module vga_test(
   input             clk ,
   input             rstn ,
   output     reg    vs_out ,
   output     reg    hs_out ,
   output     reg[3:0]r_out ,
   output     reg[3:0]g_out ,
   output     reg[3:0]b_out
   );

parameter   X_WIDTH = 4'd12;
parameter   Y_WIDTH = 4'd12;

reg    de_out ;

    parameter V_TOTAL = 12'd806;
    parameter V_TB = 12'd0;
    parameter V_BB = 12'd0;
    parameter V_FP = 12'd3;
    parameter V_BP = 12'd29;
    parameter V_SYNC = 12'd6;
    parameter V_ACT = 12'd768;
    parameter H_TOTAL = 12'd1344;
    parameter H_LB = 12'd0;
    parameter H_RB = 12'd0;
    parameter H_FP = 12'd24;
    parameter H_BP = 12'd160;
    parameter H_SYNC = 12'd136;
    parameter H_ACT = 12'd1024;
    parameter HV_OFFSET = 12'd0;

    localparam H_ACT_ARRAY_0 = H_ACT/8;
    localparam H_ACT_ARRAY_1 = 2* (H_ACT/8);
    localparam H_ACT_ARRAY_2 = 3* (H_ACT/8);
    localparam H_ACT_ARRAY_3 = 4* (H_ACT/8);
    localparam H_ACT_ARRAY_4 = 5* (H_ACT/8);
    localparam H_ACT_ARRAY_5 = 6* (H_ACT/8);
    localparam H_ACT_ARRAY_6 = 7* (H_ACT/8);
    localparam H_ACT_ARRAY_7 = 8* (H_ACT/8);

    reg [X_WIDTH-1:0]        h_count = 'd0;
```

```
reg [Y_WIDTH-1:0]        v_count = 'd0;

reg [X_WIDTH-1:0]        x_act = 'd0;
reg [Y_WIDTH-1:0]        y_act = 'd0;

reg hs ;
reg vs ;
reg de ;

U_PLL PLL_PIX (
      .clkout0(pix_clk),    // output
      .lock(lock),          // output
      .clkin1(clk)       // input
 );

/* horizontal counter */
always @(posedge pix_clk)
begin
    if (!rstn)
        h_count <= `UD 0;
    else
    begin
        if (h_count < H_TOTAL - 1)
            h_count <= `UD h_count + 1;
        else
            h_count <= `UD 0;
    end
end

/* vertical counter */
always @(posedge pix_clk)
begin
    if (!rstn)
        v_count <= `UD 0;
    else
    if (h_count == H_TOTAL - 1)
    begin
        if (v_count == V_TOTAL - 1)
            v_count <= `UD 0;
        else
            v_count <= `UD v_count + 1;
    end
end
```

```
        always @(posedge pix_clk)
        begin
            if (!rstn)
                hs <= `UD 1'b0;
            else
                hs <= `UD ((h_count < H_SYNC));
        end

        always @(posedge pix_clk)
        begin
            if (!rstn)
                vs <= `UD 4'b0;
            else
            begin
                if (v_count == 0)
                    vs <= `UD 1'b1;
                else if (v_count == V_SYNC)
                    vs <= `UD 1'b0;
                else
                    vs <= `UD vs;
            end
        end

        always @(posedge pix_clk)
        begin
            if (!rstn)
                de <= `UD 1'b0;
            else
                de <= (((v_count >= V_SYNC + V_BP + V_TB) && (v_count <= V_TOTAL - V_FP
- V_BB - 1)) && ((h_count >= H_SYNC + H_BP + H_LB) && (h_count <= H_TOTAL - H_FP - H_RB
- 1)));
        end

        // active pixels counter output
        always @(posedge pix_clk)
        begin
            if (!rstn)
                x_act <= `UD 'd0;
            else
            begin
            /* X coords - for a backend pattern generator */
                if(h_count > (H_SYNC + H_BP + H_LB - 1'b1))
                    x_act <= `UD (h_count - (H_SYNC + H_BP+ H_LB));
                else
```

```
                x_act <= `UD 'd0;
        end
    end

    always @(posedge pix_clk)
    begin
        if (!rstn)
            y_act <= `UD 'd0;
        else
        begin
            /* Y coords - for a backend pattern generator */
            if(v_count > (V_SYNC + V_BP + V_TB - 1'b1))
                y_act <= `UD (v_count - (V_SYNC + V_BP + V_TB));
            else
                y_act <= `UD 'd0;
        end
    end

    always @(posedge pix_clk)
    begin
        vs_out <= `UD vs;
        hs_out <= `UD hs;
        de_out <= `UD de;
    end

    always @(posedge pix_clk)
    begin
        if (de)
        begin
            if(x_act < H_ACT_ARRAY_0)
            begin
                r_out <= 4'hf;
                g_out <= 4'hf;
                b_out <= 4'hf;
            end
            else if(x_act < H_ACT_ARRAY_1)
            begin
                r_out <= 4'hf;
                g_out <= 4'hf;
                b_out <= 4'h0;
            end
            else if(x_act < H_ACT_ARRAY_2)
            begin
                r_out <= 4'h0;
```

```
                g_out <= 4'hf;
                b_out <= 4'hf;
            end
            else if(x_act < H_ACT_ARRAY_3)
            begin
                r_out <= 4'h0;
                g_out <= 4'hf;
                b_out <= 4'h0;
            end
            else if(x_act < H_ACT_ARRAY_4)
            begin
                r_out <= 4'hf;
                g_out <= 4'h0;
                b_out <= 4'hf;
            end
            else if(x_act < H_ACT_ARRAY_5)
            begin
                r_out <= 4'hf;
                g_out <= 4'h0;
                b_out <= 4'h0;
            end
            else if(x_act < H_ACT_ARRAY_6)
            begin
                r_out <= 4'h0;
                g_out <= 4'h0;
                b_out <= 4'hf;
            end
            else
            begin
                r_out <= 4'h5;
                g_out <= 4'h5;
                b_out <= 4'h5;
            end
        end
        else
        begin
            r_out <= 4'h0;
            g_out <= 4'h0;
            b_out <= 4'h0;
        end
    end

endmodule
```

31.6. 实验现象

使用 VGA 线连接开发板上的 PMOD-VGA 转接板与显示器，下载程序，可以看到显示器显示 8 条彩条。