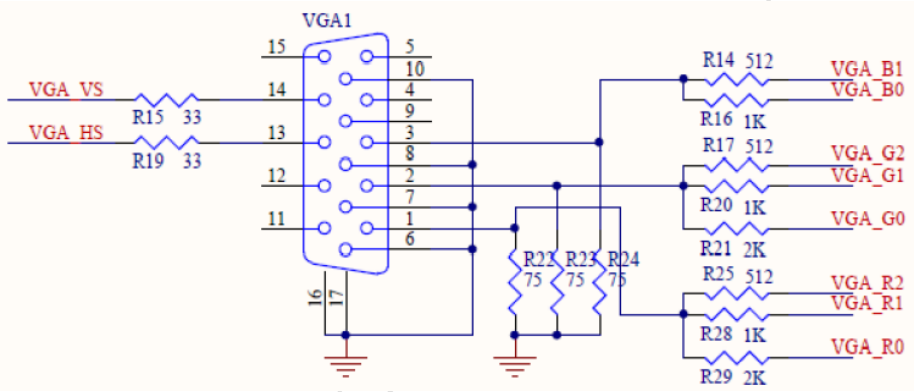
炼狱传奇-VGA 之战

VGA(Video Graphics Array,视频图形阵列)，是 IBM 于 1987 年提出的一个使用类比讯号的电脑显示标准。VGA 是最多制造商所共同支持的一个低标准，个人电脑在加载自己的独特驱动程式之前，都必须支持 VGA 的标准。

今天我们就来学习一下这个接口的驱动时序，让我们的设计从此色彩纷呈。首先我们看一下 VGA 接口的电路原理图



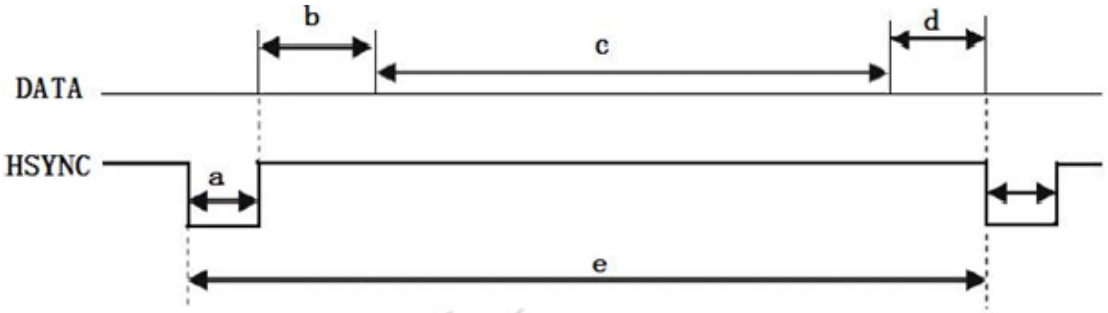
由电路图可以看出，我们的 VGA 并没有特殊的外部芯片，也就是说，我们唯一要关注的可能就是它的显示原理和时序了。那么接下来我们具体来看一下 VGA 的扫描原理是什么。

VGA 扫描方式

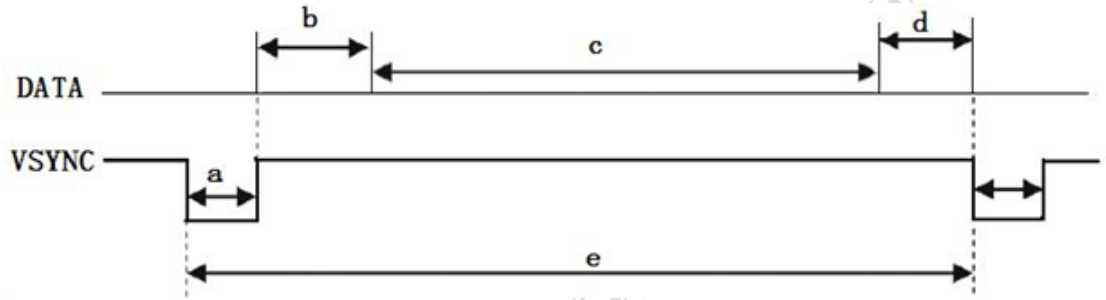
显示器扫描方式分为逐行扫描和隔行扫描：逐行扫描是扫描从屏幕左上角一点开始，从左向右逐点扫描，每扫描完一行，电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，开始下一帧。隔行扫描是指电子束扫描时每隔一行扫一线，扫完一屏后再返回来扫描剩下的线，隔行扫描的显示器闪烁快速，可能会使使用者眼睛疲劳（本实验采用逐行扫描的方式）

扫描原理清楚以后，紧接着大家再来看看 VGA 的行、列同步时序

列同步时序



行同步时序



VGA 中定义行时序和列时序都需要同步脉冲（a 段），显示后沿（b 段）、显示时序段（c 段）和显示前沿（d 段）四部分。VGA 工业标准显示模式要求：行同步、列同步都为负极性，即同步脉冲要求是负脉冲。

由 VGA 行时序可知：每一行都有一个负极性行同步脉冲（a 段），是数据行的结束标志，同时也是下一行的开始标志。在同步脉冲之后为显示后沿（b 段），在显示时序段（c 段）显示器为亮的过程，RGB 数据驱动一行上的每一个像素点，从而显示一行。在一行的最后为显示前沿（d 段）。在显示时间段之外没有图像投射到屏幕，而是插入消隐信号。同步脉冲、显示后沿和显示前沿都是在行消隐间隔内，当消隐有效时，RGB 信号无效，屏幕不显示数据。

VGA 的列时序与行时序分析基本一致。

VGA 也有许多的显示标准，接下来我们通过表格来一探究竟。

显示模式	时钟 (MHz)	列时序（列数）					行时序（行数）				
		a	b	c	d	e	a	b	c	d	e
640*480*60	25.175	96	48	640	16	800	2	33	480	10	525
640*480*75	31.5	64	120	640	16	840	3	16	480	1	500
800*600*60	40.0	128	88	800	40	1056	4	23	600	1	628
800*600*75	49.5	80	160	800	16	1056	3	21	600	1	625
1024*768*60	65	136	160	1024	24	1344	6	29	768	3	806
1024*768*75	78.8	176	176	1024	16	1312	3	28	768	1	800
1280*1024*60	108.0	112	248	1280	48	1688	3	38	1024	1	1066
1280*800*60	83.64	136	200	1280	64	1680	3	24	800	1	828
1440*900*60	106.47	152	232	1440	80	1904	3	28	900	1	932

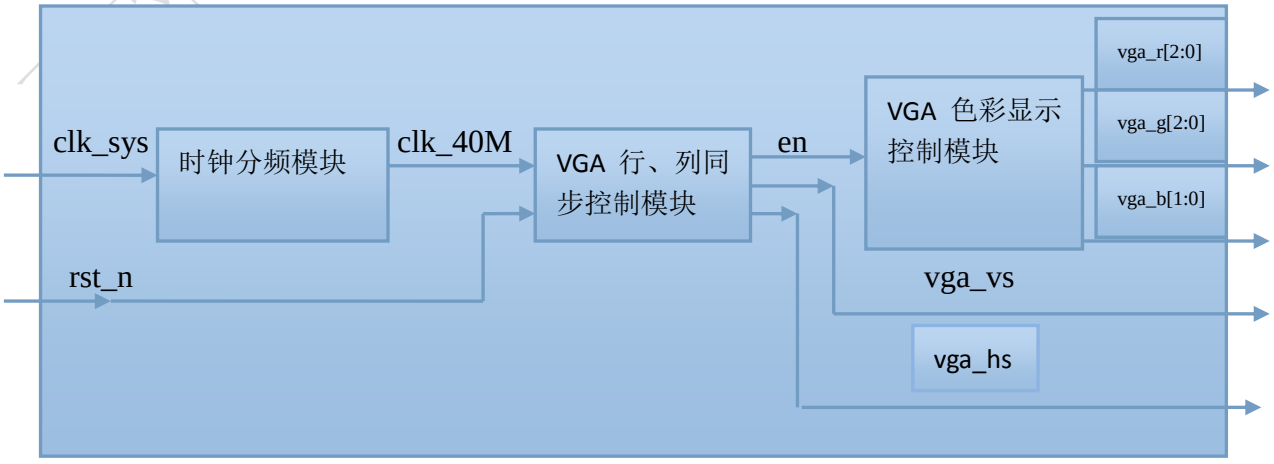
以本实验的显示标准 800*600*60Hz 为例。(800 为列数，600 为行数，60Hz 为刷新一屏的频率)

行时序：屏幕对应的行数为 628（a+b+c+d=e 段），其中 600（c 段）为显示行；每行均有行同步信号（a 段），为 4 个行周期的低电平；

列时序：每个显示行包括 1056 列（a+b+c+d=e 段），其中 800（c 段）为有效显示区，每一行有一个行同步信号（a 段），为 128 个行周期的低电平。

扫描时钟频率：40MHZ

原理清楚以后，接下来我们设计系统模块图如下：



模块说明：

(1)时钟分频模块

我们开发板上使用的晶振为 50MHZ，由于我们的显示标准为 800*600*60HZ，所以我们需要分频输出 40MHZ 的系统时钟。时钟分频模块，我们可以通过调用锁相环来实现。

(2)VGA 行列同步控制模块

VGA 显示标准需要设定行列同步信号，标定出有效显示区域，这也是整个 VGA 驱动模块的核心部分

(3)VGA 色彩显示控制模块

在图像有效显示区域内，输出控制颜色的 r、g、b 信号

接下来我们学习每个功能模块的具体代码实现，由于分频模块我们采用的是锁相环，而之前的章节中，我们对锁相环的使用有专门的论述，所以此处我们不再赘述。

VGA 行列同步控制模块具体代码如下：

```
module vga(     //显示分辨率为800 * 600 *60 MHZ
//端口信号：模块的输入输出接口
    input       clk,    //连接至分频时钟，为40MHz
    input       rst_n,  //低电平复位

    output reg vga_hs,//VGA列同步信号
    output reg  vga_vs,//VGA行同步信号
    output          en      //显示有效区域的使能信号
    );
//-----------------VGA时序-----------------
//        显示模式        时钟
//        800*600@60   40Mhz
//        同步 后沿 有效 前沿 周期
//hs 128  88        800  40        1056
//vs 4        23        600  1         628
//VGA显示相应参数
parameter       hy_all = 11'd1056,   //列时序
                hy_a   = 11'd128,
                hy_b   = 11'd88,
                hy_c   = 11'd800,
                hy_d   = 11'd40,

                vy_all = 11'd628,    //行时序
                vy_a   = 11'd4,
                vy_b   = 11'd23,
                vy_c   = 11'd600,
                vy_d   = 11'd1;
```

```
//用计数器限定VGA显示相应区域
reg [10:0] cnt_h;//列计数器
reg [10:0] cnt_v;//行计数器

//--------------------------列计数-------------------------
always@(posedge clk or negedge rst_n)
    if(!rst_n)
        cnt_h   <= 11'd0;//列计数器复位
    else    if(cnt_h == (hy_all-1))//所有列扫描完毕
        cnt_h   <= 11'd0;//列计数器清零
    else
        cnt_h <= cnt_h + 1'b1;//列计数器累加

//--------------------------行计数-------------------------
always@(posedge clk or negedge rst_n)//在一列计数完之后将行加1
    if(!rst_n)
        cnt_v   <= 11'd0;//行计数器复位
    else    if(cnt_v == (vy_all-1))//所有行扫描完毕
        cnt_v   <= 11'd0;//行计数器清零
    else    if(cnt_h ==(hy_all-1))//所有列扫描完毕
        cnt_v   <= cnt_v + 1'b1;//行计数器加一
```

```
always@(posedge clk or negedge rst_n)
   if(!rst_n)
      vga_hs  <= 1'b1;//复位时置列同步信号为高电平
   else if(cnt_h ==0) //列开始扫描
      vga_hs  <= 1'b0;//置列同步信号为低电平
   else if(cnt_h == hy_a)//保持hy_a个时钟周期
      vga_hs  <= 1'b1;//置列同步信号为高电平

//--------------------限定行同步信号--------------------
always@(posedge clk or negedge rst_n)
   if(!rst_n)
     vga_vs   <= 1'b1;//复位时置行同步信号为高电平
   else if(cnt_v ==0) //行开始扫描
      vga_vs  <= 1'b0;//置行同步信号为低电平
   else   if(cnt_v == vy_a)//保持vy_a个时钟周期
      vga_vs  <= 1'b1;//置行同步信号为高电平

//--------------------限定显示有效区域，设定使能信号--------------------
wire [12:0] en1;//列有效标志
wire [12:0] en2;//行有效标志

assign en1 = (cnt_h >= hy_a + hy_b && cnt_h <= hy_a + hy_b + hy_c)?(cnt_h-hy_a-hy_b):11'd0;//列有效区域
assign en2 = (cnt_v >= vy_a + vy_b && cnt_v <= vy_a + vy_b + vy_c)?(cnt_v-vy_a-vy_b):11'd0;//行有效区域
assign en = (en1 > 0 && en2 > 0 )?1'b1:1'b0;//行、列共同有效区域800*600

endmodule
```

VGA 色彩显示控制模块具体代码如下：

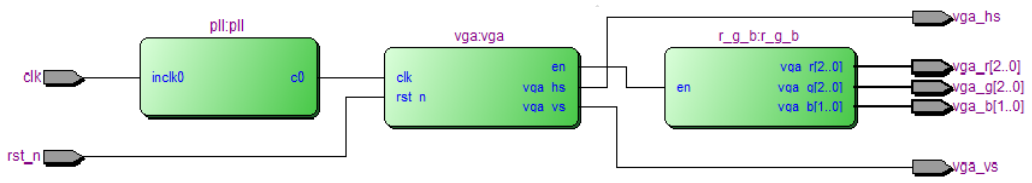
```
/////////////////////////////////////////////////////////////////////
*    Module Name        :    r_g_b
*    Engineer           :    梦翼师兄
*    Target Device      :    EP4CE10F17C8N
*    Tool versions      :    Quartus II 13.0 SP1
*    Create Date        :    2014-12-31
*    Revision           :    v1.0
*    Description        :    VGA色彩输出信号控制器
*    Website            :    www.zxopen.com
*    Forum              :    www.fpgaw.com
/////////////////////////////////////////////////////////////////////*/
module r_g_b(     //颜色控制模块
//端口信号：模块的输入输出接口
    input          en,   //使能信号

    output [2:0]   vga_r, //红色(3位：根据数值的变化，控制颜色的深浅)
    output [2:0]   vga_g, //绿色(3位：根据数值的变化，控制颜色的深浅)
    output [1:0]   vga_b  //蓝色(2位：根据数值的变化，控制颜色的深浅)
    );

//-------------------在使能区域显示相应的颜色-------------------
assign vga_r = en?3'b111:3'b000;//在使能信号下输出红色
assign vga_g = 3'b000;   //无绿色输出
assign vga_b = 2'b00;    //无蓝色输出
endmodule
```

编写完成各子模块以后，为了将所有模块连接起来，我们需要建立顶层文件如下：

```
module top(     //顶层模块：将各个模块组合
 //外部接口
    input           clk,   //系统时钟50MHz
    input           rst_n,//低电平复位
    output          vga_vs,//VGA行同步信号
    output          vga_hs,//VGA列同步信号
    output [2:0]    vga_r,//红色输出信号
    output [2:0]    vga_g,//绿色输出信号
    output [1:0]    vga_b  //蓝色输出信号
    );
//内部信号：模块内部的接口信号，比如模块pll的输出信号c0，通过内部信号clk_40与模块vga的输入信号clk相连
wire clk_40;
wire en;

//模块例化
pll pll(      //分频时钟，用锁相环产生，在显示标准800*600*60HZ中，时钟指定为40MHz
    .inclk0(clk),
    .c0(clk_40)
    );
```

```
vga vga(      //接入分频时钟，限定显示的有效区域
    .clk(clk_40),
    .rst_n(rst_n),
    .vga_vs(vga_vs),
    .vga_hs(vga_hs),
    .en(en)
   );

r_g_b r_g_b(    //在有效区域内，显示相应颜色
    .en(en),
    .vga_r(vga_r),
    .vga_g(vga_g),
    .vga_b(vga_b)
    );

endmodule
```

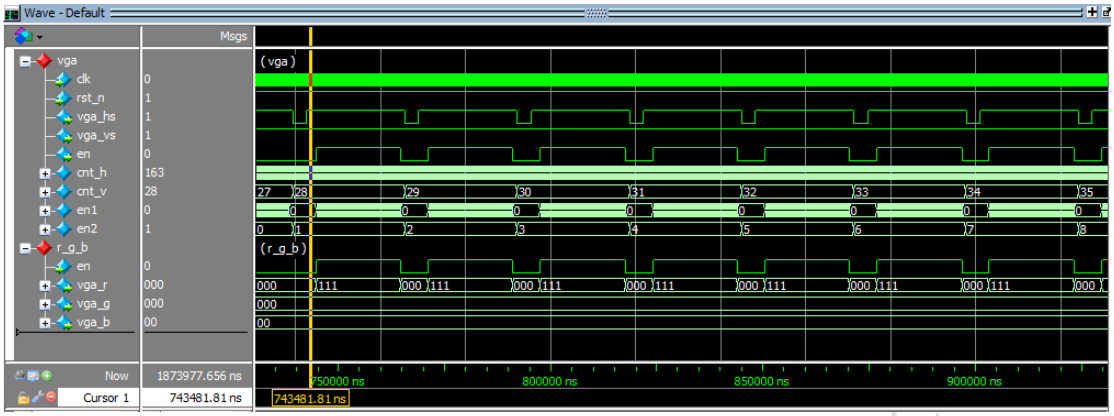
综合编译以后，我们可以查看 RTL 视图，查看电路综合结果和预想是否一致，调用 RTL 视图如下：



由此可以，电路模块综合结果和我们预先设定相同。接下来我们编写测试代码如下，用来验证我们设计的正确性

```
`timescale 1 ns/ 1 ns   //设置仿真时间单位与精度分别为1ns/1ns
module test;   //测试模块：主要是将激励信号赋相应的值，仿真之后观察波形，验证与实际功能是否一样
//端口信号定义，激励信号为reg型
reg         clk;
reg         rst_n;
wire [2:0]  vga_r;
wire [2:0]  vga_g;
wire [1:0]  vga_b;
wire        vga_hs;
wire        vga_vs;
//初始化激励，以及给相应激励赋值
initial
    begin
        clk = 0;
        rst_n = 0;   //在复位阶段，将激励赋初值
        #200  rst_n = 1; //在延时200ns后将复位信号置为1
    end
always  #10 clk = ~clk;     //时钟的表示，即每隔10ns翻转一次，一个周期的时间即为20ns，时钟为1/20ns = 50MHZ
//模块例化
top top(
            .clk(clk),
            .rst_n(rst_n),
            .vga_hs(vga_hs),
            .vga_vs(vga_vs),
            .vga_r(vga_r),
            .vga_g(vga_g),
            .vga_b(vga_b)
        );
endmodule
```

查看仿真波形



如图可以看出，当 en 有效时，vga_r 输出 3’b111，说明设计正确。

将代码下载到开发板，运行效果图如下：