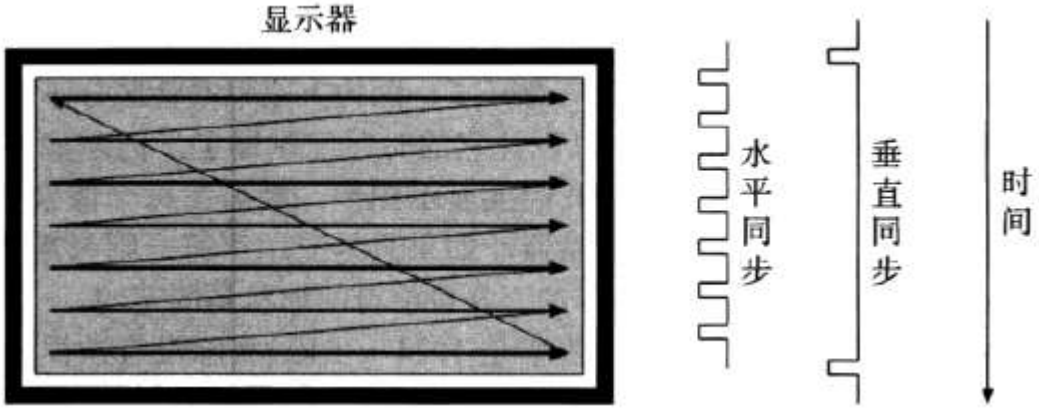
实验四　VGA

VGA(Video Graphics Aray)是 IBM 公司制定的一种视频数据的传输标准。它的接口信号主要有 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。水平同步和垂直同步又称行(Line)同步和帧(Framc)同步。这些都是模拟(Analogue)信号，用于连接诸如 CRT 和 LCD 等显示器。以下简称红、绿、蓝为 RGB。

1 接口控制器设计

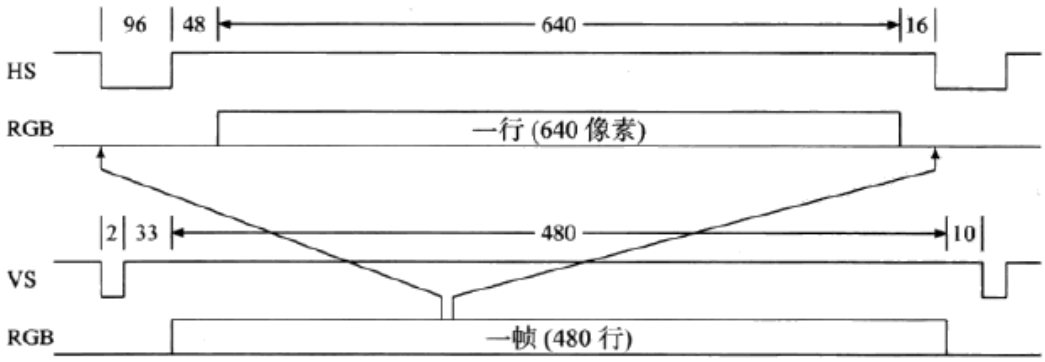
图像的显示是以像素 (Pixel)为单位从左上角开始一行一行进行的。行同步信号是负脉冲。负脉冲来时要由 RGB 送出在当前行显示的像素。下一个负脉冲用来显示下一行。当整个屏幕(一帧)显示一遍后，由帧同步信号送出一个负脉冲，又从左上角开始显示，如下图所示。



显示器(CRT 或 LCD)的分辨率(Resolution)是指屏幕每行有多少个像素及每帧有多少行。标准的 VGA 的分辨率是 640x480，即每行有 640 个像素，每列有 480 个像素。高分辨率的显示器有 XGA(1024x768)、SXGA(1280x1024)、UXGA(1600x1200)、OXGA(2048x1536)、OSXGA(2560x2048)以及宽屏的 WXGA(1366x768)、WSXGA(1680x1050)和 WUXGA(1920x1200)等。从视觉效果考虑，每秒显示的帧数不应小于 24。分辨率越高，每秒送出的像素数也应越多。由于传送速率的限制，有些特高分辨率的显示器每秒也只能刷新十几帧。这对像作者一样眼神差一些的人来讲也是完全没有问题的。

并不是所有的时间都在传送像素。由于 CRT 的电子束回扫(从一行的尾到下一行的头或

从一帧的尾到下一帧的头)也需要时间，这时必须要让 RGB 送出电压为 0 的信号(黑色)。下图所示的是 VGA 的同步信号长度以及何时才能送出像素。图中水平同步的数字是像素数，垂直同步的数字是行数。



如果负责送出像素的时钟的频率是 25.2MHz，我们可以计算出每秒可以刷新多少帧:$25.2x10^6/[(96+48+640+16)x(2+33+480+10)]= 60$。

像素的颜色由模拟信号 RGB 的电压决定，而这三个信号是 D/A 转换器的输出，因此原始的表示红绿蓝的三个数据的位数就决定了能够显示的颜色的数量。所谓的真彩色(True Color)是指每个数据都有 8 位，即用 3x8=24 位来表示一个像素，其颜色的数量就有 2^{24}=16777216 种。如果你看到产品广告说有 1677 万种色彩，就是这个意思。也有用 27 位的，再多就没有意义了，因为人眼根本就区分不出来。

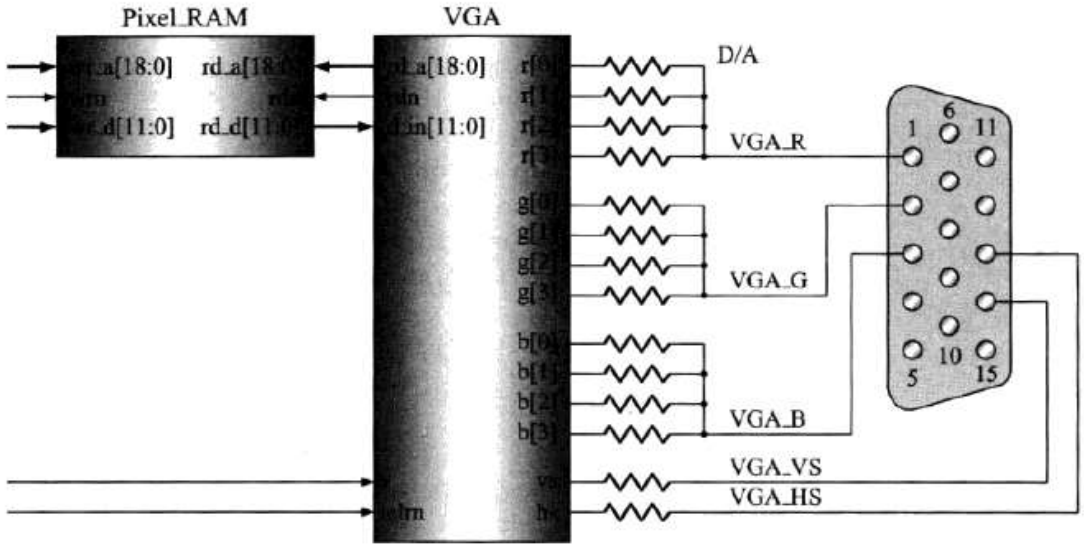
搞懂了 VGA 的时序，我们就可以设计它的时序控制器了。但是如何产生 RGB 相对应的数据还是有些讲究的。如果我们要显示图像，则需为每个像素指定一个颜色，这就需要有一个比较大的视频存储器。如果只是显示字符，则只需要较小的视频缓冲区，因为一个字符的颜色是一样的，并且字符的形状是固定的。

为了计算上的方便，我们假设分辨率是 1024x512(实际没有这样的产品)。如果要用真彩色来显示图像(比如照片)，那么视频存储器的容量需要多大呢?很简单，答案是 512Kx24 位，即 1.5MB。需要 19 位地址，每次读出 24 位。如果不要求显示图像而只是作为字符显示器来用，就不需要这么大的存储器了。假设有 128 种不同的字符，每个字符也用真彩色来显示，一个字符用 16x8 的点阵来表示(16 行 8 列)，那么需要多大容量的存储器呢?停一下先算算看，搞清楚了就会设计电路了。

解答如下。一个字符可用 7 位来表示(10g128=7)，另由一个 24 位的数据来表示这个字符的颜色。一帧有 512/16=32 行，每行有 1024/8=128 个字符。因此，显示缓冲区需要有

(128x32)x(7+24)=4Kx31 位，即 15.5KB。另外还要有一个字模产生器(字模表)，它的容量是 128x16x8=2Kx8 位，即 2KB。一共需要 17.5KB，大约是 1.5MB 的 1%(注意:1M=1024K)。如果全屏字符只用一种颜色，比如黑底白字，则每个字符的 24 位表示颜色的数据也可以省去。设计时需要注意字模点阵送出的次序:16 行像素显示一行字符。当然，如果已经有了视频存储器，则可以把字符点阵送入视频存储器。

以下描述一种简单的用于图像显示的 VGA 的时序控制器，见下图。VGA 模块产生视频存储器 PixelRAM 的读地址，从视频存储器中读出 12 位的像素数据(每种颜色用 4 位表示，共可表示 4096 种颜色)。读来的数据 d_in[11:0]经锁存后由 r[3:0]、g[3:0]和 b[3:0]送出。D/A 转换器由简单的电阻阵列实现。注意接到每种颜色高位数据的电阻应有较低的阻抗，然后依次递增，接到最低位的电阻的阻抗值最高。阻抗值的设定应使数字信号为最大值时输出的电压为 0.7V，最小值(0)时输出的电压为 0V。作为参考，四个电阻的阻抗值可分别设为 510Ω、1kΩ、2kΩ 和 4kΩ 水平同步信号 hs 和垂直同步信号 vs 也经电阻(比如 82.5Ω)连到显示器接口。



VGA 的分辨率是 640x480，为了简化设计及提高读地址产生的速度，我们只是把行列的像素地址拼接在一起。行地址(row)记录行号，因为一有 480 行，因此需要 9 位地址;列地址(col)记录列号，因为一有 640 列，因此需要 10 位地址。总的地址位数为 19。注意，这种做法会导致视频存储器的浪费，因为 19 位地址的存储器可以存放 1024x512 个像素。另外的一种产生地址的做法是行地址乘以 640 再加上列地址，但由于需要一个乘法器，导致电路成本的增加及地址产生的速度变慢。以下是 VGA 时序控制器的 VerilgHDL 代码。输入时钟信号 clk 为 50MHz，经二分频后的时钟信号 vga-clk 为 25MHz;clmn 是低电平有效的复位信

号;din 是从视频存储器读来的 12 位像素数据;rda 是视频存储器的 19 位地址;rdn 是低电平有效的读信号;r、g、b 分别是 4 位的红、绿、蓝信号;hs 和 vs 分别是水平同步和垂直同步信号;h_count 和 v_count 是内部信号，送出来仅供测试用，它们分别是水平方向和垂直方向的计数器。阅读代码时请参看上面的时序图。

```
// VGA signal generator, 640 x 480, 25MHz
module vga (clk,clrn,d_in,rd_a,rdn,r,g,b,hs,vs,h_count,v_count);
    input  clk; // 50MHz
    input  clrn;

input  [11:0] d_in; // rrrr_gggg_bbbb, pixel
output [18:0] rd_a; // pixel RAM addr, 640 (1024) x 480 (512)
output rdn; // read pixel RAM (active_low)
output [3:0] r, g, b; // red, green, blue, 4-bit for each
output hs, vs; // horizontal and vertical synchronization
// for test
output [9:0] h_count; // VGA horizontal counter (0-799)
output [9:0] v_count; // VGA vertical counter (0-524)
// internal signals
reg vga_clk; // 25MHz
reg [9:0] h_count; // VGA horizontal counter (0-799): pixel
// __    ___________________________    _______________
//   |__|                           |__|
//   |96|   ______________________   |96|   ____________
// _____|       1 line         |_____| (next line)
//   |96|48|-------- 640 ---------|16|96|48|
//   | 144 |                     |  |
//   |------------ 784 -----------|  |
//   |-------------- 800 ------------|

reg [9:0] v_count; // VGA vertical counter (0-524): lines
// __    ___________________________    _______________
//   |__|                           |__|
//   | 2|   ______________________   | 2|   ____________
// _____|       1 frame        |_____| (next frame)
//   | 2|33|-------- 480 ---------|10| 2|33|
//   |  35 |                     |  |
//   |------------ 515 -----------|  |
//   |-------------- 525 ------------|
reg [11:0] data_reg; // latched rrrr_gggg_bbbb, pixel
reg video_out; // VGA signal enable
// vga_clk: 25MHz
always @ (posedge clk or negedge clrn) begin
    if (clrn == 0) begin
        vga_clk <= 1'b1;
    end else begin
        vga_clk <= ~vga_clk;
    end
end
// h_count: VGA horizontal counter (0-799)
always @ (posedge vga_clk or negedge clrn) begin
    if (clrn == 0) begin
        h_count <= 10'h0;
    end else if (h_count == 10'd799) begin
        h_count <= 10'h0;
```

```
        end else begin
            h_count <= h_count + 10'h1;
        end
    end
    // v_count: VGA vertical counter (0-524)
    always @ (posedge vga_clk or negedge clrn) begin
        if (clrn == 0) begin
            v_count <= 10'h0;
        end else if (h_count == 10'd799) begin
            if (v_count == 10'd524) begin
                v_count <= 10'h0;
            end else begin
                v_count <= v_count + 10'h1;
            end
        end
    end
    // video_out: VGA signal enable
    always @ (posedge vga_clk or negedge clrn) begin
        if (clrn == 0) begin
             video_out <=  1'b0;
             data_reg  <= 12'h0;
        end else begin
             video_out <= ~rdn;
             data_reg  <= d_in; // pixel RAM access time: < 40ns
        end
    end
    // rdn: one vga_clk cycle ahead due to video_out delay
    assign rdn = ~(((h_count>=10'd143) && (h_count<10'd783)) &&
                   ((v_count>=10'd35)  && (v_count<10'd515)));
    wire [9:0] row = v_count - 10'd35;
    wire [9:0] col = h_count - 10'd143;
    assign rd_a = {row[8:0],col};
    assign hs = (h_count >= 10'd96); // horizontal synchronization
    assign vs = (v_count >= 10'd2);  // vertical synchronization
    assign r = (video_out) ? data_reg[11:8] : 4'h0; // 4-bit red
    assign g = (video_out) ? data_reg[07:4] : 4'h0; // 4-bit green
    assign b = (video_out) ? data_reg[03:0] : 4'h0; // 4-bit blue
endmodule
```

由于读视频存储器需要时间，因此我们先读出像素，将其锁存后再显示。如果读边显示的话，可能会造成屏幕显示不稳定。但如果视频存储器本身就是同步的渝出用时钟锁存)，则没必要再锁存了。本例中使用 vga-cIk(25MHz)的一个时钟周明读视频存储器，因此视频存储器的访问周期时间不应大于 40ns。

2 VGA 显示键盘输入字符