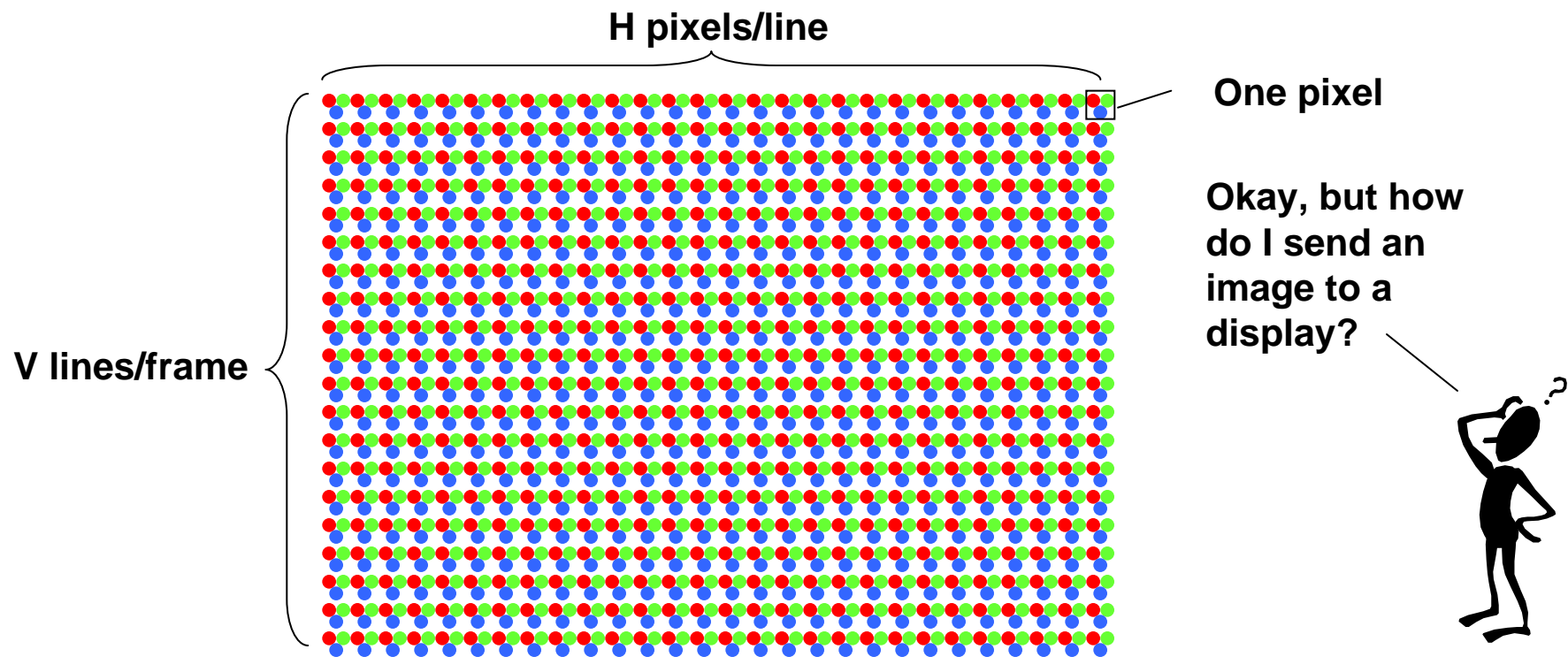


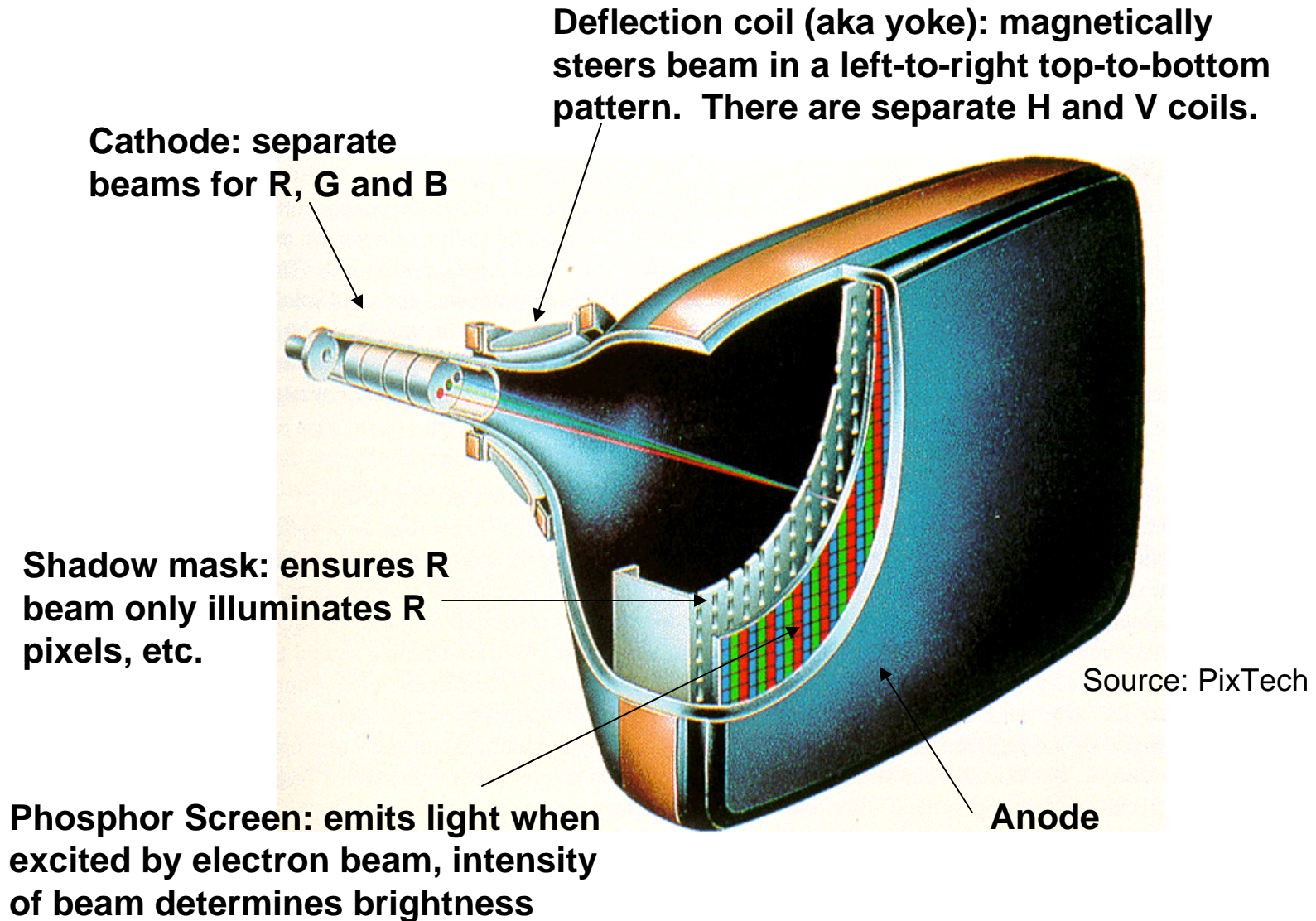
# Generalized Video Display

Think of a color video display as a 2D grid of picture elements (pixels). Each pixel is made up of red, green and blue (RGB) emitters. The relative intensities of RGB determine the apparent color of a particular pixel.

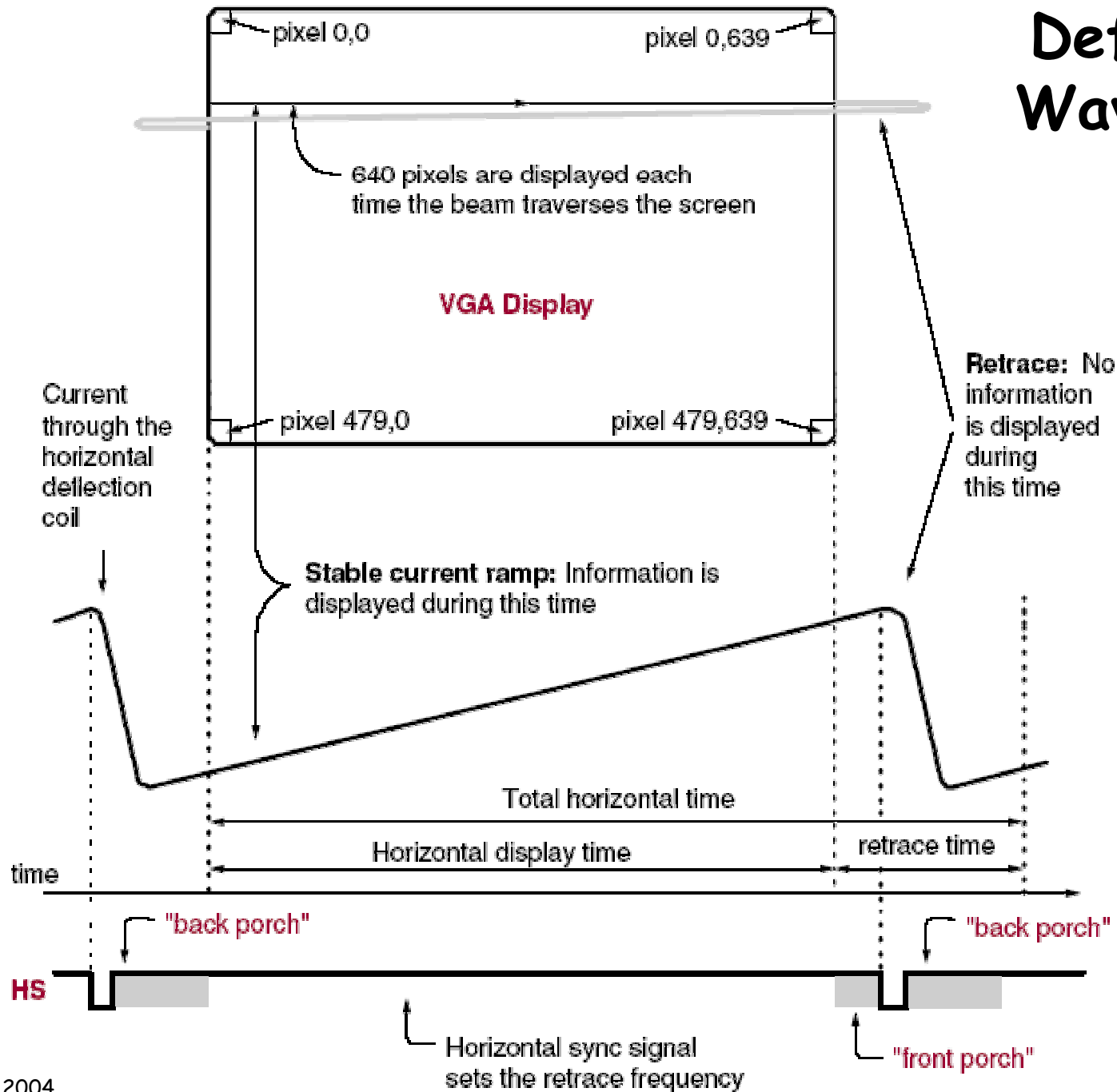


Traditionally  $H/V = 4/3$  or with the advent of high-def  $16/9$ .  
Lots of choices for  $H, V$  and display technologies (CRT, LCD, ...)

# Background: Cathode Ray Tubes

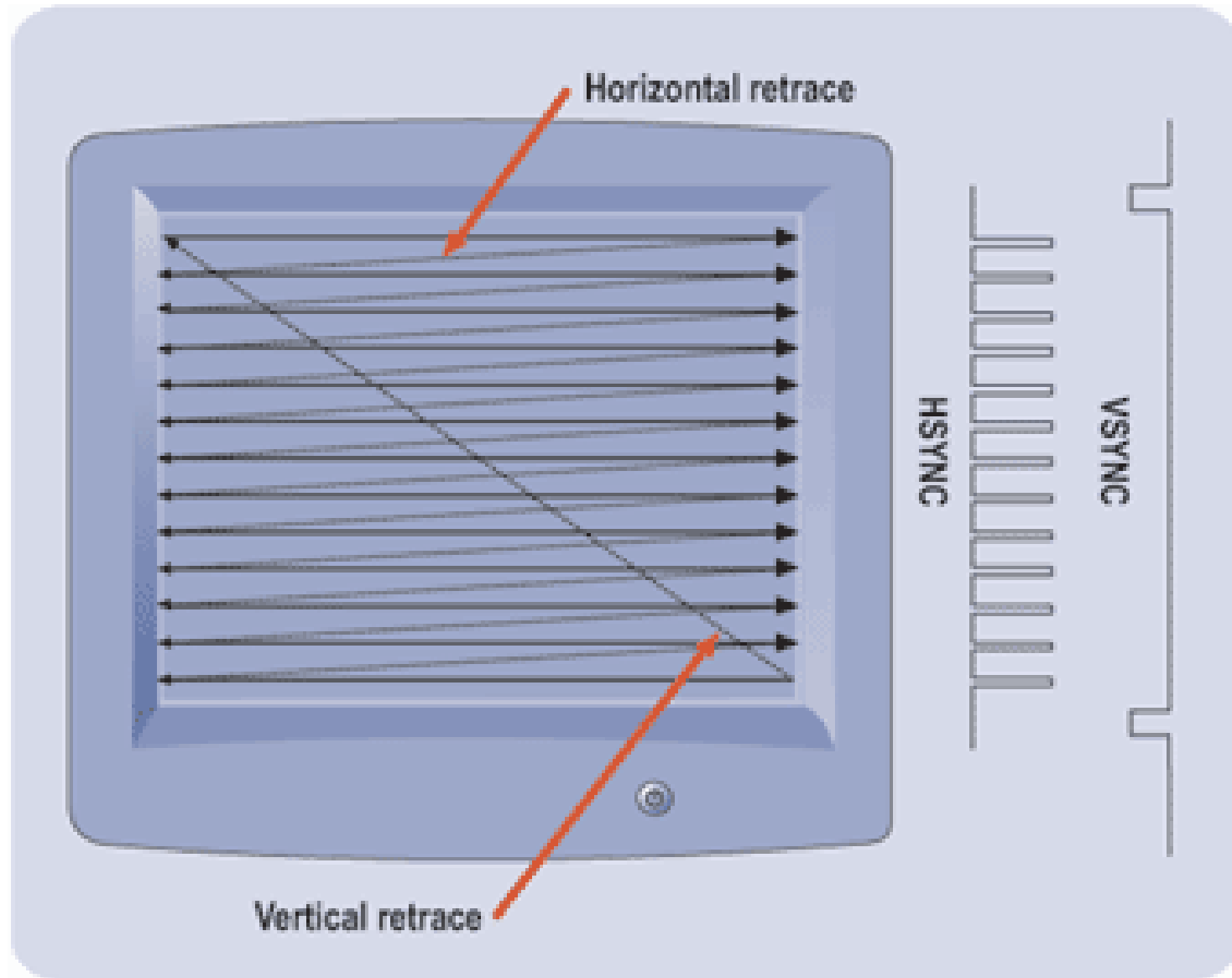


# Deflection Waveforms



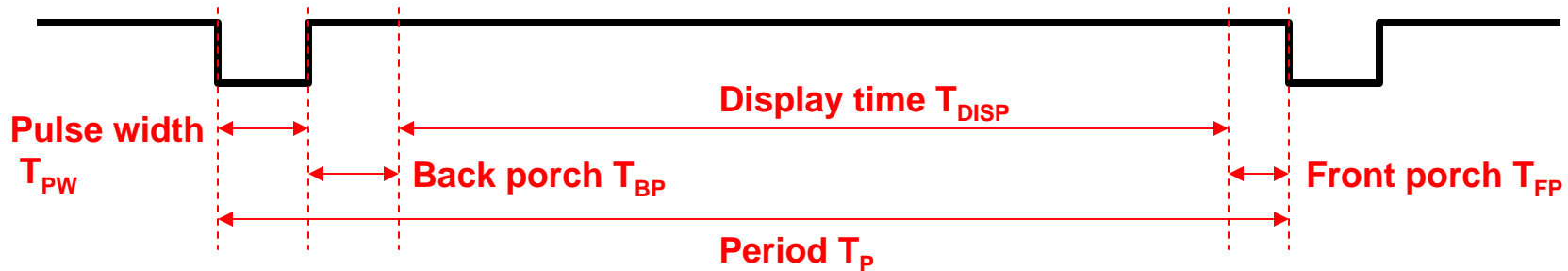
Source: Xilinx Spartan-3 Starter Kit Board User Guide

# Sync Signals (HS and VS)



# Sync Signal Timing

The most common ways to send an image to a video display (even displays that don't use deflection coils, eg, LCDs) require you to generate two sync signals: one for the horizontal dimension (HS) and one for the vertical dimension (VS).



<i>Format</i>		<i>CLK</i>	<i>P</i>	<i>PW</i>	<i>BP</i>	<i>DISP</i>	<i>FP</i>
VGA	HS (pixels)	25Mhz	794	95	47	640	13
	VS (lines)	--	528	2	33	480	13
XGA	HS (pixels)	65Mhz	1344	136	160	1024	24
	VS (lines)	--	806	6	23	768	9

# Interlace

Non-interlaced (aka progressive) scanning:

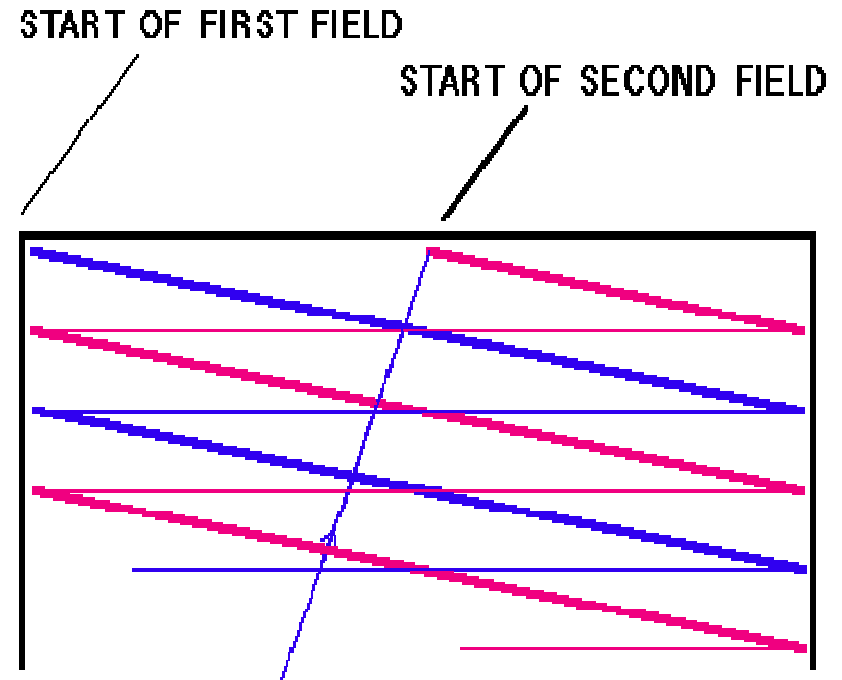
- VS period is a multiple of HS period
- Frame rate  $\geq$  60Hz to avoid flicker

Interlaced scanning:

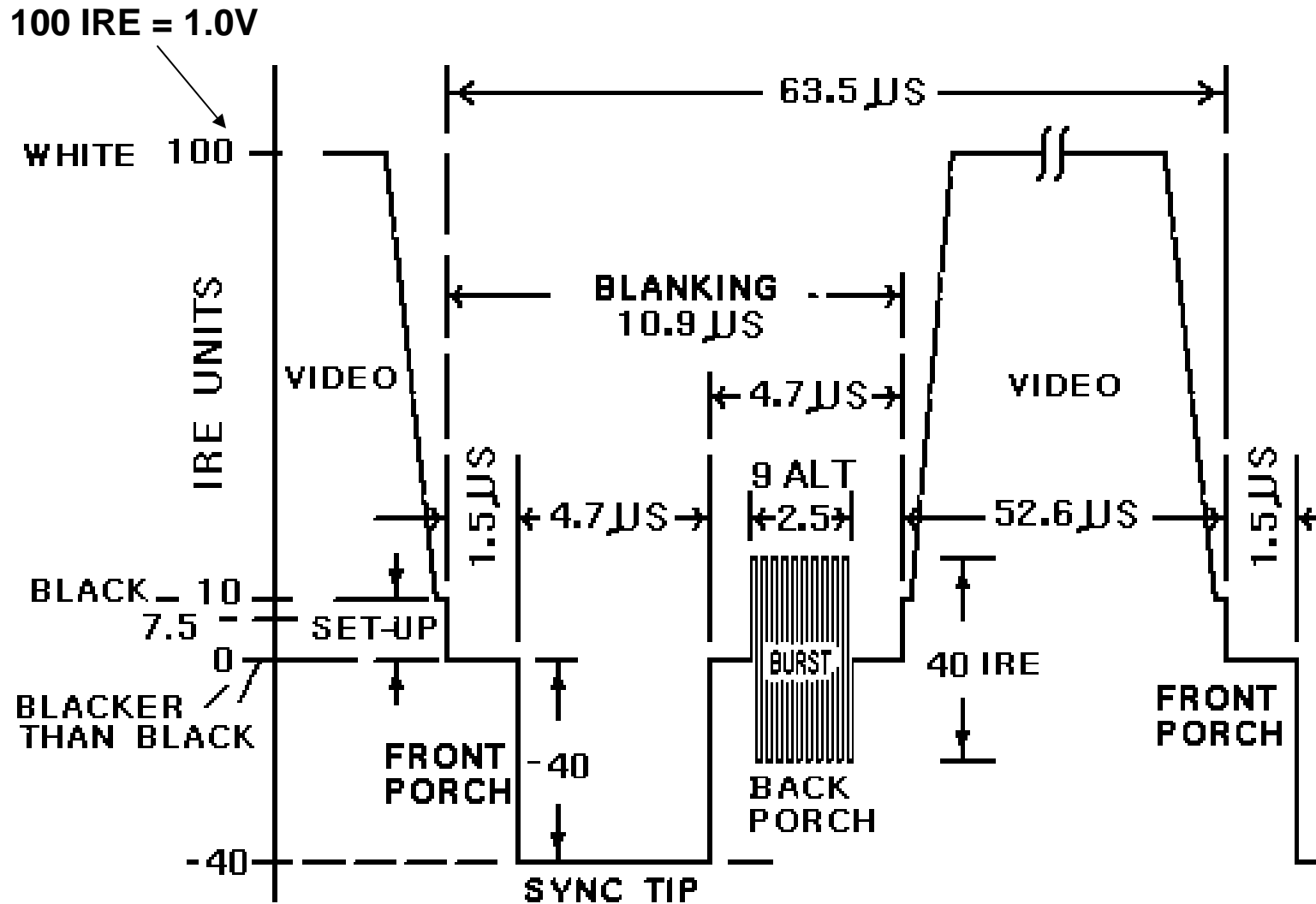
VS period is *not* a multiple of HS period, so successive vertical scan are offset relative to horizontal scan, so vertical position of scan lines varies from frame to frame.

NTSC example:

- 525 total scan lines (480 displayed)
- 2 fields of 262.5 scan lines (240 displayed).
- Field rate is 60Hz, frame rate = 30Hz



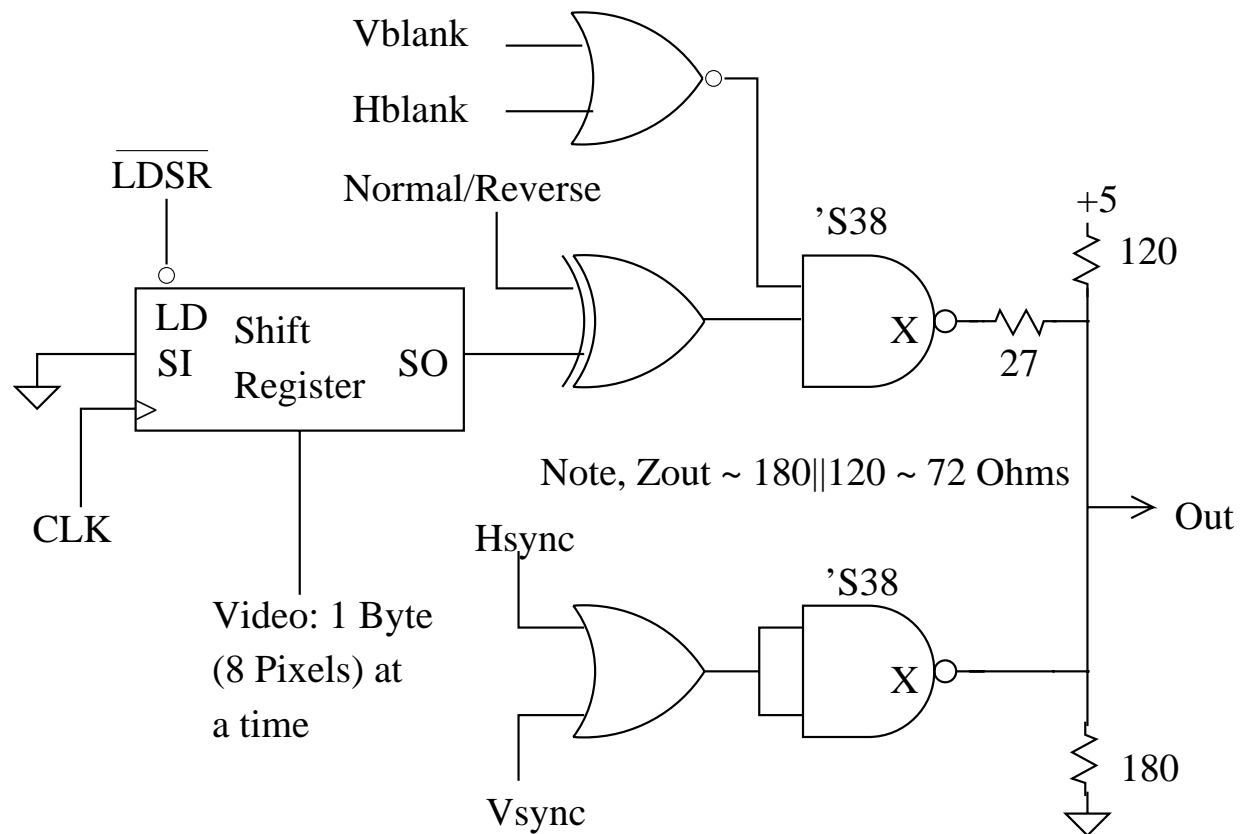
# Encoding: NTSC



Source: <http://www.ntsc-tv.com>

# Generating B/W NTSC Signals

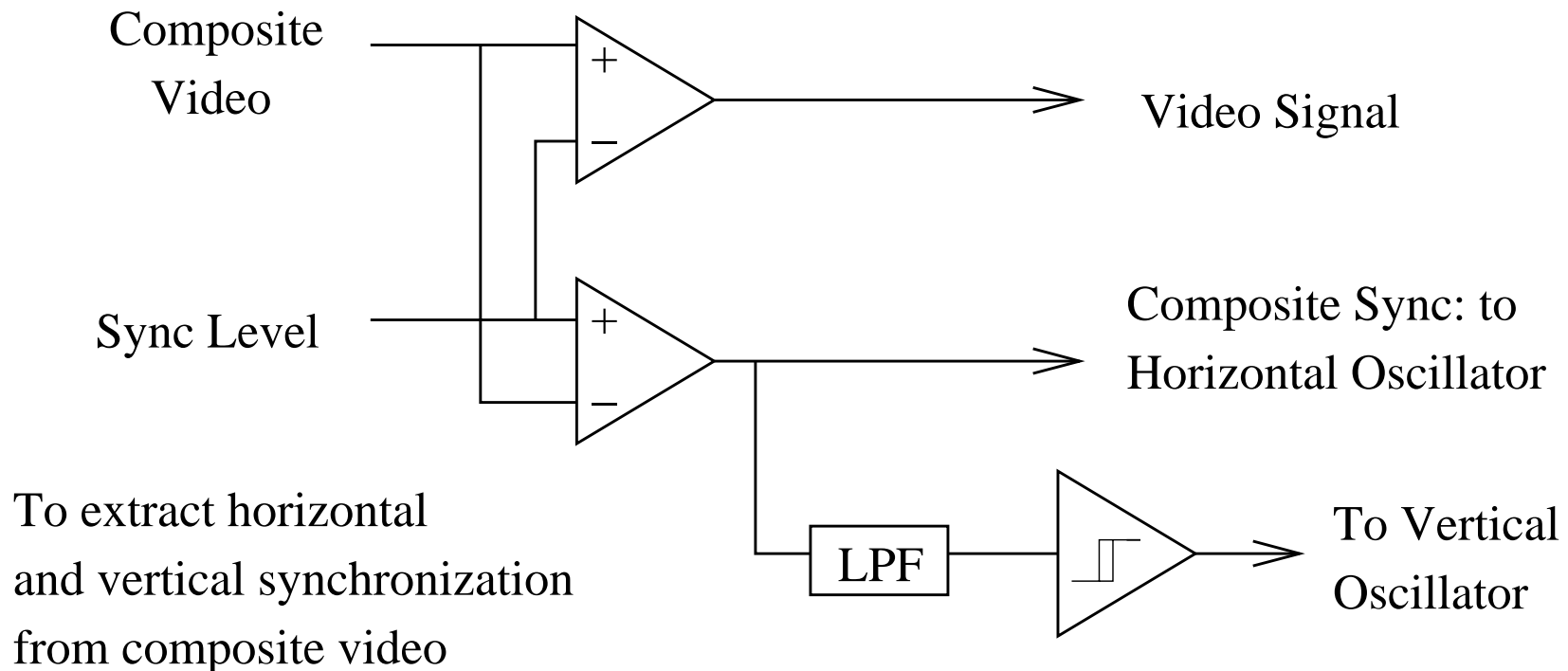
- Assume one bit per pixel and provide for reverse video.
- This is a simple 'D/A' to generate monochrome composite video.
  - The 'S38 is an open collector part so the voltages are determined by the resistor network. The output resistance is  $\sim 75$  ohms.





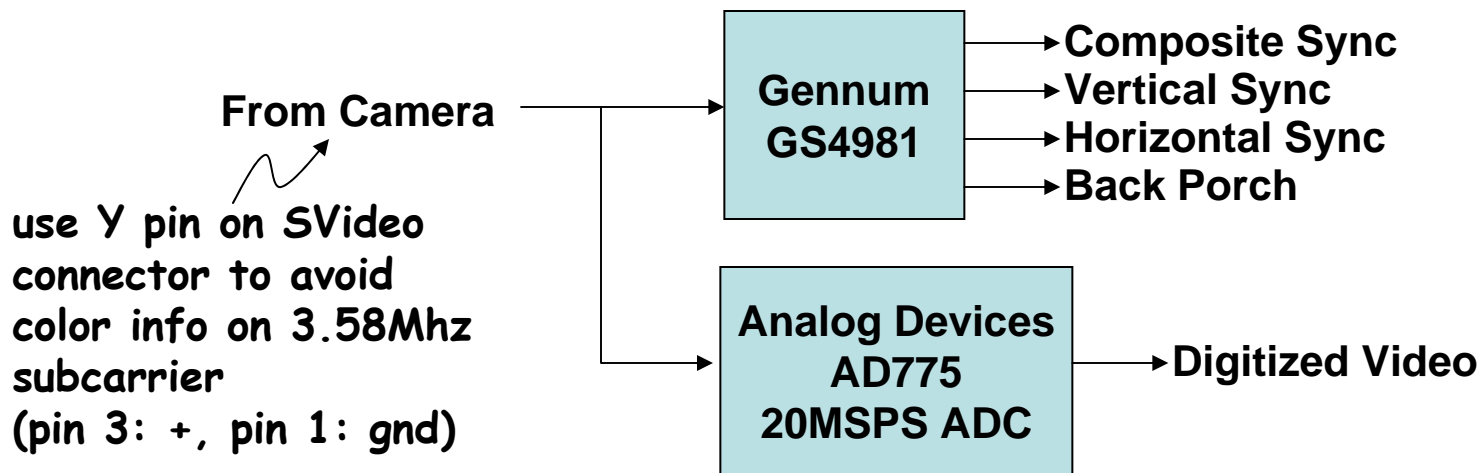
# (Conceptual) Recovery of Signals

- Composite video has picture data and both syncs.
  - Picture data (video) is above the sync level.
  - Simple comparators extract video and composite sync.
- Composite sync is fed directly to the horizontal oscillator.
- A low-pass filter is used to separate the vertical sync.
  - The edges of the low-passed vertical sync are squared up by a Schmidt trigger.



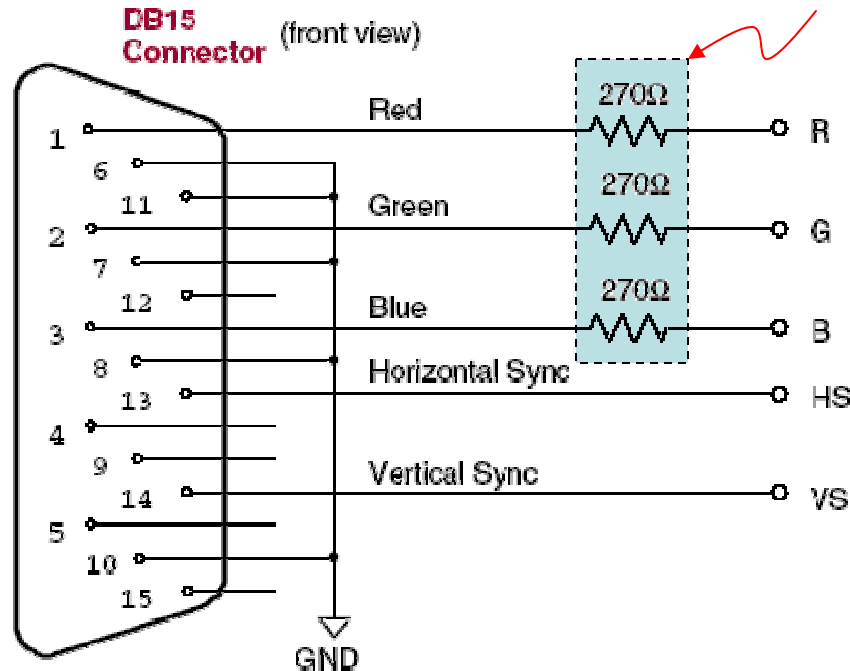
# Video Capture

- A sync separator is used to recover sync from a composite video signal.
  - GS4981 generates composite sync from video. It also generates separated sync signals.
- The sync separator is not easy to implement in an HDL as its input is an analog signal.
- However, your pixel clock must be synchronized with the recovered horizontal sync.
  - If you do this synchronization with the pixel clock signal directly, then the pixel clock used will “crawl” a whole pixel time.
  - It is better to use a faster clock, say 4 times faster, to do the synchronization and then the “crawl” will only be  $\frac{1}{4}$  of a pixel time (distance).



# Encoding: VGA-compatible

## Poor man's Video DAC



Your circuitry should produce TTL-level signals (3.3V high level)

HS, VS are active-low signals.

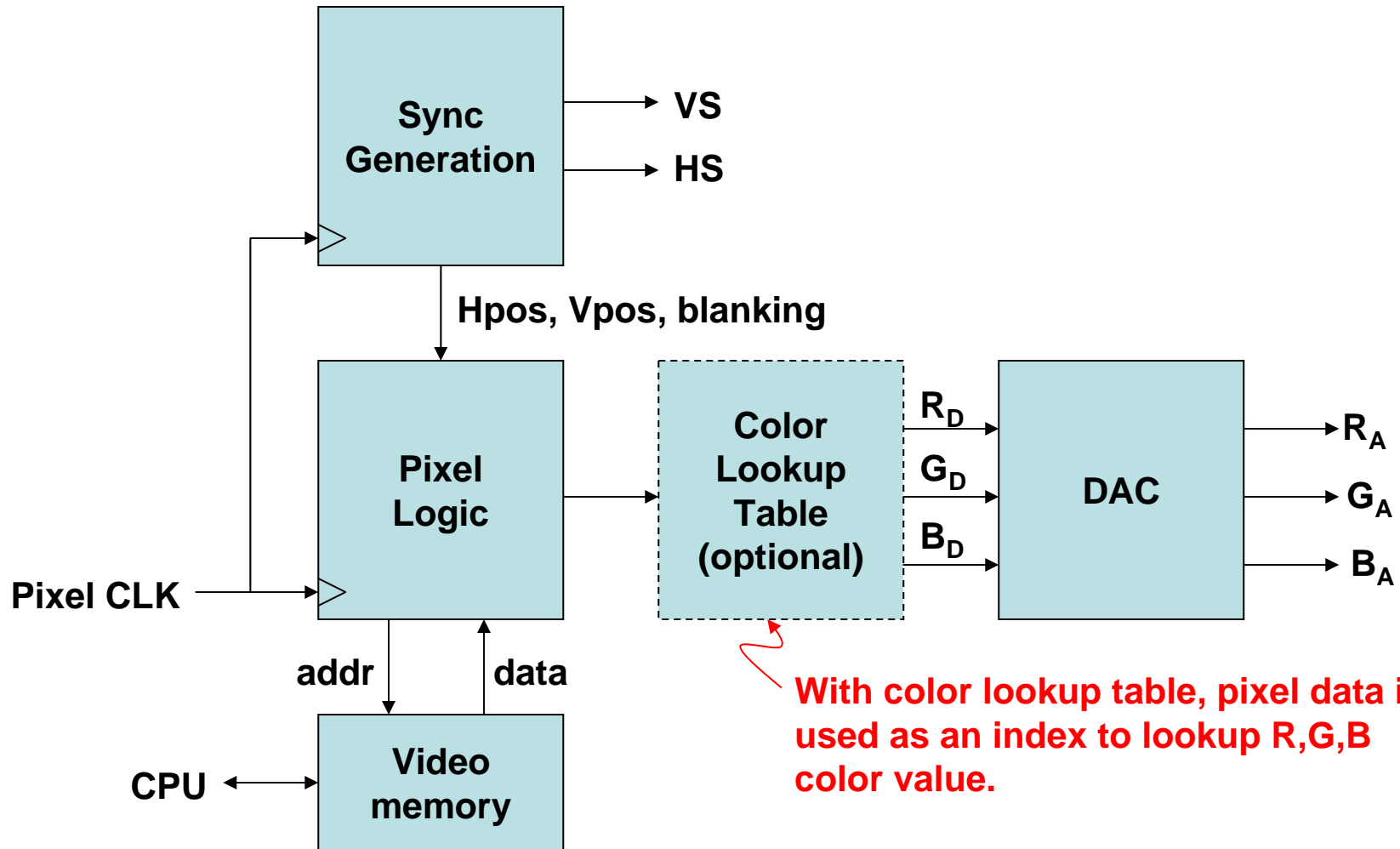
R, G, B are active-high.

Shown: a simple "8-color" scheme

The R, G and B signals are terminated with 75 Ohms to ground inside of the VGA monitor. So when you drive your 3.3V signal through the 270 Ohm series resistor, it shows up at the monitor as 0.7V - exactly what the VGA spec calls for.

$$0.7V = \left(\frac{75}{75 + 270}\right)(3.3V)$$

# Generating VGA-style Video



With color lookup table, pixel data is used as an index to lookup R,G,B color value.

Without color lookup table, pixel data is used directly as R,G,B value (aka "true color")

# Verilog: VGA Display

```
module vga(clk,select,hsync,vsync,rgb);
    input clk;                // 25 Mhz
    input [7:0] select;
    output hsync;
    output vsync;
    output [2:0] rgb;

    reg hsync,vsync,hblank,vblank;
    reg [9:0] hcount;         // pixel number on current line
    reg [9:0] vcount;        // line number

    wire hsyncon,hsyncoff,hreset,hblankon;
    wire vsyncon,vsyncoff,vreset,vblankon;

    // see next slide for generation of timing signals

    // sync and blanking
    always @(posedge clk) begin
        hcount <= hreset ? 0 : hcount + 1;
        hblank <= hreset ? 0 : hblankon ? 1 : hblank;
        hsync <= hsyncon ? 0 : hsyncoff ? 1 : hsync;    // active low

        vcount <= hreset ? (vreset ? 0 : vcount + 1) : vcount;
        vblank <= vreset ? 0 : vblankon ? 1 : vblank;
        vsync <= vsyncon ? 0 : vsyncoff ? 1 : vsync;    // active low
    end
end
```

# VGA (640x480) Sync Timing

```
// assume 25 Mhz pixel clock

// horizontal: 794 pixels = 31.76us
// display 640 pixels per line
assign hblankon = (hcount == 639); // turn on blanking
assign hsyncon = (hcount == 652); // turn on sync pulse
assign hsyncoff = (hcount == 746); // turn off sync pulse
assign hreset = (hcount == 793); // end of line (reset counter)

// vertical: 528 lines = 16.77us
// display 480 lines
assign vblankon = hreset & (vcount == 479); // turn on blanking
assign vsyncon = hreset & (vcount == 492); // turn on sync pulse
assign vsyncoff = hreset & (vcount == 494); // turn off sync pulse
assign vreset = hreset & (vcount == 527); // end of frame
```

# Demo

- **Frame**

```
always @(posedge clk) begin
    if (vblank | (hblank & ~hreset)) rgb <= 0;
    else
        rgb <= (hcount==0 | hcount==639 |
                vcount==0 | vcount==479) ? 7 : 0;
    end
```

- **Color bars**

```
always @(posedge clk) begin
    if (vblank | (hblank & ~hreset)) rgb <= 0;
    else
        rgb <= hcount[8:6];
    end
```

<i>RGB</i>	<i>Color</i>
000	black
001	blue
010	green
011	cyan
100	red
101	magenta
110	yellow
111	white

# Demo (cont'd.)

Character Display (80 columns x 40 rows, 8x12 glyph)

