ΗΜΥ 664
ΨΗΦΙΑΚΟΣ ΣΧΕΔΙΑΣΜΟΣ ΜΕ FPGAs
Χειμερινό Εξάμηνο 2010

ΔΙΑΛΕΞΗ ΕΡΓΑΣΤΗΡΙΟΥ:
Video Display Interface (VGA)

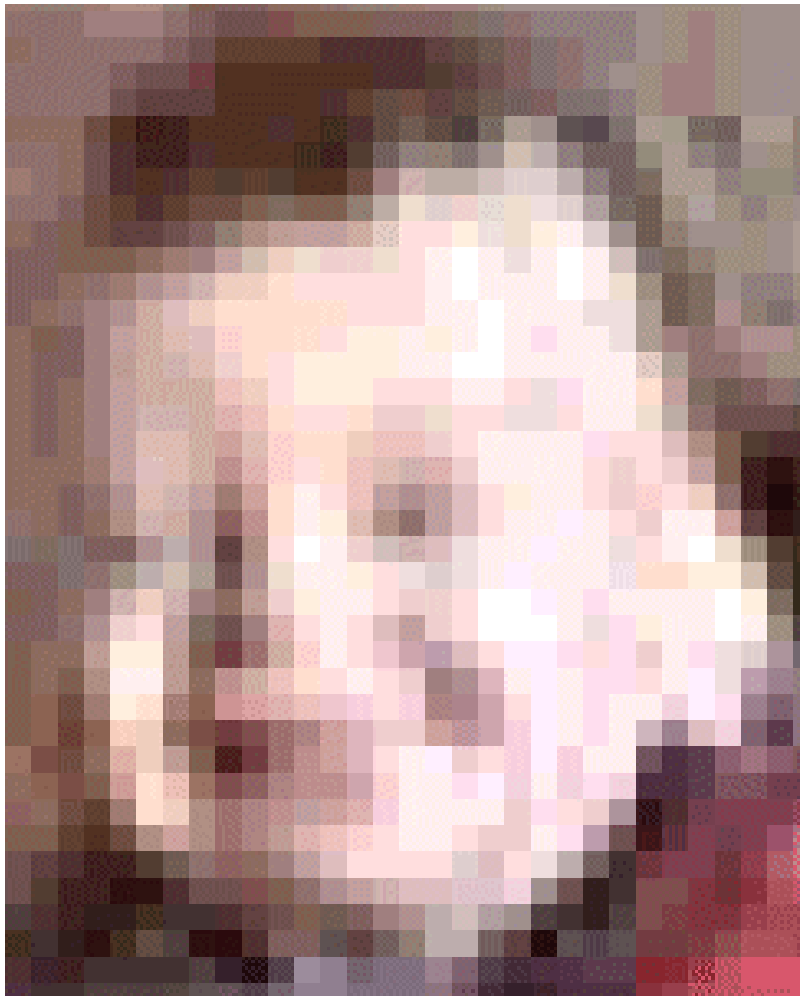
ΧΑΡΗΣ ΘΕΟΧΑΡΙΔΗΣ
(ttheocharides@ucy.ac.cy)

Today's Lecture Objectives

- Introduce the VGA video "standard"
- Provide an outline of how to use the triple DAC chip on the XUP board to drive video output

Dots to a Picture

- Brain can make this into something recognizable

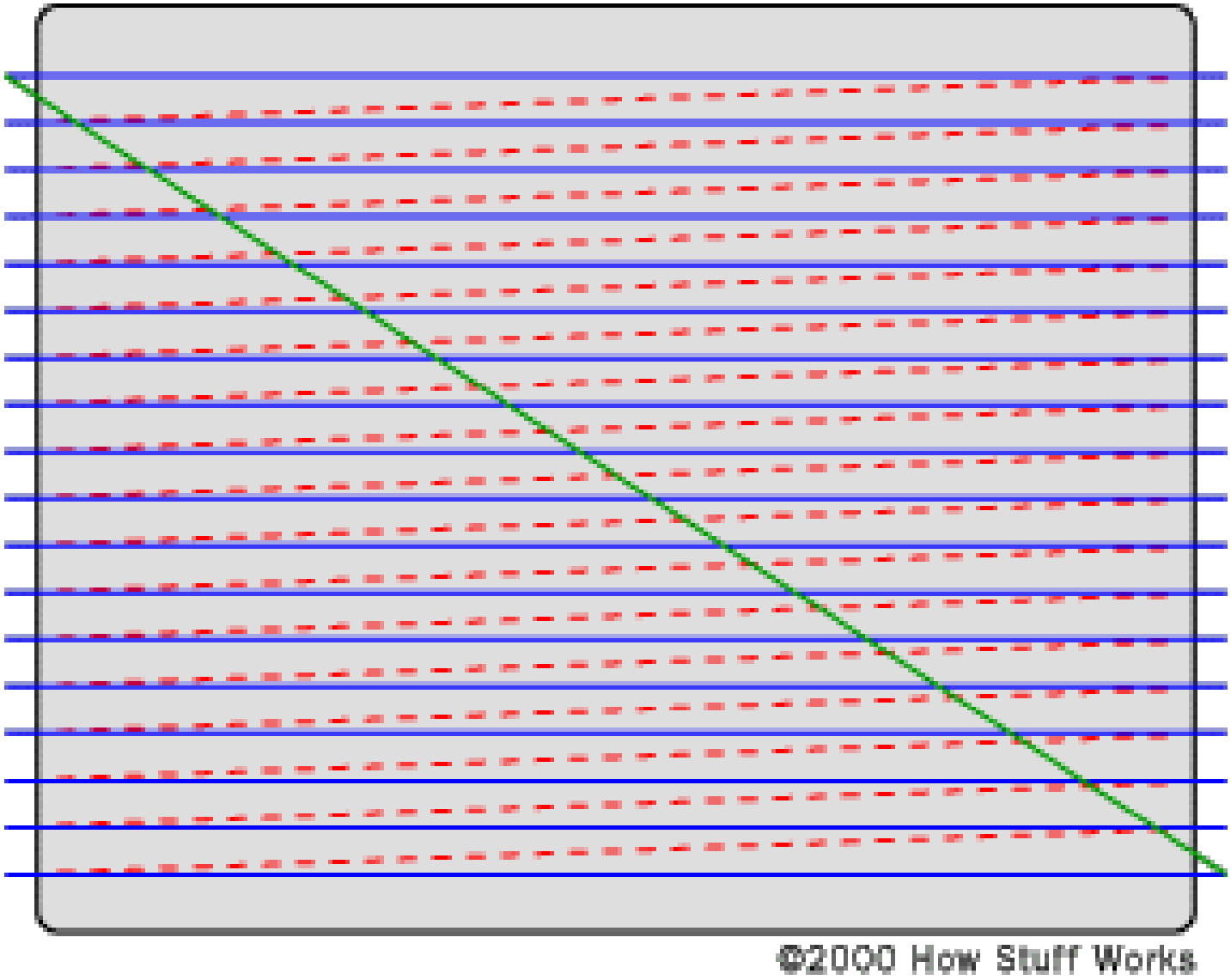


Images from howstuffworks.com

Many Still Images

- Video (and movies) are a series of stills
- If it goes fast enough
  - 50-60 Hz or more to not see flicker
- Your brain interprets as moving imagery
- Electron beam scans across
- Turned off when
  - Scanning back to the left (horizontal retrace)
  - Scanning to the top (vertical retrace)

Simple Scanning TV



Scanning

- TVs use *interlacing*
  - Every other scan line is swept per field
  - Two fields per frame (30Hz)
  - Way to make movement less disturbing
- Computers use *progressive scan*
  - Whole frame refreshed at once
  - 60Hz or more, 72Hz looks better

VGA Timing

- You supply two pulses, hsync and vsync, that let the monitor lock onto timing
- One hsync per scan line
- One vsync per frame

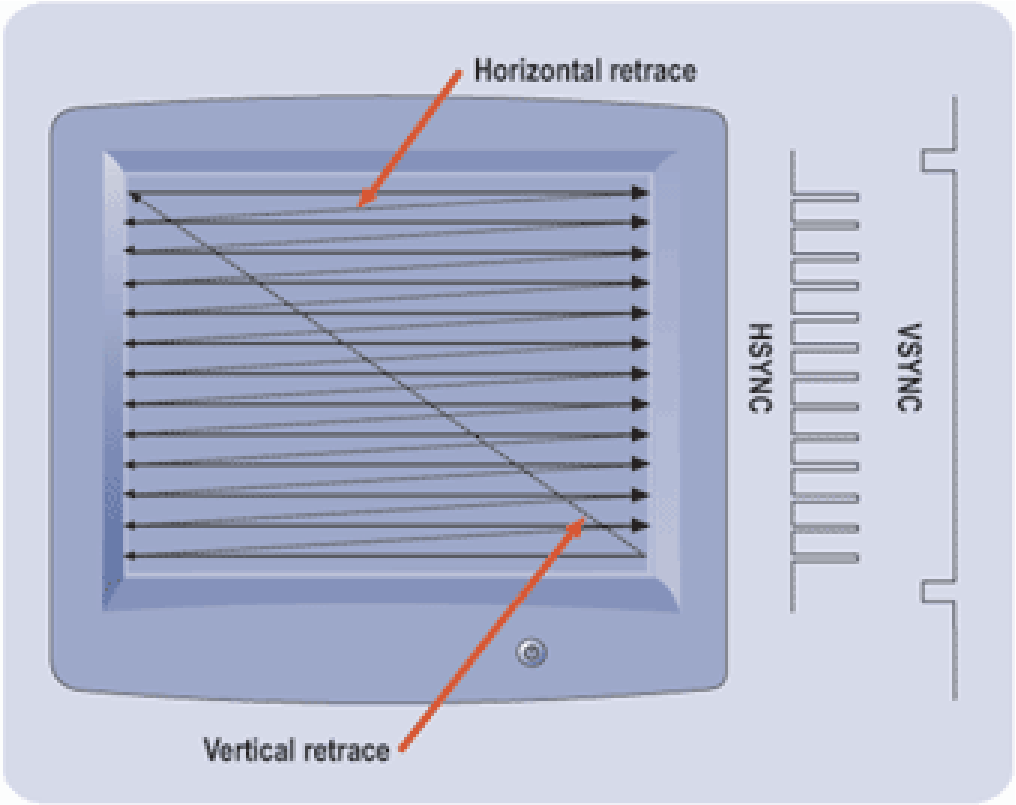
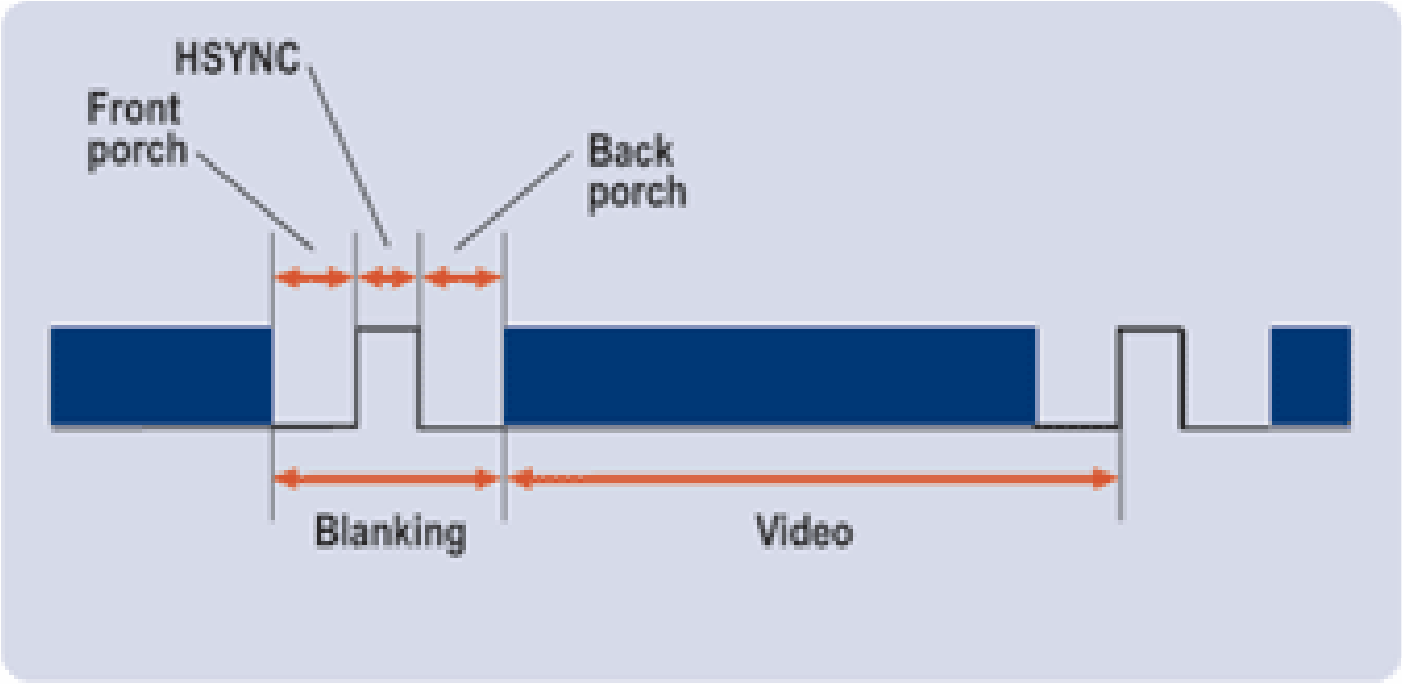


Image from dell.com

Horizontal Timing Terms

- hsync pulse
- Back porch (left side of display)
- Active Video
  - Video should be *blanked* (not sent) at other times
- Front porch (right side)

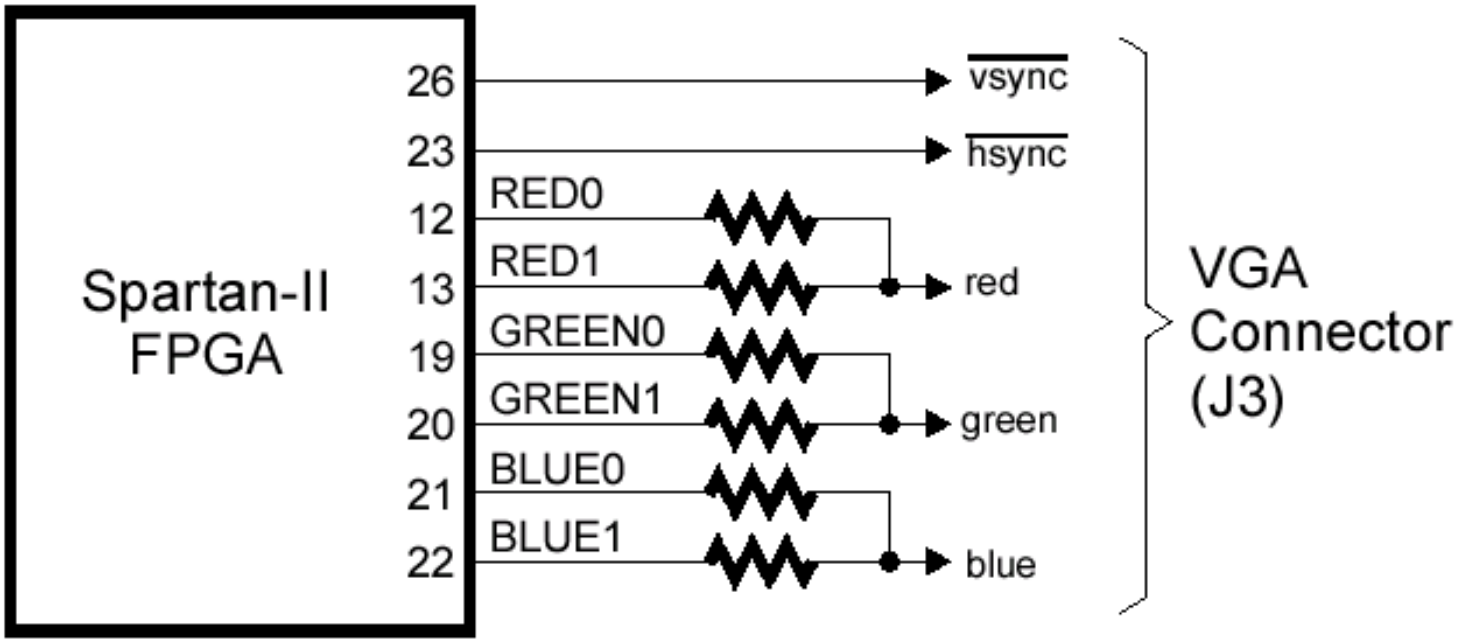


Standards

- 640 x 480 (60Hz) is “VGA”
- You can also try for 800x600 at 60 Hz (40 MHz exactly)
- or 800x600 at 72 Hz (50 MHz exactly)

Color Depth Example with Spartan-II

- Voltage of each of RGB determines color
- 2-bit color here (4 shades)
- Turn all on for white



PC Display Standards

- Monochrome Display Adapter (MDA)
  - Earliest display system for IBM PCs
  - Text-only – 80x25 characters, each character is 9x14 pixels
  - Effective resolution 720x350 @ 50 Hz, but pixels not individually addressable
    - IBM provided extra characters in its character set that allowed primitive graphics (boxes, lines, etc.) to be “drawn” in MDA
- Hercules Graphics Card
  - Third-party system, adds monochrome graphics to MDA text
- Color Graphics Adapter (CGA)
  - First mainstream color system for IBM PC
  - Text up to 80x25, graphics range from 640x200 in monochrome to 160x200 in 16 colors
  - Text displayed with lower quality than MDA (8x8 pixels/char)

PC Display Standards

- Enhanced Graphics Adapter (EGA)
  - 16 colors out of palette of 64 at 640x350, 80x25 text at 60 Hz
  - Minimum requirement for Windows 3.x
- Video Graphics Adapter (VGA)
  - Last really accepted standard defined by IBM (consequence of IBM losing control of the “PC”)
  - 256 colors at 320x200, 16 colors at 640x480
  - Not 256 colors at 640x480, even though that’s what most people accept when they say “VGA Resolution”
- Super VGA and other formats
  - Much less well-defined
  - VESA standards developed for software compatibility between different cards

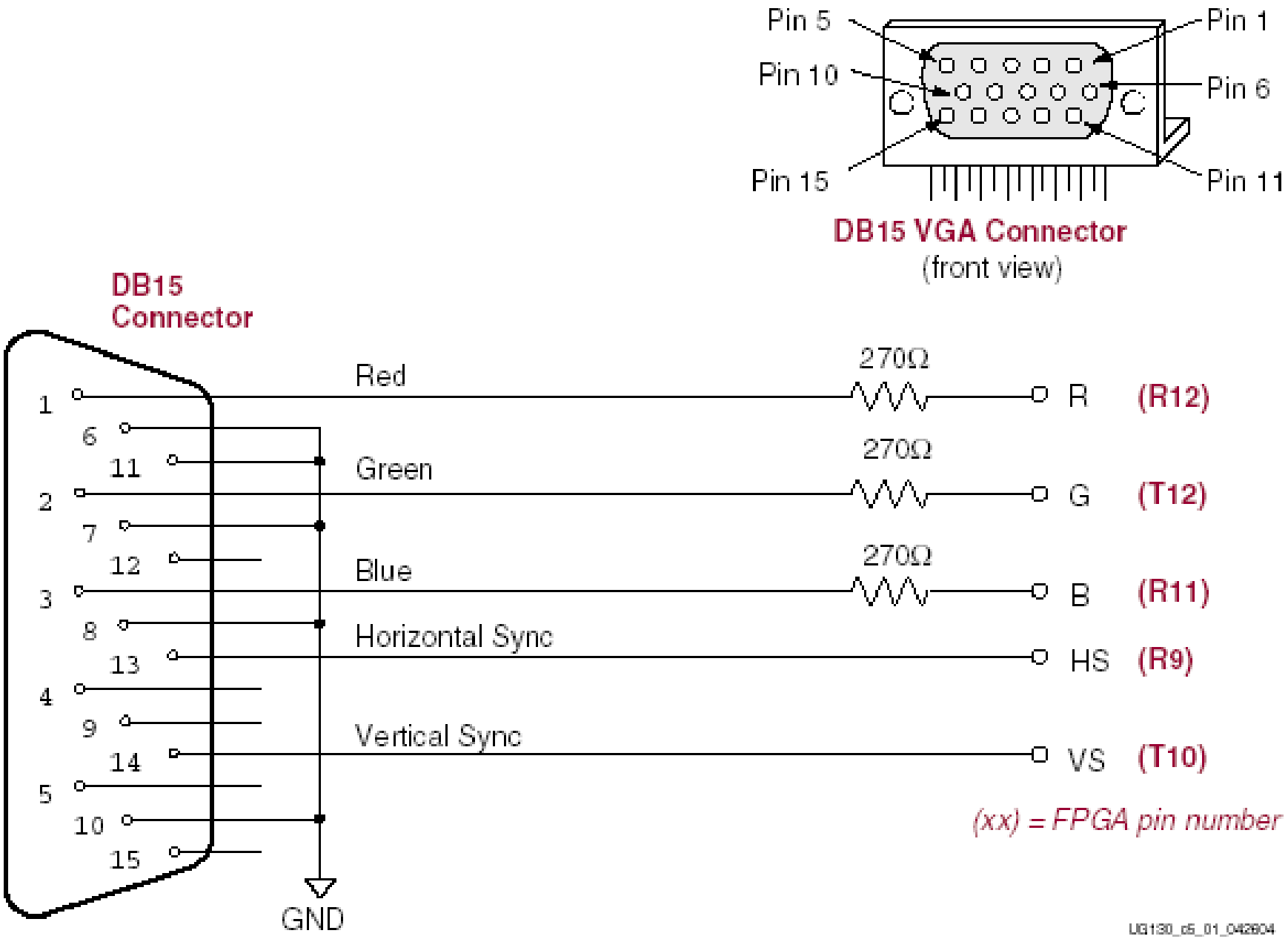


Figure 5-1: **VGA Connections from Spartan-3 Starter Kit Board**

Table 5-2: **3-Bit Display Color Codes**

Red (R)	Green (G)	Blue (B)	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

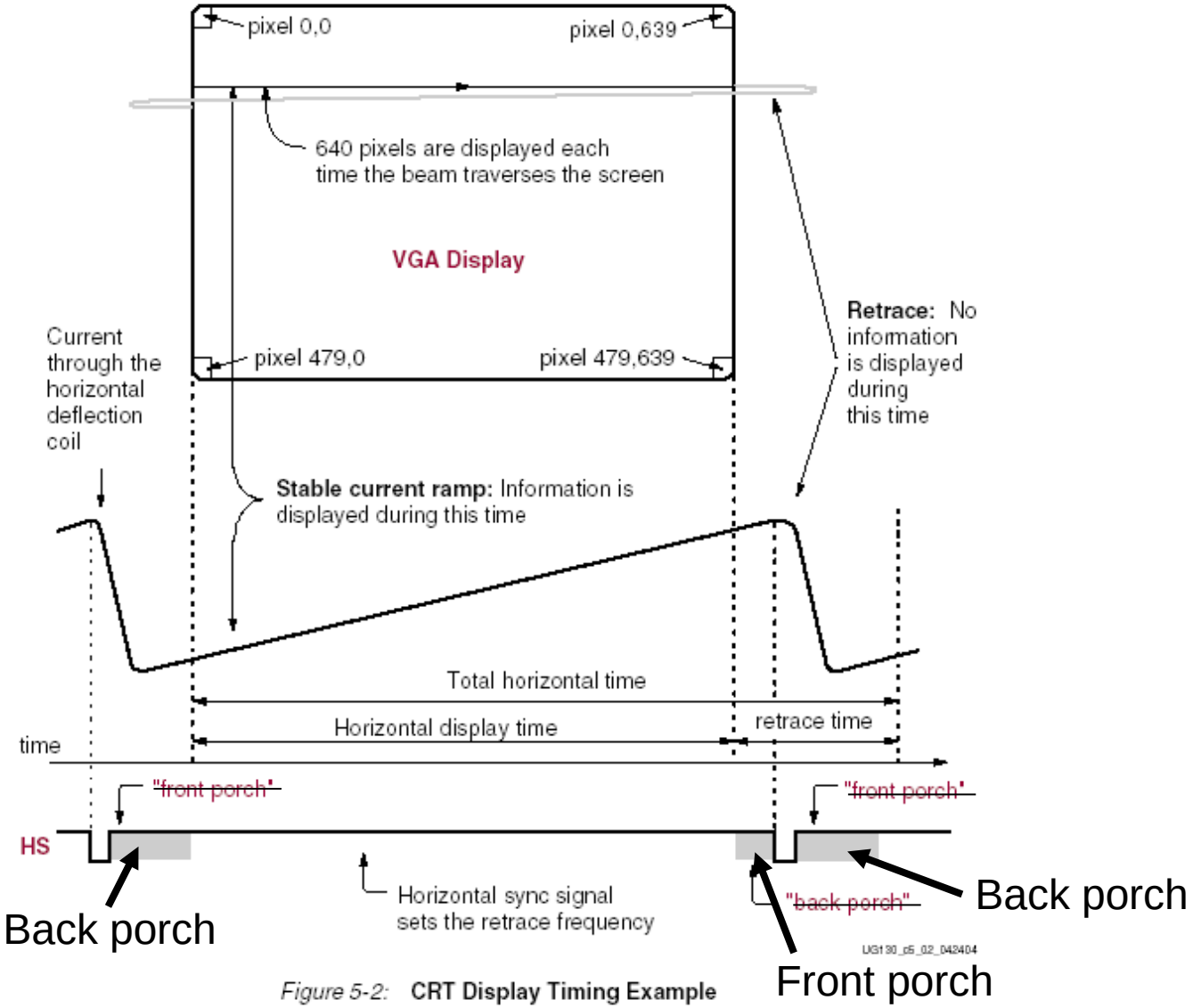
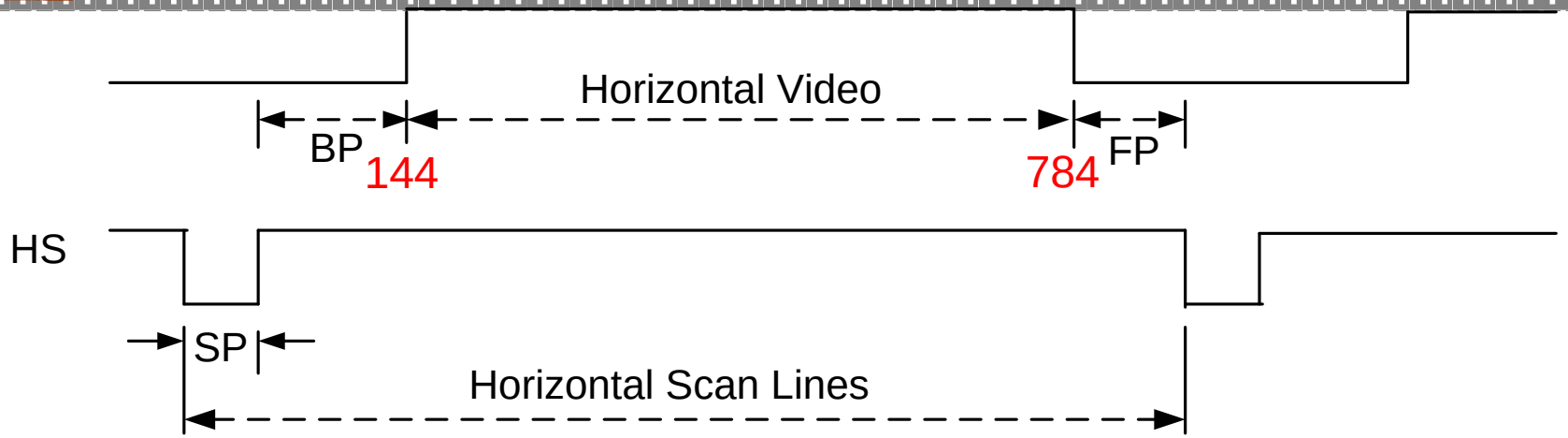


Figure 5-2: **CRT Display Timing Example**

Horizontal Timing



Pixel clock = 25 MHz Pixel time = 0.04 μs

Horizontal video = 640 pixels x 0.04 μs = 25.60 μs

Back porch, BP = 16 pixels x 0.04 μs = 0.64 μs

Front porch, FP = 16 pixels x 0.04 μs = 0.64 μs

Sync pulse, SP = 128 pixels x 0.04 μs = 5.12.μs

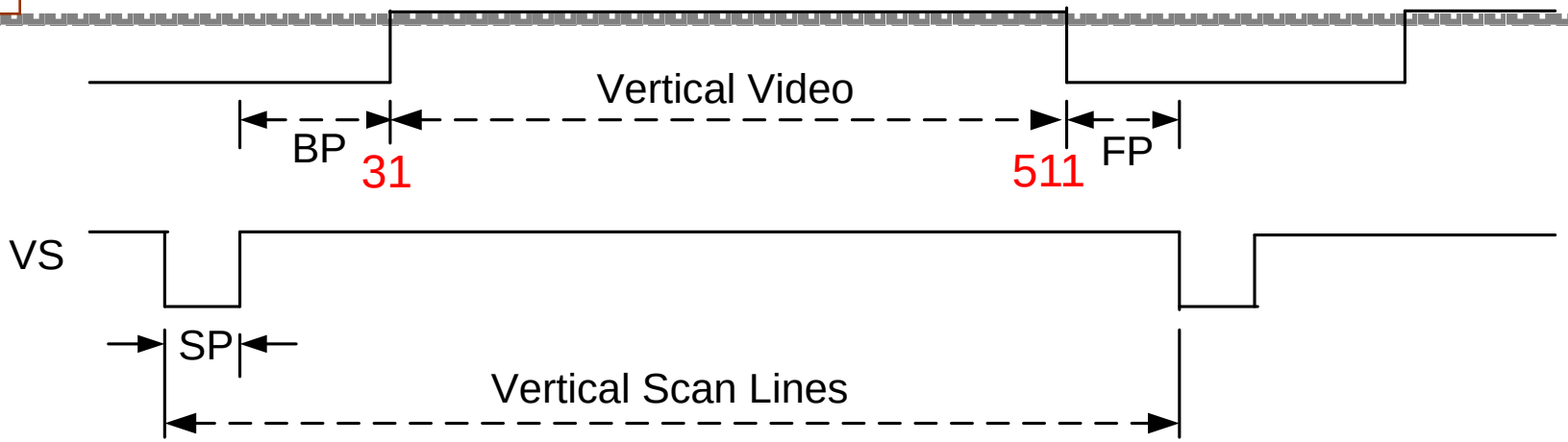
Horizontal Scan Lines = SP + BP + HV + FP

= 128 + 16 + 640 + 16

= 800 pixels x 0.04 μs = 32 μs

1/60 Hz = 16.67 ms / 32 μs = 521 horizontal scan lines per frame

Vertical Timing



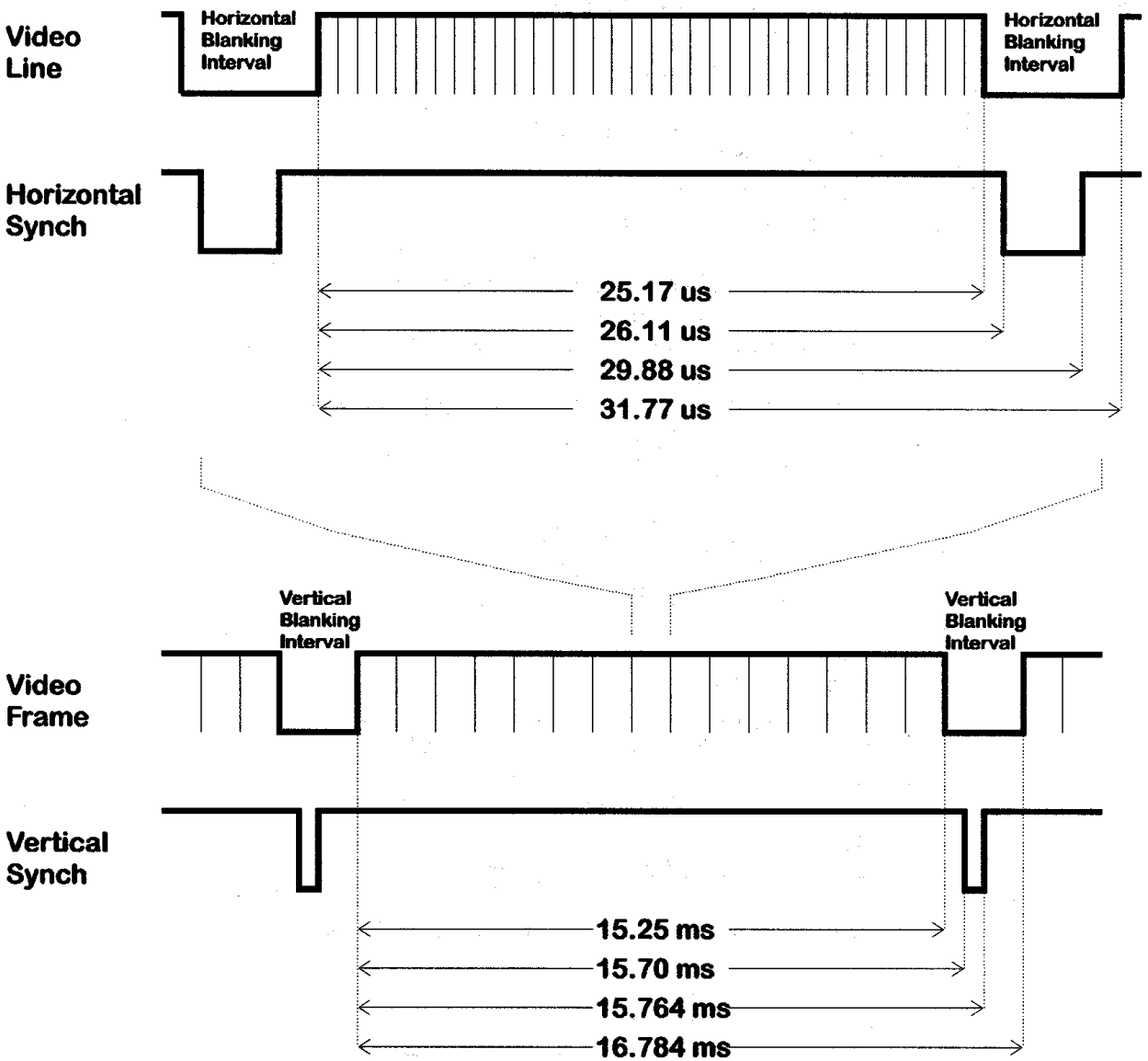
Pixel clock = 25 MHz Horizontal scan time = 32 μs
Vertical video = 480 lines x 32 μs = 15.360 ms
Back porch, BP = 29 lines x 32 μs = 0.928 ms
Front porch, FP = 10 lines x 32 μs = 0.320 ms
Sync pulse, SP = 2 lines x 32 μs = 0.064 ms
Vertical Scan Lines = SP + BP + VV + FP
= 2 + 29 + 480 + 10
= 521 lines x 32 μs = 16.672 ms

1/60 Hz = 16.67 ms

How to Make the VGA Work!

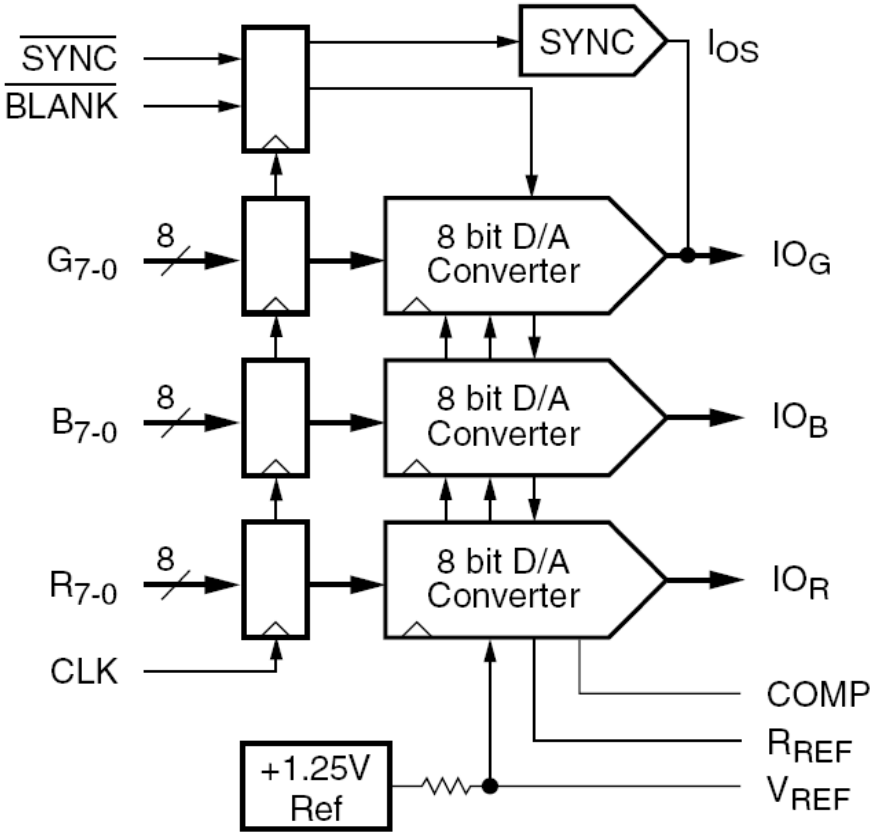
- Basically, driving a VGA display involves doing the video decoder operation in reverse.
- Start with a digital representation of the image
- Hardware (video DAC) converts digital pixels into analog voltages used by the monitor
  - Different display standards have used either analog or digital signals in the past
- Your hardware has to drive the control signals to the display and provide pixel values at the right rate
  - DAC just does conversion

VGA Timing

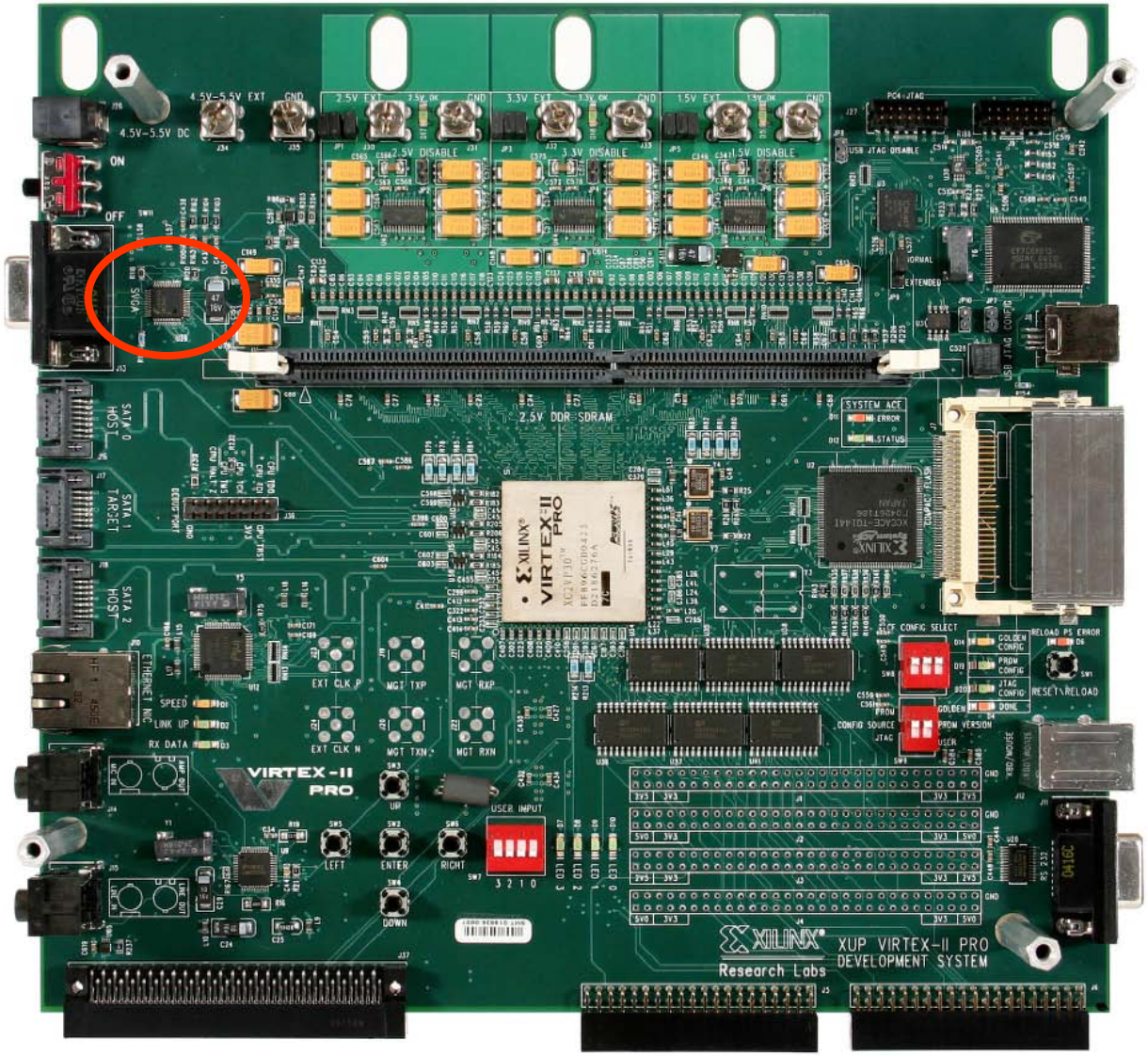


FMS3818KRC Triple DAC

- Converts digital pixel data to VGA outputs



XSGA Video DAC



General Operation

- Operation
  - See datasheet and XUP manual* for supported modes

- You are responsible for managing the timing of the VGA signal
  - Clock rate/time per line will give you the max number of pixels/line you can drive
  - Number of lines on the screen determined by ratio of time/frame and time/screen
  - FPGA has to drive HSYNC and VSYNC signals directly

Driving the FMS3818KRC

- CLK used to latch input registers
- 24-bit pixel input bus carries color of each pixel
- BLANK* signal specifies blanking interval
  - Assert this, don't just drive a color that should be black
- SYNC* inserts sync pulse on I_{OG} output
  - Does not override any other data
  - Use only during blanking interval

Making VGA Work

1. You may start with say an 8 bit per pixel image (see below).
2. Generate an image in the appropriate 8-bit format
   - For video capture, you could use an 8-bit RGB spectrum, with 3 bits of red, 3 of green, and 2 of blue (human eyes are less sensitive to blue)
   - Previously, students reported they'd found routines to convert an RGB image into 8-bit format and generate the optimal palette for that image
3. Drive the image pixels to the triple DAC with appropriate timing signals

Advice / Experience

- A number of course projects had working 640x480 video in 256 colors working for their final project, so that path seems pretty stable
- Would recommend against going to higher resolutions/color depths for your projects
  - You only have ~250KB of memory available presently, and one frame of 640x480x256 video takes 300 KB
  - Maximum clock rate for the FMS3818KRC is 180 MHz, which allows about 10x more pixels than 640x480, but handling that clock rate is out of our system
  - Higher color depths not supported by this chip
  - RAM not present on chip as in past semesters

Reference Designs

- Your VP2CD contains several designs that use the VGA controller, and you may use code snippets from there. You may also use the entire VGA controllers from there, provided of course you realize how they work…

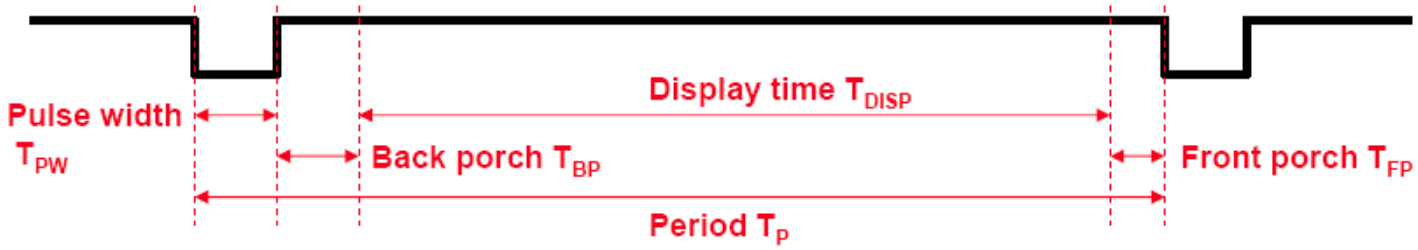
VGA – UCF Files (From Built-In Demo)

```
NET "VGA_COMP_SYNCH_N"  LOC = "G12" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_HSYNCH"  LOC = "B8" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLANK_N"  LOC = "A8" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<0>"  LOC = "D15" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<1>"  LOC = "E15" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<2>"  LOC = "H15" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<3>"  LOC = "J15" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<4>"  LOC = "C13" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<5>"  LOC = "D13" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<6>"  LOC = "D14" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_BLUE<7>"  LOC = "E14" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<0>"  LOC = "G10" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<1>"  LOC = "E10" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<2>"  LOC = "D10" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<3>"  LOC = "D8" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<4>"  LOC = "C8" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<5>"  LOC = "H11" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<6>"  LOC = "G11" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_GREEN<7>"  LOC = "E11" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_PIXEL_CLOCK"  LOC = "H12" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = FAST ;
NET "VGA_OUT_RED<0>"  LOC = "G8" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<1>"  LOC = "H9" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<2>"  LOC = "G9" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<3>"  LOC = "F9" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<4>"  LOC = "F10" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<5>"  LOC = "D7" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<6>"  LOC = "C7" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_OUT_RED<7>"  LOC = "H10" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
NET "VGA_VSYNCH"  LOC = "D11" | IOSTANDARD = LVTTL  | DRIVE = 12  | SLEW = SLOW ;
```

VGA Timing

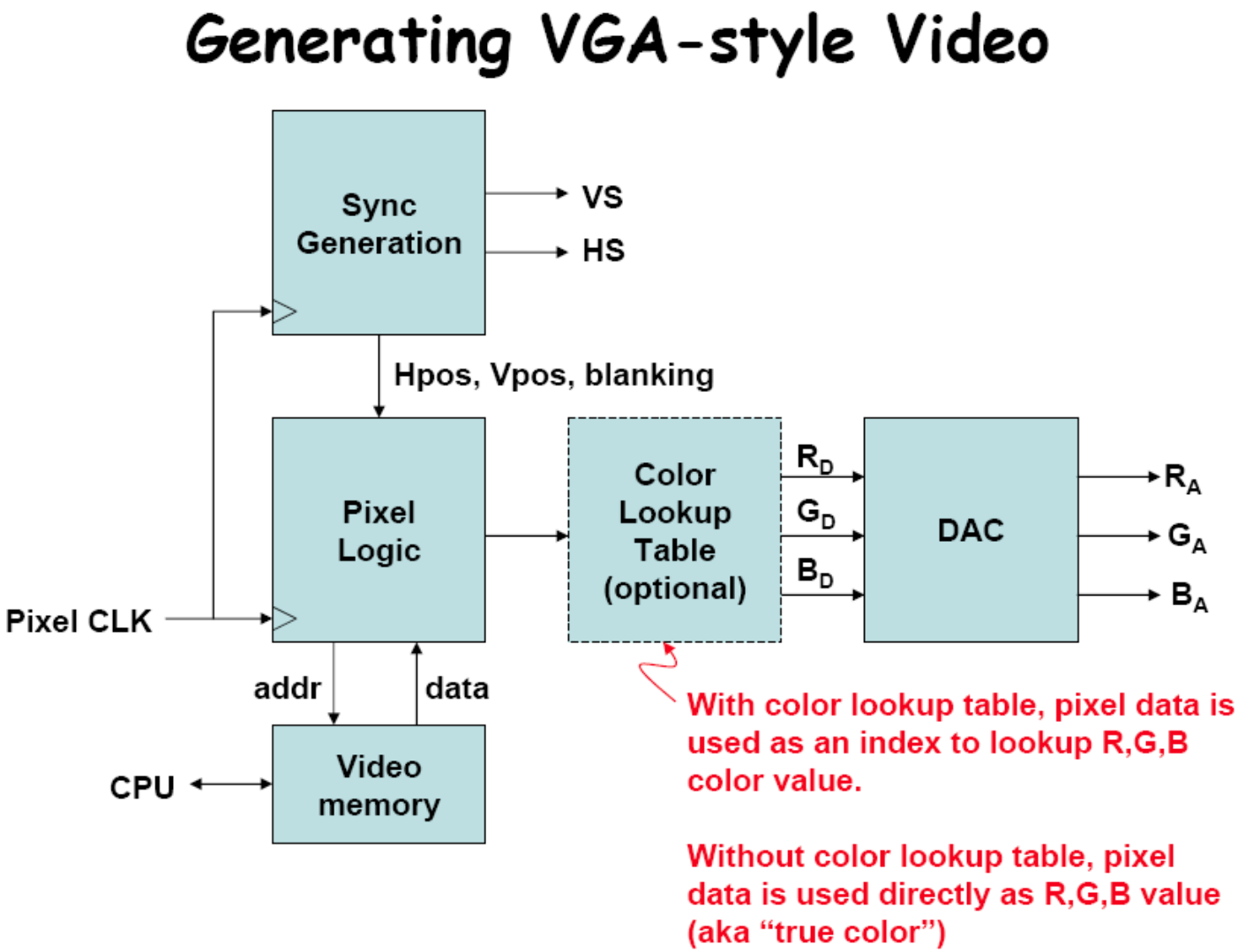
Sync Signal Timing

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).



Format		*CLK*	*P*	*PW*	*BP*	*DISP*	*FP*
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

VGA Generation



Example…

```
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
```

Verilog:
VGA Display

Example…

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
```

Links

- VGA Timing
  - http://www.epanorama.net/documents/pc/vga_timing.html
  - http://appsrv.cse.cuhk.edu.hk/~ceg3480/Tutorial7/tut7.doc
- Code (more complex than you want)
  - http://www.stanford.edu/class/ee183/index.shtml
- Interesting
  - http://www.howstuffworks.com/tv.htm
  - http://computer.howstuffworks.com/monitor.htm
  - http://www.howstuffworks.com/lcd.htm
  - http://plc.cwru.edu/
  - Liquid Crystals by S. Chandrasekhar, Cambridge Univ. Press
- http://server.oersted.dtu.dk/personal/sn/31002/?Materials/vga/main.html (Really Good VHDL)