CS/EE 3710 – Computer Design Lab
Lab 3 – VGA

Due Thursday September 30, 2010

Lab Objectives

The objectives of this lab are to build and demonstrate a simple VGA controller driving a VGA display. I had a colleague who liked to say: "There are two kinds of engineers – those who think VGA is hard, and those who understand VGA." That is to say, it's a little intimidating when you first see it, but it's really a very simple protocol for sending output information to a video screen. The steps are:

1. Read and understand the VGA specification. I recommend breaking this down into a VGA control circuit that generates the timing signals, and a bitGen circuit that defines what color each pixel should be as the information is being sent to the display.

2. Design and implement a VGA control/timing circuit and simulate that circuit using ISE.

3. Design and demonstrate a very simple bitGen circuit that takes input from three switches on the Spartan3e board and drives the entire screen to the color represented by the values on those switches.

4. Design and demonstrate a slightly more interesting bitGen2 circuit that drives the display with the output from your TBird turn signal state machine. That is, in addition to lighting up the LEDs on the Spartan3e board, you will drive the VGA display as a TBird output device.

VGA Basics

The term VGA really means one of two things depending on how you use the acronym. It's either a standard 15-pin connector used to drive video devices (e.g. a VGA cable) or it's the protocol used to drive information out on that cable (e.g. a VGA interface spec.). The interface defines how information is sent across the wires from your board to the VGA device. The cable defines which pins you use on the standard connector for those signals.

There are some slides on VGA on the class web site, and there is a nice description of VGA (both the connector and the protocol) in the Spartan3E board User Guide, also linked to the class web site. The most basic thing to know about VGA is that it is a protocol designed to be used with analog CRT (cathode ray tube) output devices. On these devices the electron beam moves across the screen from left to right as you're looking at the screen at a fixed rate (the refresh rate defines how fast the beam moves), and also moves down the screen from top to bottom at a fixed rate. While it's moving across and down the screen, you can modify the Red, Green, and Blue values on the VGA interface to control what color is being painted to the screen at the current location. So, painting a certain color on the screen is as easy as keeping track of where the beam is, and making sure the R, G, and B signals are at the right values when the beam is over the point on the screen where you want that color.

If you don't do anything to stop it, the beam will move to the right and bottom of the screen and get stuck there. You can force the beam to move back to the left by asserting an active-low signal called hSync (horizontal sync). You can force the beam to move back to the top of the screen by asserting an active-low signal called vSync (vertical sync). Because the beam moves at a fixed

rate (defined by the monitor's refresh rate), you can keep track of where the beam is on the screen by counting clock ticks after the hSync and vSync signals.

So, the basics of the VGA control/timer circuit are just a pair of counters to count horizontal ticks and vertical ticks of the VGA clock. How many ticks are there? That depends on how fast your clock is, and how many pixels you want to paint during the time the beam moves across the screen. The basic (ancient) standard for "plain" VGA is 640 pixels on each line, and 480 lines down the screen. This is "640x480" mode. Figure 1 shows a 640x480 screen, and the horizontal sync (hSync) timing required to make it work. After the hSync pulse, you must wait for a certain number of ticks before painting pixels to the screen. This gives the beam time to get back to the left and start moving forward again. This time is called the "back porch" because it's in back of the hSync timing pulse. Then you count 640 pixels as the beam moves. After the 640th pixel, you wait for some amount of time (this is the "front porch" because it's in front of hSync), then assert the hSync signal (asserted low) for a certain amount of time. Note that the figure in the Spartan3e User Guide has the front porch and back porch reversed. This figure is correct.

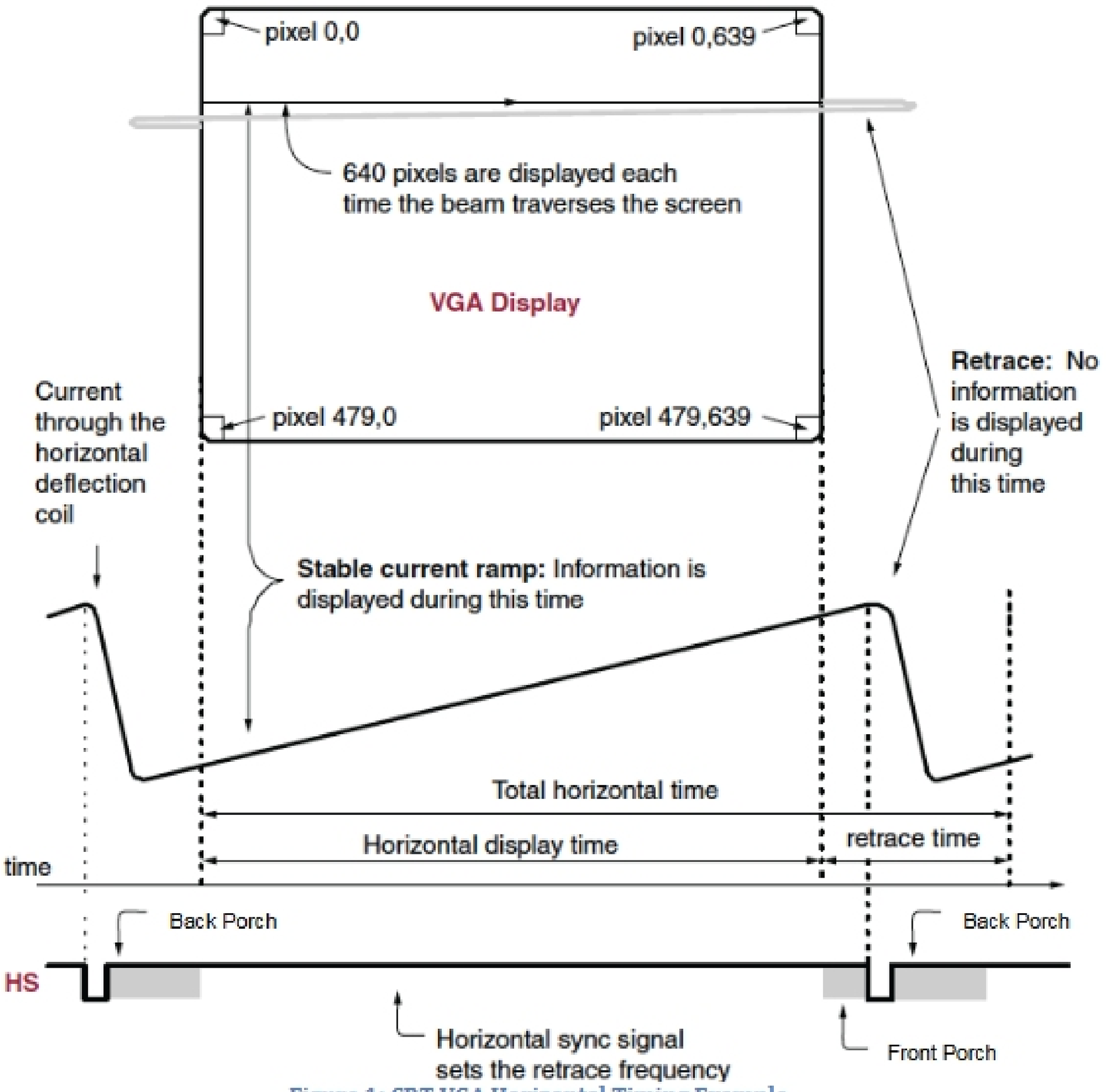

Figure 1: CRT VGA Horizontal Timing Example

The timing for all this depends on the monitor refresh rate. For a monitor with 60Hz refresh in 640x480 mode and a 25MHz pixel clock, you can use the timings in Figure 2. The timings in this figure define the display time (the time when the pixel is one of the 640 visible pixels in a line), pulse width (hSync or vSync), and the front porch and back porch timings. These times (in µs or ms) can also be measured in terms of the number of ticks of the 25MHz pixel clock, or in terms of the number of horizontal lines. That is, the vertical timing can be measured in terms of how many hSync pulses have been seen. The bottom line is that both the horizontal and vertical timing for the vgaControl are just counters. You may have to enable things or reset things or change things when the counters get to a certain value, but basically they're just counters.

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_S	Sync pulse time	16.7 ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back porch	928 µs	23,200	29	1.92 µs	48

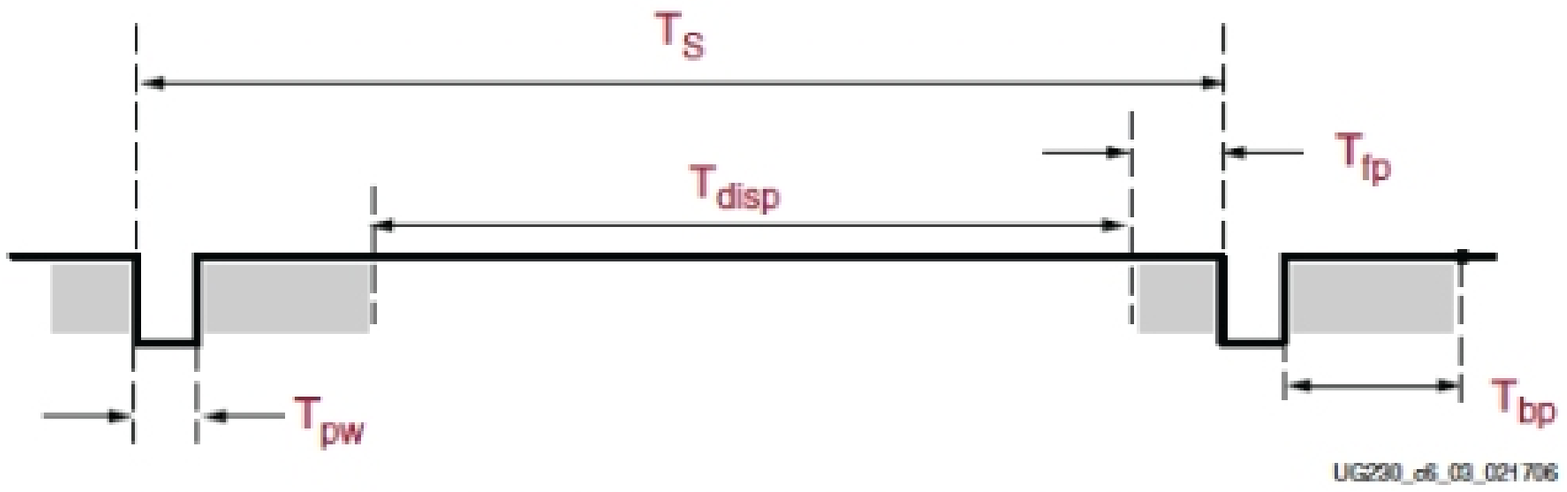

Figure 2: VGA Control Timing at 60Hz refresh and 25MHz pixel clock

I recommend designing a circuit that has two counters: one to keep track of horizontal location and one that keeps track of vertical location. The horizontal counter should count up in 25MHz ticks, and the vertical counter should count once for each full horizontal line.

Note that if you're going to use a purely synchronous design method (which I highly recommend!), you should NOT expect to have a 25MHz clock at your disposal. The system clock on the Spartan3e board is 50MHz. You could divide that by two and use that output as the 25MHz clock, but that would NOT be purely synchronous because you would be using a derived signal as your clock. Instead you should generate an enable signal that is asserted every two 50MHz clocks, and use that enable signal in your horizontal pixel counter. For example, the horizontal pixel counter might look like:

```
// An example of a counter that only counts when En is high.
// This particular En is assumed to be high for a single clock
// cycle when it's time to count
always@(negedge clr, posedge clk50MHz)
       if (clr == 0) count = 0;          // clear count if clr is asserted
       else if (En) count = count + 1;   // If En is high, count
              // otherwise, hold previous value (default)
```