Lab 4: VGA Display

Introduction

This lab will introduce the concepts of counting, state machines, and timing in digital systems in order to directly control a VGA monitor. Successful implementation of the VGA monitor controller will require effective use of nested instantiation of modules and hierarchical design.

Background

The vector graphics array (VGA) standard was developed to enable the visualization of photorealistic images on computers. While the human eye is capable of distinguishing millions of different color hues, there are a finite number of pins that can connect between a digital control system and a monitor. Proper implementation requires precise timing control and knowledge of D/A conversion techniques. In this lab you will use an FPGA to generate the timing signals and also the individual pixel colors for display on the computer monitor. The full VGA display system is shown in figure 1.

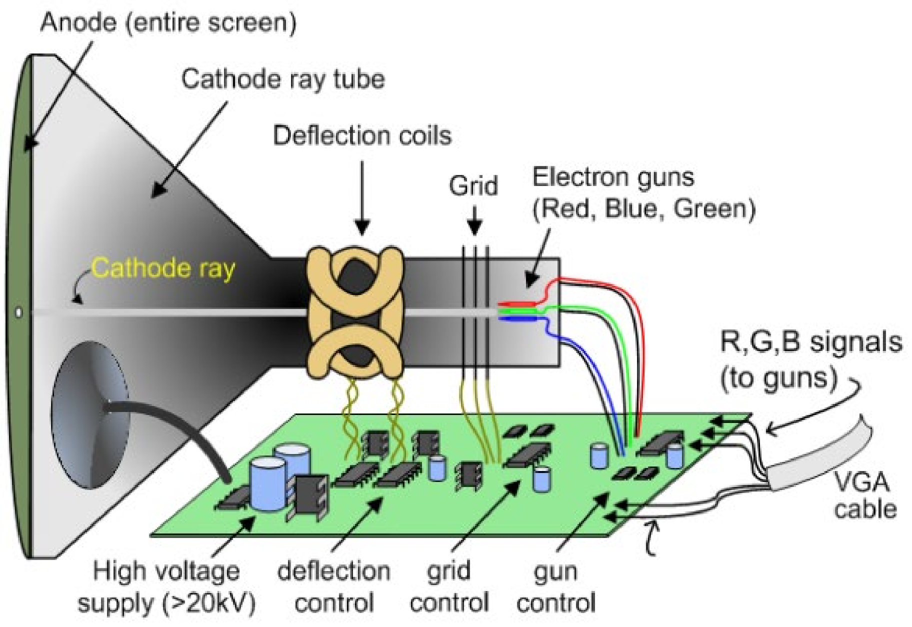


Figure 1

There are 5 signals that need to be generated the VGA controller:

1. VSYNC – the vertical synchronization signal (digital).
2. HSYNC – the horizontal synchronization signal (digital).
3. RED – the red component (analog, 0~0.7V 75Ω termination).
4. GREEN – the green component (analog, 0~0.7V 75Ω termination).
5. BLUE – the blue component (analog, 0~0.7V 75Ω termination).

Several other signals are part of the standard but these signals are the minimum set that a designer needs to generate. Given a digital clock signal that oscillates at 100 MHz and a predefined pattern and 3 digital-to-analog converters, it is possible to create a digital circuit that generates screen images. Figure 2 shows how the digital signals need to be generated (HSYNCH in this particular case.)

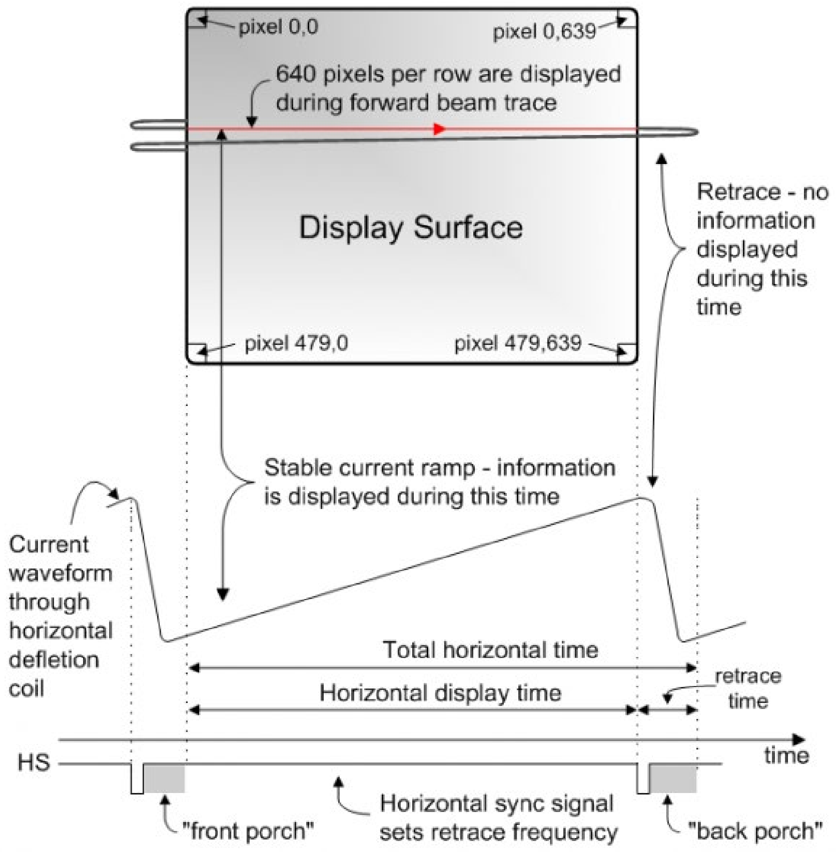

pixel 0,0
pixel 0,639
640 pixels per row are displayed during forward beam trace
Display Surface
Retrace - no information displayed during this time
pixel 479,0
pixel 479,639
Stable current ramp - information is displayed during this time
Current waveform through horizontal defletion coil
Total horizontal time
Horizontal display time
retrace time
HS
time
"front porch"
Horizontal sync signal sets retrace frequency
"back porch"

Figure 2

In old cathode ray tube (CRT) systems, an electron gun was aimed by using a sawtooth waveform. When the sawtooth wave wrapped around suddenly switching from high to low voltage, there would be a discontinuity which would cause visual anomalies. Therefore, during this wraparound time, the electron gun would not cause the monitor to display. This moment in time is called the “front-porch”. Synchronization of the sawtooth wave and the VGA controller would be done using the HS or HSYNCH signal—a digital signal that drops to zero to indicate the start of a row, then rises to indicate the beginning of the front-porch. A similar waveform is generated for the VS or VSYNCH signal with different timing requirements.

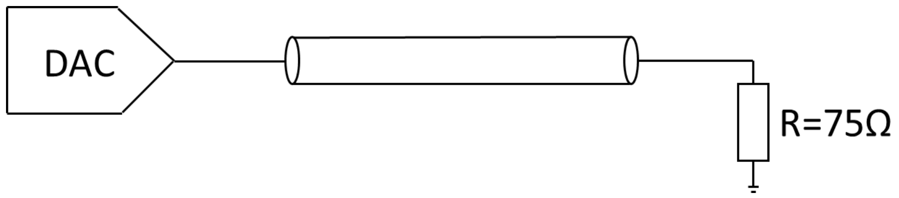


Figure 3

Figure 3 and figure 4 show the digital to analog converter (DAC) interface that is connected to the Basys3 FPGA which transmits the red/green/blue signals. The four resistors on the left side of figure 4 represent a 4-bit DAC. The 75Ω resistor is within the monitor and the DAC circuit is on the FPGA board. The FPGA digital pins connect to the digital-to-analog controller board.

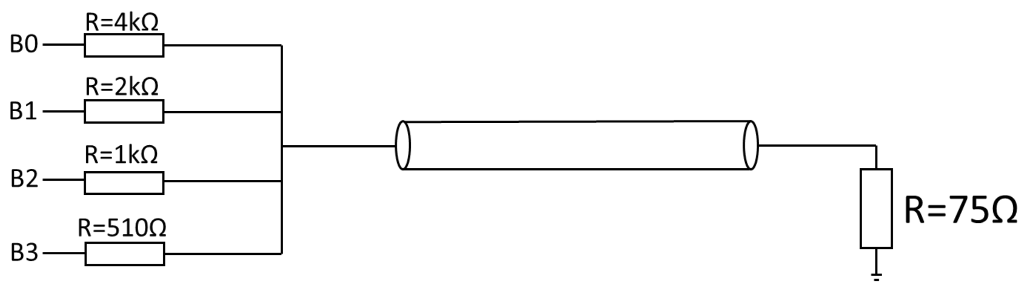
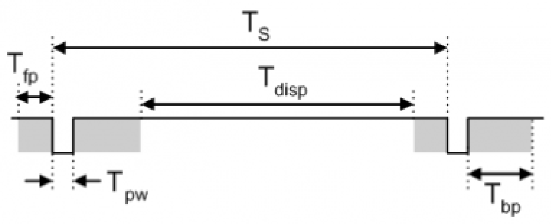
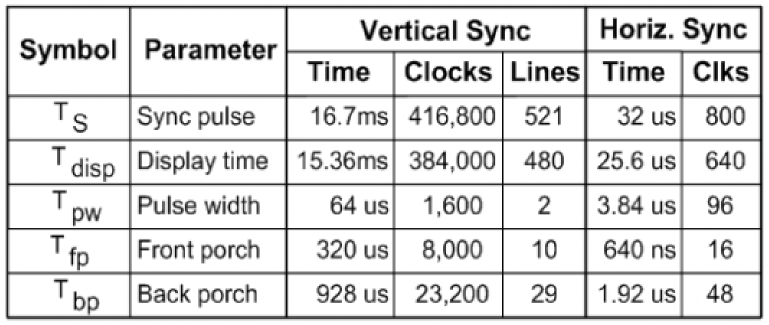


Figure 4

There are three, 4-bit DACs on the Basys3 that control the red, green, and blue color fields of each pixel. This means that the Basys3 can display 12-bit color images.

The timing of HSYNCH and VSYNCH are specified per figure 5.





Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Figure 5

Further information about how to connect the FPGA to the VGA port can be found in the Basys3 manual: https://digilent.com/reference/programmable-logic/basys-3/reference-manual

Preparation

Please review the textbook Digital Design and Computer Architecture by Harris & Harris, Chapter 4 for information on how to specify flip-flops and state machines in Verilog. For practice, you may attempt to build simple counter circuits that flash an LED on and off.

Procedure

In this lab you will implement a VGA control module in Verilog HDL that outputs the VSYNCH, HSYNC, and digital RGB components. First you will develop a simple counter circuit then demonstrate its functionality with a simulation testbench and demonstrate functionality with the oscilloscope. You will then develop a module that outputs the digital HSYNC and VSYNC timing signals. Finally, you will develop a state machine that outputs an image to the screen.

Experiment 1

For this experiment you will design a counter which acts like a clock divider which has a 75% duty cycle. This means that the output is high for 75% of the time, and low for 25% of the time. Implement and module as follows with its test bench.

```
module genclk(input clk, input rst, output gclk);
  reg [13:0] ctr; // declare 14-bit register
  wire gclk; // declare output signal

  // continuously toggle the clock
  assign gclk = (ctr < 14'd7500) ? 1'b1 : 1'b0; // uses if/else

  // trigger at the positive edge of the clock or active low reset
  always @(posedge clk or negedge rst) begin
    if (~rst) begin // if rst==0
      ctr <= 14'd0; // set the flip flops all equal to zero
    end else begin
      if (ctr < 14'd10000) begin
        ctr <= ctr + 14'd1;
      end else begin
        ctr <= 14'd0;
      end
    end
  end
endmodule

module testbench();
  reg rst,clk;
  wire gclk;

  genclk gc0(clk,rst,gclk);

  always begin
    #10 clk = ~clk; // trigger the clock every 20 ticks
  end

  initial begin
    rst = 1'b0; // reset the system
    clk = 1'b0; // set the initial value of the clock
    #15
    rst = 1'b1; // start the state machine
    #4000000000 // let it run for 400 million time ticks
  end

  // when a rising edge is detected on clock, print the time and the
  // associated values of clk and gclk.
  always @(posedge clk) begin
    $display("clk=%b gclk=%b time=%d",clk,gclk,$time);
  end
endmodule
```

This module and testbench demonstrate how to build a counter and automatically generate a clock in the testbench to test it. The **initial** block specifies a series of events that happen in order only once at the beginning. The **always** blocks specify simulation actions to take at certain events.

Please demonstrate to the professor that your simulation is working.

After simulation, you will need to create a bitstream for your genclk module. In order to properly test it on the bench, assign the **gclk** signal to a PMOD port (please consult the Basys3 manual at https://digilent.com/reference/programmable-logic/basys-3/reference-manual). Also, assign the input **clk** to the Basys3 100 MHz clk (again see the reference manual). Test the output with an oscilloscope: use wires to connect the probe ground to the GND line on the Basys3, and another wire to connect to the signal line on the scope probe. What clock frequency do you measure on the oscilloscope? What is the pulse width? Report these is your lab report. Also, include a scope image in your lab report of both measurements.

Please demonstrate to the professor that your circuit is working.

Experiment 2

For this experiment, you will develop a module to output the horizontal and vertical synching signals. The input clock signal CLK on the Basys3 board oscillates at 100 MHz. In order to achieve an approximate 60 Hz screen refresh rate, you must create a set of counter circuits that generates the following clock signals:

- Pixel clock: 25 MHz
- Horizontal clock (HSYNC): 31.25 kHz
- Vertical clock (VSYNC): 59.98 Hz

Please declare a Verilog module called `VGAController` with the following definition:

```
module VGAController(input clk, input rst, output hsync, output vsync);

// your code goes here

endmodule
```

You will need to use multiple counters to develop the three clock signals. Multiple implementation methods are possible. Some are easier than others. Also, you have to be extra careful to ensure that the pulse width is correct. Use experiment 1 as a guide for how to develop the **hsynch** and **vsynch** signals.

IMPORTANT: the **vsynch** signal should drop to zero precisely when the **hsynch** signal drops to zero. Use this information to ensure that everything is properly timed.

Once you verify your module on the testbench in simulation, demonstrate it working in FPGA to the professor.

The next step is to demonstrate the timing of the **hsynch** and **vsynch** signals from the FPGA on the workbench with the oscilloscope. Connect both signals to the PMOD port before generating your bitstream. Prepare two scope probes for testing your signals. You will need four wires (2 GND, 2 signal connections). Do the generated frequencies conform to the standard? Are the HSYNC and VSYNC signals synchronized properly? Include measured pulse width and frequency of both signals in the lab report as well as screen captures as proof.

Please demonstrate to the professor that your circuit is working.

Experiment 3

In this final experiment, you will add the red, green, and blue components to your VGA controller. Using the multiple counters from experiment 2 and a bit of math, you can calculate the pixel (x,y) position on the monitor. Please implement this pixel position calculation first.

Next, you need to generate a color image for the monitor. The RGB color scheme for the VGA monitor follows from computer graphics standards (e.g. 24-bit color for 8-bit RGB components). The Basys3 uses 12-bit color, 4-bits per component. As a simple test, try implementing a state machine that generates the flag of Russia or France. You will need to modify the module definition to output the color components as follows:

```
module VGAController(input clk, input rst, output hsync, output vsync, output
[3:0] red, output [3:0] green, output [3:0] blue);

// your code goes here

endmodule
```

Before you generate your bitstream, please connect your signals to the VGA port as specified in the Basys3 manual. Note: you should not connect to the PMOD port for this experiment. I recommend simulating a testbench to verify the first two rows of pixels.

Please demonstrate to the professor that your circuit is working.

For extra credit, generate a screen that outputs the flag of Japan to the monitor.

For double extra credit, add an input switch that allows you to select one flag or another.