Chapter 5

VGA Synchronization Controller

The purpose of this laboratory is to design a circuit for generating the VGA synchronization and timing signals. You will use this controller to create a “color bar” on the display. You will also use this controller in future labs to display other things on a VGA compatible display.

Learning Objectives

- Understand the timing signals required to drive a VGA display
- Understand how to create colors on the VGA display
- Create a reusable digital circuit for controlling the timing of a VGA display

Exercises

Exercise 1 – VGA Timing

Before proceeding, carefully read pages 10-12 of Nexys2 reference manual. This section describes the VGA port, VGA timing, and how to drive the VGA signals from the FPGA. After reading the pages in the Nexys2 reference manual, watch the following screencast to learn more about the VGA monitor display:

Screencast: VGA Overview

Much of the material in the manual and the screencast will be repetitive but it is essential that you understand how the VGA timing works in order to complete this lab and both resources are provided to help you understand how the VGA display operates.

Color Generation

One of the responsibilities of the VGA controller is to determine the *color* of each pixel in the display. Custom images, displays, and pictures are generated by strategicly setting the color of each pixel. A unique color is obtained by mixing the following primary colors: Red, Green, and Blue. The VGA cable contains one wire for each of these primary colors and the disiplay will sample the analog value of these three wires to deterimine what color to project. A higher voltage on one of these “color” wires will result in a color that is higher in intensity than a lower voltage. The VGA controller must determine the intensity of Red, Green, and Blue for each pixel by determining the analog value of these three primary colors.

A simple resistor network has been added onto the Nexys2 board VGA circuit so that you can generate the analog color signals with discrete digital values. There are eight VGA color signals available for you to set the current pixel color: three signals for red (RED0, RED1, and RED2), three signals for green (GRN0, GRN1, and GRN2), and two signals for blue (BLU0 and BLU1). The analog signal for each color is created by a resistor circuit network as shown in Figure 5.1.

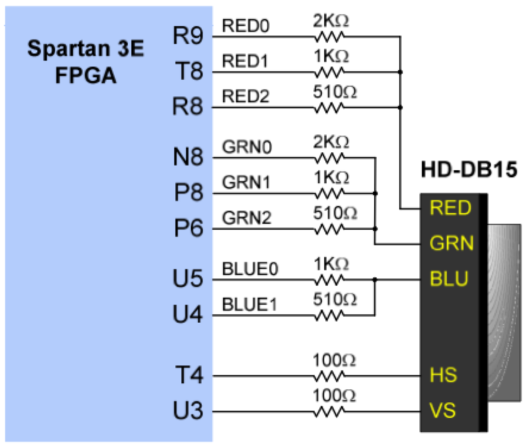


Figure 5.1: VGA Color Resistor Network.

With three digital signals for red, you can create 8 (2^3) different red intensities. If RED0, RED1, and RED2 are all set to zero, there will be no red color in the pixel (no red intensity). If RED0, RED1, and RED2 are all set to one, the pixel will have full red intensity. Intensities of red between the two extremes can be set by providing the corresponding three bit value for the REDx signals (where RED2 is the most significant).

Question: How many different colors can be displayed with this VGA controller?

Question: What values on the RED0, RED1, RED2, GRN0, GRN1, GRN2, BLUE0, and BLUE1 signals are needed to generate the "full intensity" red color (i.e., provide the maximum analog voltage on the RED signal and the minimum analog voltage on the GREEN and BLUE signals)?

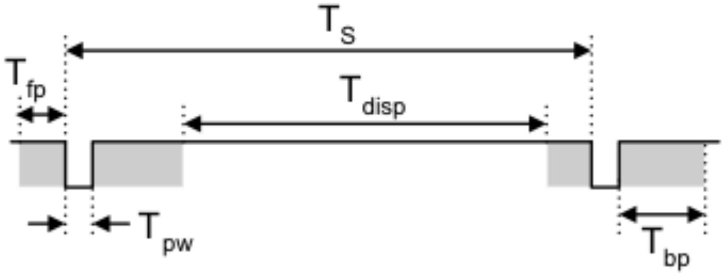
Question: What is the voltage on the Green analog signal if the values of Green are: GRN0=1, GRN1=1, and GRN2=0? (this is a simple voltage divider problem. Assume that the input impedence to the VGA cable is very high)

VGA Horizontal Timing vspace.1in

To display an image on the VGA display, the VGA controller must generate the proper timing signals for the given VGA display format. Each display format has a different set of timing parameters and the VGA controller circuit can generate displays with different formats by changing the timing signals. You will create the timing signals for the 640x480 VGA display format. This format displays 640 unique vertical columns, 480 unique horizontal lines, and a total of 307,200 pixels (640 columns $\times$ 480 lines). Each pixel on the display can be referenced by its x,y cordinates where x is the column (starting at 0 on the left and incrementing as you move to the right) and y is the line (starting at 0 on the top and incrementing as you move down).

The most important timing paramter of the VGA controller is the pixel clock rate. The pixel clock rate determines the rate at which the pixels are displayed. The pixel clock rate for the 640x480 format is 25 MHz. This means that each pixel will be displayed for 1/25 MHz = 40 ns. All other timing signals are derrived from this pixel clock.

The VGA controller must generate the "Horizontal Sync" signal (known as **HS**). The horizontal sync signal is a periodic signal that indicates to the display the start of a new horizontal line, the frequency of horizontal line, and the time in which the display should be blanked. There timing of the horizontal sync is summarized in Figure 5.2.



(a)

Horiz. Sync		Symbol	Parameter
Time	Clks		
32 us	800	T_S	Sync pulse
25.6 us	640	T_{disp}	Display time
3.84 us	96	T_{pw}	Pulse width
640 ns	16	T_{fp}	Front porch
1.92 us	48	T_{bp}	Back porch

(b)

Figure 5.2: VGA Horizontal Sync.

As indicated in the table, the HS signal has a frequency of T_S = 32 us or 800 pixel clocks. This means that a new horizontal line will be displayed every 32 us. Only 640 pixels are displayed during this time. The rest of the time is involved in a process called "retrace" in which the display is getting ready to display the next line. No pixels are displayed during this retrace period.

To correctly drive a VGA monitor, the VGA controller must generate a horizontal sync signal with four phases as shown in Figure 5.2. The timing of each phase must be strictly met. These four phases are as follows:

- **Display Time**. This is the time in which the VGA display is displaying pixels on a horizontal line. HS must be high during this phase which lasts 640 pixel clocks or $T_{disp} = 25.6$ us.
- **Front Porch** This is the time after the display period in which retrace begins. HS must be high during this phase also and this phase is 16 pixel clocks or $T_{fp} = 640$ ns.
- **Sync Pulse** The HS signal is low (asserted) during this phase and the signal must be low for 96 pixel clocks or $T_{pw} = 3.84$ us.
- **Back Porch** The final phase is the backporch. The HS signal must be high during this phase which is 48 pixel clocks or $T_{bp} = 1.92$ us.

The sum of these four phases is 800 pixel clocks or 32 us. The HS signal must continuosly sequence through all four of these phases. You will generate a circuit that generates this repeating sequence.

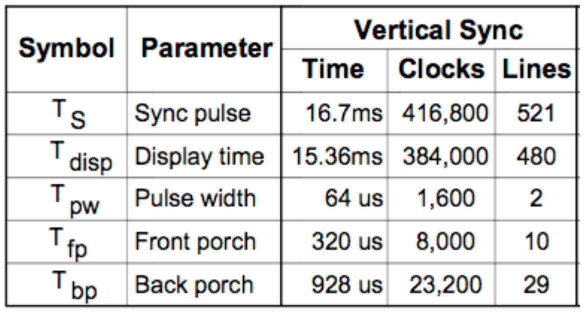
Question: What is the frequency of the pixel clock for the 640x480 VGA resolution?

Question: What is the frequency of the horizontal sync signal (HS) for the 640x480 resolution?

Question: During the scan of one horizontal row, 640 pixels are displayed. However, additional time is needed during the horizontal scan for retracing. How many pixel clocks are needed during each horizontal sync for this retracing (i.e. how many pixel clocks are used when no pixel is displayed in a horizontal scan)?

VGA Vertical Timing

In addition to a horizontal sync signal, the VGA controller must also generate a vertical sync signal (VS). The vertical sync signal indicates to the display the end of a full frame of display. Like the horizontal sync signal, the vertical sync signal includes the following four phases: display time, front porch, sync pulse, and back porch. The timing of these signals, however, is different and is summarized in Figure 5.3.



Symbol	Parameter	Vertical Sync		
		Time	Clocks	Lines
T_S	Sync pulse	16.7ms	416,800	521
T_{disp}	Display time	15.36ms	384,000	480
T_{pw}	Pulse width	64 us	1,600	2
T_{fp}	Front porch	320 us	8,000	10
T_{bp}	Back porch	928 us	23,200	29

Figure 5.3: VGA Vertical Sync Timing.

The timing for the vertical sync is specified in terms of "lines" rather than pixel clocks. The display time, for example, is specified as 480 lines. With each line requiring 800 pixel clocks, the display time is 480 lines/frame $\times$ 800 pixels/line $\times$ 50 ns/pixel = 19.2 ms. The other phases are also specified in terms of full 800 pixel lines.

Question: How many frames per second are generated with this timing?

Question: How many lines are NOT displayed during a full frame (i.e. lines that are blanked during a vertical retrace)

Do NOT proceed to the design until you have answered the questions above. If you do not understand the timing of the VGA controller and are not able to answer these questions you will not be able to proceed with your VGA timing controller design.

Exercise 2 – VGA Timing Controller

Begin your VGA timing controller by creating a new VHDL file named `vga_timing.vhd` that contains the entity for the timing generator for your VGA controller. Create an entity with the following ports:

Port Name	Direction	Width	Purpose
clk	Input	1	Input clock (50 MHz)
rst	Input	1	Asynchronous reset
HS	Output	1	Low asserted horizontal sync VGA signal
VS	Output	1	Low asserted vertical sync VGA signal
pixel_x	Output	10	Indicates the column of the current VGA pixel
pixel_y	Output	10	Indicates the row of the current VGA pixel
last_column	Output	1	Indicates that the current pixel_x correspond to the last visible column
last_row	Output	1	Indicates that the current pixel_y corresponds to the last visible row
blank	Output	1	Indicates that the current pixel is part of a horizontal or vertical retrace and that the output color must be blanked. The VGA pixel must be set to "Black" during blanking.

Table 5.1: Entity Interface for the VGA Timing Controller.

After you have created your entity, create your architecture that implements the VGA timing. Follow these guidelines as you create your VGA timing controller. Address each of these items in order.

VGA Clocking

For the 640x480 VGA resolution that we are implementing, the pixels change at a frequency of 25 MHz (this is called the "pixel clock"). This is half the frequency of the 50 MHz clock that is provided with our board. There are a few ways to address this issue. One way is to create a 25 MHz clock by dividing the 50 MHz clock and use this new 25 MHz clock for the VGA timing circuitry. While this is a fairly straight-forward way to solve this problem, creating a 25 MHz clock and introducing it into the circuit introduces other timing challenges that you are not ready to resolve.

For this lab, we will use a different approach. This approach, described in Section 9.1.3 of the textbook, uses the 50 MHz clock to generate a synchronous enable signal. This enable signal will toggle every other clock cycle to allow the VGA timing registers to change every other clock cycle of the 50 MHz clock. If the registers change every other clock cycle, the registers will behave as if it were driven by a 25 MHz clock. A block diagram of this approach is shown below in Figure 5.4.

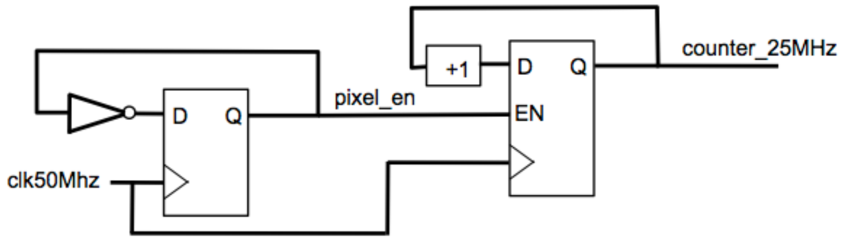


Figure 5.4: 25 MHz Pixel Clock Circuit.

Create a signal named "pixel_en" that can be used by internal synchronous circuitry to "enable" the updating of VGA timing counters. With a 50 MHz input clock, this pixel_en signal should be asserted every other clock cycle. During one clock cycle, pixel_en is asserted and all VGA timing counters are updated. During the next clock cycle, pixel_en is de-asserted and all VGA timing counters keep their old values. This synchronous signal should be set to zero when the rst signal is asserted.

Horizontal Pixel Counter

Create a horizontal pixel "column" counter. This counter is used to sequence through the various phases of the horizontal synch pulse (i.e., display, front porch, pulse, back porch, etc.). This counter should reset to zero when rst is asserted. Also, this counter should only update when the pixel_en signal is asserted as described above. This counter should be designed to count through the full sequence of pixel columns including the retrace pixel columns. For the 640x480, this counter should count from 0 to 799 (i.e. 640 displayed pixels, 16 front porch pixels, 48 back porch pixels, and 96 "pulse" pixels). When this counter is 0, it indicates that the first visible pixel is being displayed. When the counter value is 639, the counter is indicating that the last visible pixel is being displayed. When the counter is greater than 639, it indicates a column associated with the retrace phase. The output of the counter should used to drive the pixel_x outputs of your synchronization module.

Decode the horizontal counter to generate the HS sync signal. You will need to refer to the timing diagrams in the board user's manual to determine when the HS sync signal should be asserted (do not forget that HS is low-asserted). Remember to order your back porch, pulse width, and front porch correctly. Decode the horizontal counter to generate the "last_column" output signal. This signal should be asserted when the last visible pixel in a row is being displayed (i.e. when the current column is 639).

Vertical Pixel Counter

Create a vertical pixel "row" counter. This counter is used to sequence through the vertical sync phases and indicate the current line being displayed. This counter should reset to zero when rst is asserted. Like the horizontal counter, this counter should only update when the pixel_en signal is asserted (the counter should hold its value when pixel_en is not asserted). In addition, you should only increment this counter when your horizontal counter has a value of 799 (i.e., increment the vertical counter when the horizontal counter transitions from 799 to 0).

When the vertical counter is 0, it indicates that the first visible line is being displayed. When the counter value is 479, the counter indicates that the last visible line is being displayed. The output of this counter should be used to drive the pixel_y outputs of your module.

Decode the vertical counter to generate the VS sync signal (again, refer to the reference manual for vertical timing). Also, decode the vertical counter to generate the "last_row" output signal. This signal should be asserted when the last visible row is being displayed (i.e. during row 479).

Blank Signal

Create the logic for the “blank” signal. This signal indicates that the color should be blanked (i.e., display black). Assert the blank signal when the current pixel_x, pixel_y location is not a part of the visible region of the screen. This signal will be used by other modules to determine when a black color should be given on the RGB color signals.

After creating your VHDL, remove all compilation errors and perform a few simple tests to see if the circuit is working as you expect it to. Write a simple .tcl script to simulate the behavior of your circuit. When you are satisfied with the operation of your circuit, test your module with the testbench shown below.

Upload: Submit your .tcl simulation script

Like the previous labs, a testbench has been provided for you to test your VGA timing controller. This testbench will simulate your VGA timing controller and measure your signals to see if they follow the appropriate timing. This testbench is very picky about timing and will complain if there are any timing problems. If the testbench gives you errors, spend some time to understand the error and resolve the problem.

Download: tb_vga_timing.vhd

Begin by simulating for only a few microseconds. If you have problems with your horizontal timing, you will see messages in a short amount of time. After resolving any of your horizontal timing problems, simulate for a longer period of time to see if your vertical timing is correct. Continue simulating and debugging until you do not see any error messages from the testbench. You will need to simulate for a relatively long period of time until the “simulation done” message appears.

Upload: Submit your working VGA timing controller

Exercise 3 – Top-Level Design and Synthesis

The VGA controller you used in the previous exercise is not intended to be used as a top-level design but will be used as a reusable component within other top-level designs that incorporate a VGA display. For this final exercise, you will create a simple top-level VHDL file that instances the VGA controller and adds logic to determine the colors to project on the display. Begin your top-level VGA color generator by creating a new VHDL file named `vga_top.vhd`. Create an entity with the following ports:

Port Name	Direction	Width	Purpose
clk	input	1	System Clock
btn	Input	4	Button Inputs
sw	Input	8	Switch Inputs
Hsync	Output	1	Low asserted horizontal sync VGA signal
Vsync	Output	1	Low asserted vertical sync VGA signal
vgaRed	Output	3	VGA Red color
vgaGreen	Output	3	VGA Green color
vgaBlue	Output	2	VGA Blue color

Table 5.2: Entity Interface for the Top-Level VGA Controller Circuit.

After creating the top-level entity, instance your VGA controller into the top-level design. The high-level architecture of this top-level design is shown in Figure 5.5. Attach the top-level clock to the clock of your VGA controller and attach button #3 to the reset input of your controller. When button #3 is pressed, your VGA timing controller will be reset and start over.

The HS and VS signals generated within the VGA controller are generated with combinational logic. As such, these signals will likely contain temporary, short-term “glitches”. For a synchronous system, these glitches are usually acceptable. In this system, however, these signals are used by the monitor to synchronize the display with your pixel data. To facilitate proper synchronization, it is important that there are no glitches in these synchronization timing signals. Glitches on the VGA timing signals may lead to improper VGA synchronization and improper colors on the display. To prevent these glitches, create a register for both the HS and VS signal as shown in Figure 5.5. Drive the top-level output Hsync and Vsync ports with the output of these registers.

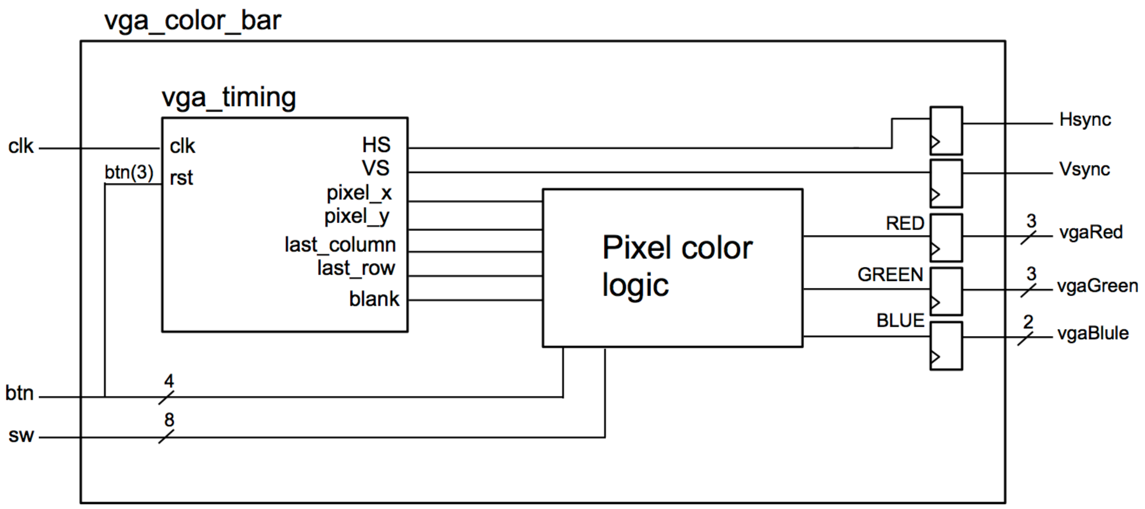


Figure 5.5: Architecture of Custom Image Circuit.

The VGA timing controller that you created in the previous exercise will generate the two VGA timing signals, VS and HS. This timing controller, however, does not generate the pixel color signals (RED[2:0], GREEN[2:0], and BLUE[1:0]). The values of these signals will determine what is displayed on the attached monitor. You will create additional circuitry that works in conjunction with your VGA timing controller to specify the color of each pixel on the VGA display. This pixel generation logic, however, will reside in the top-level file. You can create different VGA display functionality by reusing the VGA timing controller and creating custom pixel display logic.

In this top-level design, you will create logic that displays a single color on the screen based on the values of the switches and buttons. To do so, you will need to determine the values of each of the eight color signals (RED[2:0], GREEN[2:0], and BLUE[1:0]) under all input conditions. The first constraint on the pixel color signals is that you must force the value of the pixels to “Black” (Red=“00”, Green=“000”, and Blue=“00”) when the VGA timing controller is in a “blanking region” (i.e., when the `blank` signal from the VGA timing controller is asserted). The following VHDL code demonstrates how you can force the color “Black” on the color outputs during the blanking regions using a simple multiplexer structure:

```
red <= red_disp when blank = '0' else "000";
green <= green_disp when blank = '0' else "000";
blue <= blue_disp when blank = '0' else "00";
```

In the example shown above, all eight pixel color signals will be assigned ’0’ when the VGA timing controller `blank` signal is asserted. When the `blank` signal is `not` asserted, the signals `red_disp`, `green_disp`, and `blue_disp` determine the value of the output pixel color. You will need to create additional logic to determine the values of these signals under the following conditions (in priority order):

- When button 2 is pressed, display the color “Red” (i.e., Red= “111”, Green = “000”, and Blue= “00”).
- When button 1 is pressed, display the color “Green” (i.e., Red= “000”, Green = “111”, and Blue= “00”).
- When button 0 is pressed, display the color “Blue (i.e., Red= “000”, Green = “000”, and Blue= “11”).
- When no buttons are pressed, display a single uniform color on the screen where the color is determined by the eight switches (Red=SW[7:5], Green=SW[4:2], and Blue=SW[1:0]).

Note that button #3 should be hooked up to the asynchronous reset signal of your VGA timing controller and when pressed will hold the controller in the reset state. When reset, the timing controller should not generate any VGA timing signals and the VGA monitor will lose synchronization with your circuit. Because your VGA timing controller is not operating when this button is pressed, it does not matter what values you drive on the output color signals when this button is pressed.

The logic for creating the pixel values will contain glitches and will need registers on the output ports as was done with the synchronization signals (see Figure 5.5). Once you have created the logic for your color signals, add a flip-flop for each pixel output and attach the output of the flip-flop to the corresponding output port as shown in Figure 5.5.

After completing your top-level VGA display circuit, simulate the circuit to verify that your logic displays the proper color values on the screen. This simulation may take some time, as you will need to simulate at least one VGA frame for each of the button conditions described above. Once you are satisfied with your circuit behavior, create a .ucf file that contains pin location definitions for each of your I/O ports (you will need to create new entries for the 10 VGA signals). Synthesize your VGA display circuit and proceed to download.

Question: What is the minimum clock period of your circuit (review the "Post-PAR Static Timing Report" and search for "Minimum period")?

Question: Review the "Map Report" and determine the number of "slices" used by your design. Personal Exploration

Personal Exploration

Spend some additional time in this lab by exploring new concepts, ideas, and design modifications for your VGA controller. Ideas for personal exploration include:

- Create a different display image on the VGA screen (different shape, pattern, etc.)
- Create a counter that sequences through the eight colors and displays each color for a brief period of time (i.e., 1 second each)
- Experiment with the monitor when non-standard timing signals are used

Pass Off

- Show your VGA timing controller testbench simulation (Design Exercise #2)
- Demonstrate a working color bar display on the VGA