Chapter 6

VGA Object Display

The purpose of this laboratory is to create a circuit that generates several objects on the VGA display. You will also learn how to use an internal logic analyzer circuit for debugging your circuit in real-time.

Learning Objectives

- Create simple objects on the VGA display
- Insert a logic analyzer circuit into your VGA display
- Use the logic analyzer circuit to monitor signals within your circuit in real-time

Exercises

Exercise 1 – Color Bar Display Logic

In the previous laboratory, you created a VGA display controller that displays the same color on all pixels of the 640 x 480 display. More interesting images can be created by displaying different colors at different pixel locations within the screen. These unique images can be displayed by creating custom logic that uses the the `pixel_x` and `pixel_y` counter outputs from the VGA timing module to determine the current pixel color. Interesting objects, images, animation, and user interfaces can be displayed by creating custom logic based on the x,y location of the VGA display.

As a simple example, an image that is red on the left half of the display and green on the right half of the display can easily be created by assigning the pixel color based on the value of the `pixel_x` counter. The VHDL code that would generate such an image is shown below:

```
red_disp   <= "111" when pixel_x < 320 else "000";
green_disp <= "000" when pixel_x < 320 else "111";
blue_disp  <= "00";  -- blue is always zero for this particular display
```

In this example, when the pixel_x counter is less than 320 (i.e., the left half of the screen), the pixel color will be red (Red="111", Green="000", and Blue="00"). When the pixel_x counter is 320 or greater, the pixel color will be green (Red="000", Green="111", and Blue="00"). More interesting and complicated images can be created with more sophisticated logic. You will create several different VGA displays in future laboratory exercises.

For your this exercise, you will create the logic for a vertical color bar like the one shown in Figure 6.1. Color bars were often used in analog televisions to test the television and the television transmission. For this exercise, create the VHDL logic to generate a vertical color bar that displays eight colors as shown in Table 6.1. The width of each color bar should be 80 pixels (i.e., 640/8 = 80) and each color bar should span the full height of the display (480 pixels high).

Upload: Submit the VHDL statements required to produce a vertical color bar based on the pixel_x and pixel_y values.

Color	R	G	B	Columns
Black	“000”	“000”	“00”	0-79
Blue	“000”	“000”	“11”	80-159
Green	“000”	“111”	“00”	160-239
Cyan	“000”	“111”	“11”	240-319
Red	“111”	“000”	“00”	320-399
Magenta	“111”	“000”	“11”	400-479
Yellow	“111”	“111”	“00”	480-559
White	“111”	“111”	“11”	560-639

Table 6.1: Color Values and Column Assignments for Color-Bar Display.

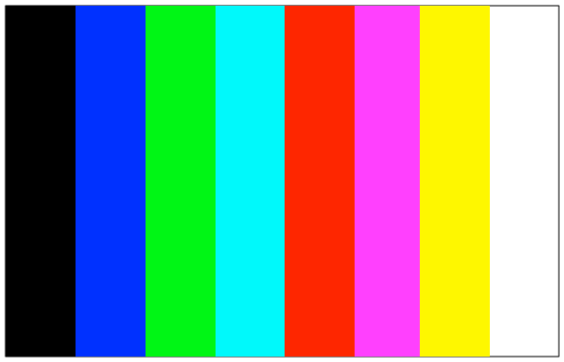
Figure 6.1: Color Bar Pattern.

Exercise 2 – Creating Multiple Objects on the Display

In the previous exercise, you created a top-level circuit that generated a simple color bar pattern on the VGA screen. You accomplished this by decoding the current value of the VGA horizontal and vertical counters (pixel_x and pixel_y). For this exercise, you will display multiple independent objects that share the VGA screen. To draw an object on the display, you will need to create logic that indicates when the object should be displayed. For example, the following VHDL defines a rectangle that is 100 pixels wide and 50 pixels high. This rectangle is displayed in the region: 150 <= `pixel_x < 250` and 100 <= `pixel_y` < 150:

```
rectangle_on <= '1' when pixel_x >= 150 and pixel_x < 250 and
                         pixel_y >= 100 and pixel_y < 150 else
                '0';
```

In addition, logic is needed to specify the color of this rectangle. The following logic specified the color blue for the rectangle.

```
rectangle_rgb <= "000" & "000" & "11"; -- blue
```

The following logic define a green square that is 50 pixels wide and 50 pixels high:

```
square_on <= '1' when pixel_x >= 350 and pixel_x <400 and
                      pixel_y >= 200 and pixel_y <250 else
             '0';
square_rgb <= "000" & "111" & "00"; -- green
```

Other custom drawing circuits could be created to draw a variety of objects.

Since the VGA display is controlled by only one RGB output, additional logic must be created to determine which object to display. In some cases, objects may overlap each other and logic must be created to determine which object has priority for display. Multiplexing logic must be created to select the appropriate color value to display at the current location based on the various object display circuits. The following example displays the rectangle and square with the rectangle having higher priority. Note that this VGA display logic will generate a "black" background where the rectangle and square are not displayed.

```
-- Default value when no object is being displayed
background_rgb <= "00000000";

rgb_out <=  rectangle_rgb  when rectangle_on = '1' else
            square_rgb     when square_on    = '1' else
            background_rgb;
```

For this exercise, create the VHDL logic to implement the four squares shown in Figure 6.2. The specification of this image is as follows:

- Draw a red square that is 100 pixels wide and 100 pixels high. The top-left corner of this square should be placed at the pixel 180,100.
- Draw a green square that is 100 pixels wide and 100 pixels high. The top-left corner of this square should be placed at the pixel 400,100.
- Draw a yellow square that is 100 pixels wide and 100 pixels high. The top-left corner of this square should be placed at the pixel 180,300.
- Draw a magenta square that is 100 pixels wide and 100 pixels high. The top-left corner of this square should be placed at the pixel 400,300.
- Provide a white background when none of the objects is being displayed.

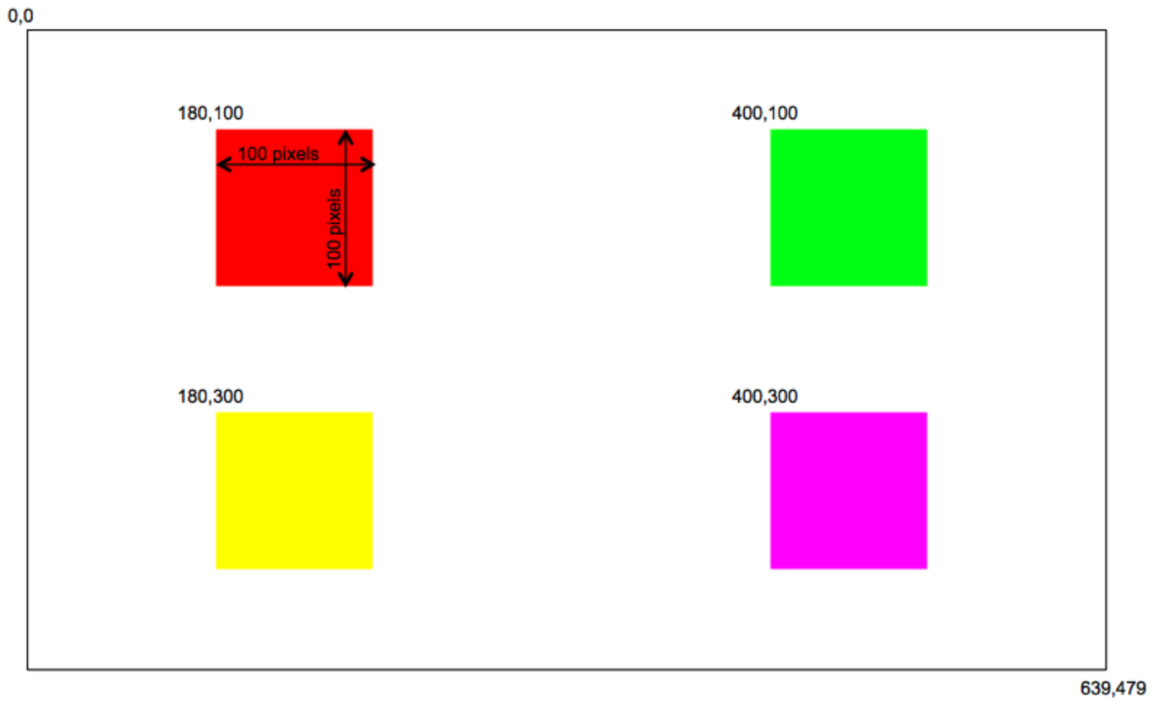

Figure 6.2: VGA Squares Pattern.

Upload: Copy and paste your VHDL for this object display.

Exercise 3 – Top-Level Design Exercise

For this laboratory, create a single top-level design named `vga_object` that includes the ports shown in Table 6.2.

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
seg	Output	7	Seven Segment Display Cathode Signals
dp	Output	1	Seven Segment Display "DP" Cathode Signal
an	Output	4	Seven Segment Display Anodes Signals

Table 6.2: Entity Interface for the Top-Level VGA Object Display.

Begin the architecture of your top-level design by instancing your VGA controller in much the same way you did in the previous lab (see Figure 5.5). Connect the top-level clock to your controller clock and connect button #3 to the reset input of the controller. Connect the HS and VS synchronization signals to flip-flops and then to the top-level output pins.

Next, instance the seven-segment display controller and connect the outputs of your controller to the top-level sevent-segment display output pins. Create a 16-bit counter that counts the number of frames being displayed. A new frame occurs when both the `last_column` and the `last_row` signals of your VGA timing controller are asserted. Note that these signals will be asserted for two clock cycles since the VGA timing module is only updated every other clock cycle. You should only increment your frame counter once per frame. Attach the output of your 16-bit frame counter to the inputs seven-segment display controller. When running, the seven-segment display should update at a frequency of 60 Hz (the VGA refresh rate).

The final step in this top-level design is to implement the logic for generating a custom image on the display. This logic will generate the values for the `vgaRed`, `vgaGreen`, and `vgaBlue` for the following conditions:

- When button 0 is pressed, display the four square pattern developed in Exercise 2.
- When button 1 is pressed, display an image with custom objects of your own making. Your image must have at least two distinct objects each with a different color.
- When button 2 is pressed, display a solid color on the screen with the color specified by the switches.
- When no buttons are pressed, display the vertical color bar pattern developed in Exercise 1.

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.

Once you have completed your top-level design, compile your design and remove all syntax errors. Simulate your design using a simple .tcl script to provide a stimulus for the clock, the buttons, and the switches. When you are satisfied that your top-level VGA object display circuit is working properly and displaying the correct pixel color values, proceed to the next exercise.

Exercise 4 – ChipScope Internal Logic Analyzer

For circuits like the VGA object display, it takes many clock cycles to fully verify that the circuit is performing correctly. Ideally, full simulations can be performed to fully test the circuit in a simulation environment. In some cases, it is difficult or impossible to fully simulate the circuit and other methods for verification are needed. Another approach for verification involves the use of a logic analyzer to watch the actual signal values of your circut in real time. To probe your circuit with a logic analyzer, you would need to attach probes to the output pins of your circuit. The logic analyzer would then record the values on these signals and provide a waveform for your review.

One disadvantage of logic analyzers is that you have limited access to signals within the circuit that do not attach to output ports. One method for accessing internal signals is to use a logic analyzer *within* your FPGA circuit. FPGA vendors provide custom circuits for you to use to help you debug your circuits by allowing you to capture the internal signals Xilinx offers a tool called "ChipScope" that performs this function. For this exercize, you will instance a ChipScope logic analyzer within your circuit and perform some in-system verification.

An example VHDL file has been created for you to demonstrate how to include the ChipScope logic analyzer into your design. Download this file and review the contents of the file before including the ChipScope into your design.

Download: chipscope_example.vhd

The ChipScope example provided in the VHDL file contains two components that must be instanced into your design. The first component, `ICON`, is the ChipScope controller and controls the operation of the logic analyzer. It also provides the ability to transfer logic analyzer data over the USB JTAG cable to your computer. The second component, `ILA`, is the actual internal logic analyzer. This component captures the signals you want to view and stores them in memory until you are ready to view them. The structural relationship of these two components is shown in Figure 6.3.

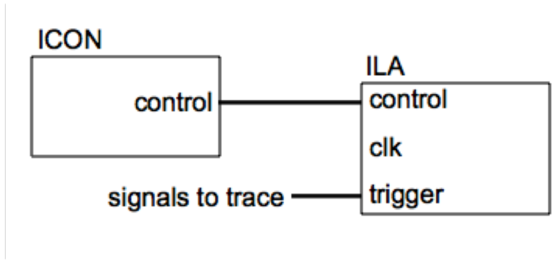

Figure 6.3: ChipScope Architecture: ICON and ILA Components.

To include these components into your design, add the component declaration of both modules into the declaration section of your top-level VHDL architecture. Next, instance both components into your design and attach the control signal between them. Attach the global clock to the `ILA` component. The last step is to create a 32-bit signal that contains all of the signals you want to analyze with the internal logic analyzer. Create a new 32-bit signal that concatonates all of the signals listed below. Attach this signal to the `trigger` input of the `ILA`.

- VGA color signals
- pixel_x
- pixel_y
- VS
- HS
- blank

The following code sample demonstrates one way to create this trigger signal:

```
trigger <=  '0' & blank & hs_int & vs_int &
               pixel_y & pixel_x & pixel_value;
```

The last step required to insert the ChipScope into your design is to add the `chipscope_icon` and `chipscope_ila` to your project. Unlike previous labs, you will not add the VHDL files for these components. Instead, you will add a special file, called a `.ngc` file, that contains the implementation details of each component. Download both `.ngc` files and add them to your project. The "?" associated with each component in your project viewer should go away when they are added.

Download: chipscope_icon.ngc

Download: chipscope_ila.ngc

When you have completed the insertion of the ChipScope logic analyzer, synthesize your design and make sure it builds properly.

Upload: Copy and paste your VHDL for your top object display design.

Exercise 5 – ChipScope Debugging

The final exercise in this laboratory assignment is to download your VGA object display circuit and perform several ChipScope verification activities. Begin this exercise by downloading your top-level circuit and viewing the VGA display. Experiment with your circuit to see if it produces the VGA display that you expect.

Once you have experimented with the VGA display, begin your ChipScope verification activities. Begin by watching the screencast on the ChipScope ILA tool.

Note that this screencast is currently not available. Instruction will be given by the laboratory TA on how to use the ChipScope logic analyzer tool.

After you have familiarized yourself with the ChipScope tool, perform the following exercises:

Question: Use the ChipScope waveform viewer to determine the number of clock cycles of the Horiztonal Sync signal

Question: Determine the range of values of pixel_x when the blank signal is asserted during a horizontal refresh

Make sure the vertical color bars are being displayed for the following experiments.

Question: Trigger the logic analyzer at the beginning of the frame (i.e., pixel_x and pixel_y = 0). What color is being displayed at this first pixel?

Question: Trigger the logic analyzer when the “Cyan” color is begin displayed. Indicate the range of values of pixel_x during one horizontal line.

Press button 0 to display the “square” pattern on the VGA display for the following experiments.

Question: Trigger the logic analyzer when the “Red” color is being displayed on line 100. Indicate the value of pixel_x and pixel_y when this first Red pixel is displayed.

Question: Trigger the logic analyzer when the pixel_y value is 350. Determine the values of pixel_x in this horizontal line when the pixel color is “Yellow”.

Personal Exploration

The lab has a lot of fun opportunities for personal exploration. For your personal exploration, modify the design to add some additional visual feature. Ideas include:

- Create a pattern or picture in the background of one of your display images.
- Provide some different functionality on the display when the buttons are pressed.
- Perform several additional ChipScope verification experiments.

Pass Off

- Show your VGA object display to the TA
- Show the TA your personal exploration
- Demonstrate your ChipScope waveforms