Laboratory Exercise 7

Memory and VGA Display

The purpose of this exercise is to learn how to create and use on-chip block random access memories (BRAMS) as well as use the video graphics adapter (VGA).

Preparation Before the Lab

You are required to complete Parts I to III of the lab by writing and testing Verilog code and compiling it with Quartus II. Show your schematic, Verilog, and simulations for Parts I to III and state diagrams for Parts II and III to the teaching assistants. You must simulate your circuit with ModelSim using reasonable test vectors.

In-lab Work

You are required to implement and test all of Parts I to III of the lab. You need to demonstrate all parts to the teaching assistants.

Part I

The FPGA provides flexible block RAMs that can be varied in bit-width and depth along with many other parameters. In this part of the lab exercise you will create a block ram verilog modules using the Quartus IP catalog and test the memory module with switches and hex displays for inputs and outputs.

The circuit we would like to create is outlined in Figure 1. It consists of a memory block, address register, data register and control registers.

Figure 1: Schematic of BRAM test circuit.

A timing diagram showing a write and a read to the memory is shown in Figure **??**.

Figure 2: Sample timing diagram for the BRAM test circuit.

Perform the following steps:

1. Open Quartus
2. Select Tools-¿IP Catalog
3. Open Installed IP-¿Library-¿Basic Functions-¿On Chip Memory-¿RAM:1-PORT
4. Choose the path and name of the Verilog module and file to be created.
5. Select an 8 bit wide memory with 256 words.
6. Unselect q to be registered. Select create a ‘rden’ signal.
7. Click Finish and save the new Verilog file.
8. Examine the newly created Verilog file.

9. Create a new Quartus project and add the new Verilog file along with your additional code.
10. Draw a schematic describing the circuit and explain it to the TA as part of your prelab.
11. Simulate your circuit with ModelSim for a variety of input settings, ensuring the output waveforms are correct. You must show this to the TA as part of your prelab.
12. Compile the project.
13. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit.

Part II

In this part, you will design a circuit to draw a *filled* square on the screen at any location in any colour. As in the previous part, your circuit will accept an x and y location as input, as well as a colour. The circuit should then draw a 4×4 pixel square whose *upper-left* corner is at the (x,y) location specified by the input. The top half of the square (2 pixels high $\times$ 4 pixels wide) should be filled with the colour specified by the switches and the other half of the square should be filed by the complement of the colour. For example, if the 3-bit colour specified is *110* then the complement colour is *001*. The filled square should be drawn when an enable button (active-low) is pressed. An active-low reset should cause the entire screen to be cleared. Squares that straddle the right edge or bottom of the screen should be truncated (i.e. they do not wrap around). After a square has been drawn, your circuit should allow additional squares to continue to be drawn (say, at different locations in possibly different colours). The high-level design of the circuit for the system is given in Figure 3. It contains 3 major blocks:

1. The VGA adapter responsible for the drawing of pixels on the screen.
2. The datapath that contains arithmetic circuitry and registers, controlled by the FSM, that compute the (x,y) values that are fed into the VGA Adapter in order to draw the 4×4 filled square.
3. A finite state machine that serves as a controller for the datapath.

Figure 3: Design Overview - State Machine, Datapath and VGA Adapter. Although not shown, ResetN signal should be connected to all the registers in the circuit (including FSM state register).

Part III

In this next part we will create a simple animation. We will create a circuit that takes a small image and moves it around the screen. To accomplish this, your circuit will have to make it seem as though the image is seemlessly moving around the screen. You will implement the circuit in two steps. First, you will design a module that is able to draw (or erase) the image at a given location. Then you will design another module that moves the image around the screen by quickly redrawing it at the different locations.

To implement this part, you will have to create a circuit that takes as input the (screen_x,screen_y) coordinate of where the top left corner of the image is to be drawn or erased.

The circuit will then either draw the image from the memory (LPM_RAM_DQ module) or erase the image starting at postion (screen_x,screen_y). TODO: need to add MIF file tutorial for BRAM load. Images are drawn or erased pixel by pixel. To draw an image, read a single entry from memory and pass that value to the "color" line on the VGA adapter. During this process, you must set the (x,y) values on the VGA adapter to the appropriate values. Erasing an image is a similar process, however, instead of passing the values from the "Image RAM" to "Color", you will set "Color" to black (constant 000). To accomplish these steps, you will need to create a state machine that performs the following:

1. Set Counter_X and Counter_Y to 0.

2. While Counter_X is less than 16, either load a pixel value from memory containing image.mif, or set the pixel value to black (if erasing). Then draw that pixel at location (screen_X+Counter_X,screen_Y+Counter_Y) on the screen. Increment Counter_X.

3. If Counter_Y is less than 15, then increment Counter_Y and set Counter_X to 0. Go to step 2.

4. Stop when Counter_Y reaches 16.

The suggested circuit diagram is shown in figure 4. The "Blank" and "Plot" inputs may be used in several different ways. One way is to have "Blank" input select whether the image should be drawn or erased (i.e., if the "Blank" is high when "Plot" is asserted, the image is erased). Another way is to have two separate inputs, one to start drawing ("Plot"), and another to start erasing ("Blank"). You are free to choose whichever method you want.

Implement the circuit by completing the following steps:

1. The project for this part is provided in the starter kit. Open the project named *part3* in the *part3* subdirectory to begin your work.

2. Create a 16x16 bitmap image that is to move around the screen. Make sure to set the image width and height to 16 pixels.

3. Use the bmp2mif.exe converter to convert the image into an MIF file. Call it image.mif.

4. In your design instantiate a memory using an LPM_RAM_DQ and use image.mif as its memory initialization file.

5. Create a circuit to draw an image at a specified location on the screen as discussed above.

6. Compile the circuit and download it onto the DE2 board. When your circuits starts you should be able to see the image you have drawn somewhere on the screen.

Figure 4: Suggested circuit layout for drawing the image

To complete the circuit, perform the following:

1. Create a circuit to change the location of the top left corner of where the image is to be drawn, once every 60^{th} of a second. You may use circuit shown in figure **??** as a reference.

2. Put the circuits together to see if your image moves around the screen. Compile the circuit and program it onto the FPGA to see if it works.

3. If the circuit works, think of a way to get the image to bounce of the sides of the screen as it moves about. Implement the enhancement and show the circuit to your Teaching Assistant.