ECE318 Laboratory #4
Lab Date: Feb 5th Report Due: Feb. 12th

VGA Experiments

1. Introduction

This laboratory is based on the material in Chapter 10 of "Rapid Prototyping of Digital Systems". Though you may think that a controller for a VGA monitor is a complex digital system, it turns out to be a fairly simple digital circuit. The timing is critical to the correct operation of the circuit and is based on a 25Mhz clock Which is generated from the 50 MHz clock o the DE2 board using a PLL). The secret to the VGA format is to redraw the entire screen of pixels which is 640x480 quickly enough so that the eye does not notice the remapping, i.e., there is no evident flicker. A refresh rate of 60Hz accomplishes this goal. In this laboratory, we will use digital logic blocks to create character display and also implement a bouncing ball program.

2. Prelab

Read Chapter 10 to understand the generation of VGA signals. Section 10.4 details the generation of horizontal and vertical synchronization signals as well as red, green and blue signals which can be sent to a VGA monitor. Examine the code carefully and make sure that you understand it. Sections 10.7 - 10.10 describe in detail the way in which characters are generated and you should read through these carefully. Section 10.14 gives the implementation of the bouncing ball.

NOTE: Download the zipped directory DE2Core and uncompress it on your thumb drive in your ECE318Lab directory. Add this and the .vhd file for the symbols to each of your projects by using “Tools – Options – General – Global User Libraries”.

3. Procedure

We will again be using Quartus and the DE2 Cyclone II for the logic synthesis. The code for the logic modules is available on the DVD which was handed out in class and on the WEB site. Today's lab will consist of four experiments that will help you understand the VGA controller. Create a subdirectory for each of the experiments on your thumb drive and download core files from the WEB site.

Experiment 1: Sync generation

A VHDL sync generation module is detailed in section 10.4 on page 196 of the lab textbook. Create a Quartus project and a Block diagram/ Schematic file. Add a VGA_SYNC block from the DE2Core library and add pin numbers (see Figure 1 below). Compile and download to the board. Describe the behavior. Look at the code from this section and describe the purpose of each of the 5 assignments at the top of pg. 198.

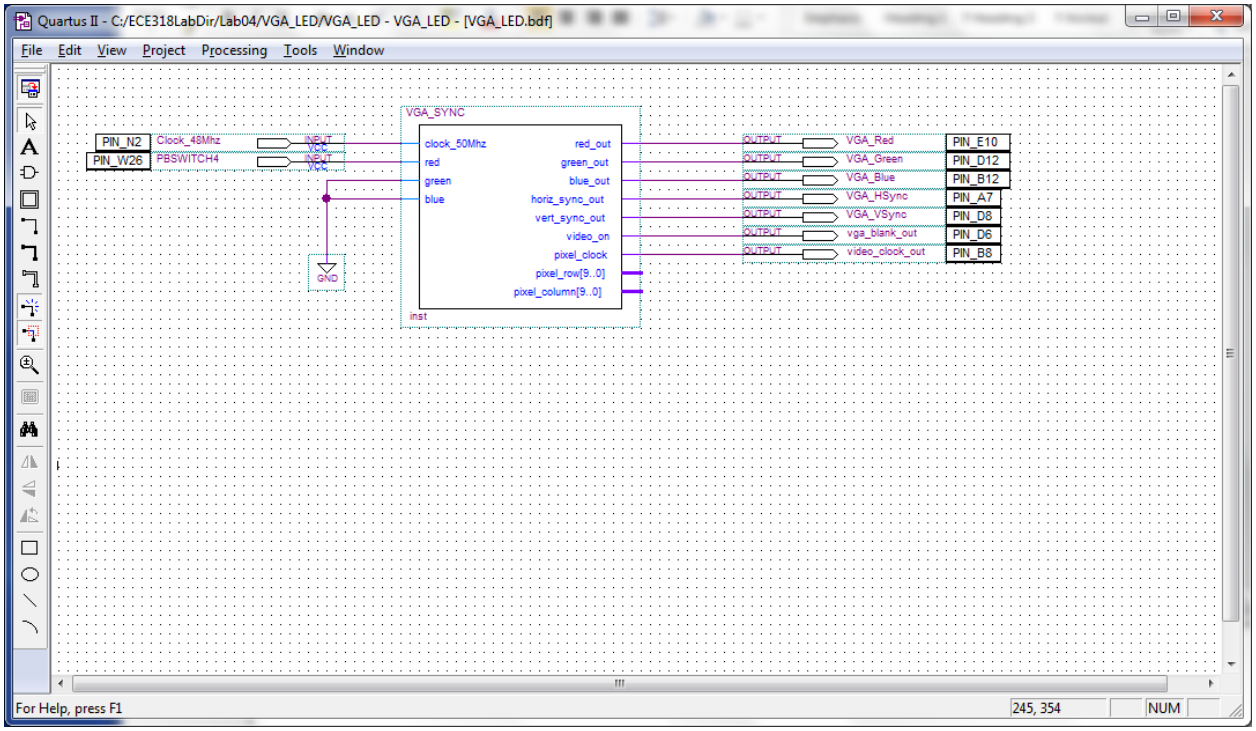

Figure 1

Experiment 2: Color Generation

Read through section 10.7 and implement the circuit as shown in the second figure of this section at the top of page 200 and also shown below in Figure 2. Create a folder and a project called VGA_BAR. What is the sequence of colors generated from left to right? Show your lab instructor once you have this working.

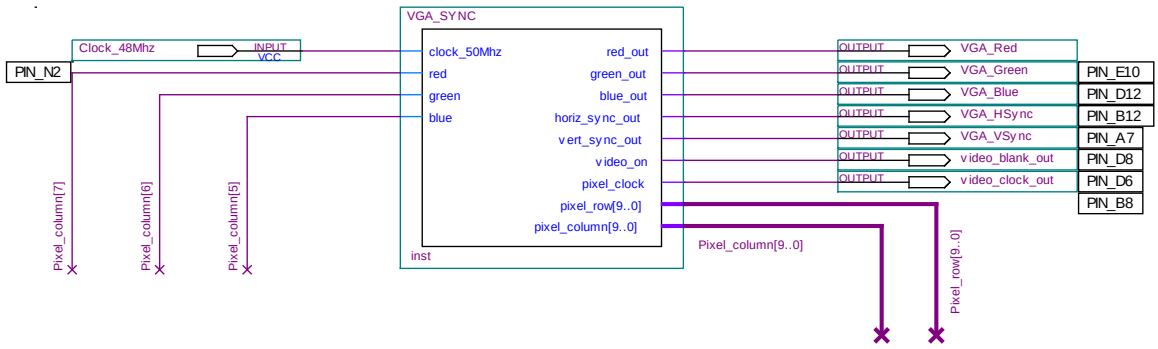

Figure 2

Experiment 3: Character Generation

Implement the character display given in section 10.10 on pg. 203 and in Figure 3 below. Here are some notes:

- The character is an 8x8 grid and each pixel in the grid must be read to paint out the character. If the font_row and font_col inputs to the char_rom block remain constant while the pixel_row and pixel_column values are changing (these are determined by the VGA_SYNC block), then the same pixel value (1 or 0) will be put in each of the pixel_row and pixel_column combinations.

- Note that the inputs to font_row and font_col are pixel_row[3..1] and pixel_column[3..1]. Why is this done and what effect does it have?

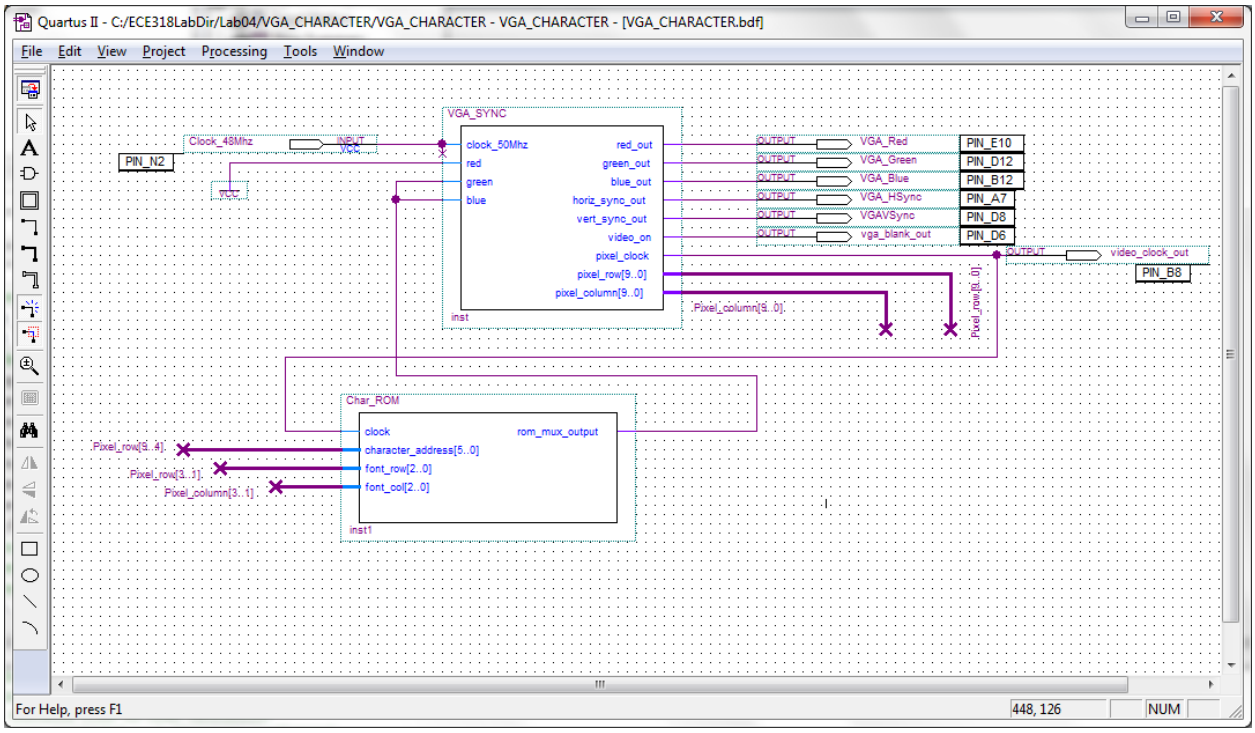

Figure 3

Make the characters larger so that they are now 64x64 in size. This is question 1 on page 210.

- There are 6 bits used to define which character is output to the screen. These 6 bits are defined in **octal** in table 10.2, pg. 203. This means that G, for example, has the address 07 which corresponds to "000111".
- Think about the code you might use to solve this problem then look at code on the ECE318 web site logopolis.union.edu. Create a symbol for the code and add it to your character generator as shown in Figure 4 below.

ii) Using just two characters (your initials) have them occupy almost the full screen. Here are some tips:

- Recall that to paint 1 pixel value in an area of the screen the font_row and font_col inputs should remain constant over this entire area. So, if you break up

the area of the screen into a grid, you can specify what the values of font_row and font_col should be in this area and pass these to the char_rom block. The output of this block is a pixel value which is passed to the VGA_SYNC block.

- From the VHA_SYNC code on pages 144-6, note that if you include the libraries std_logic_arith and std_logic_unsigned then comparisons can be made in the following manner:
  h_count is std_logic_vector(9..0) and the comparison
  if (h_count <= 755) and (h_count >= 659) then
  compiles fine.
  In truth, this is a little loose as the inclusion of the unsigned package will allow this comparison and the string will be interpreted as unsigned. If you want to be more precise, a more formally correct way of doing a similar comparison is:
  if (unsigned(h_count) = 799) then
  This may be useful to you in your implementation.
- You will see that there are functions used throughout the code from std_logic_arith and other packages. You can find a link to these packages (where there is a description of what each function does) on the class website.

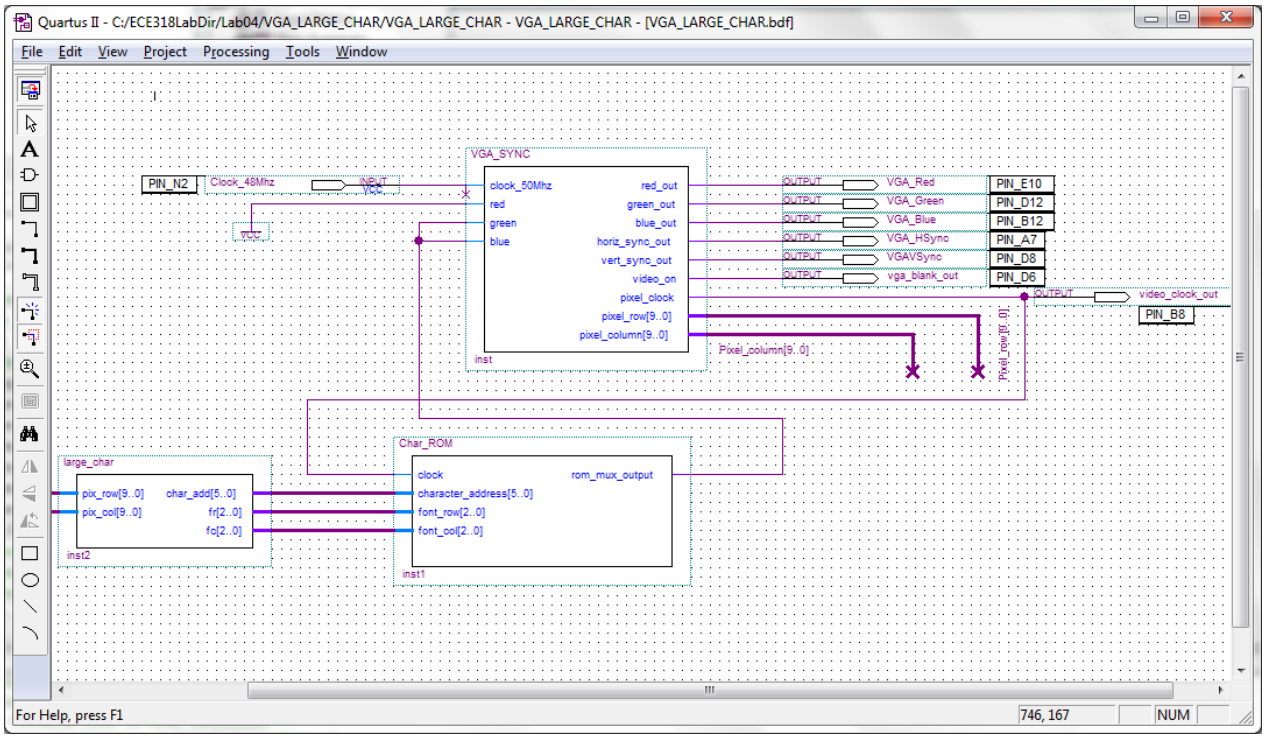

Figure 4

Experiment 4: Bouncing Ball Modification

Open a new Block Diagram File and insert the ball symbol file into the diagram as well as the VHD_SYNC.

You will now take this block and add some inputs to it and output the lines to the VGA monitor as we have done previously. Here is what the final design file should look like:

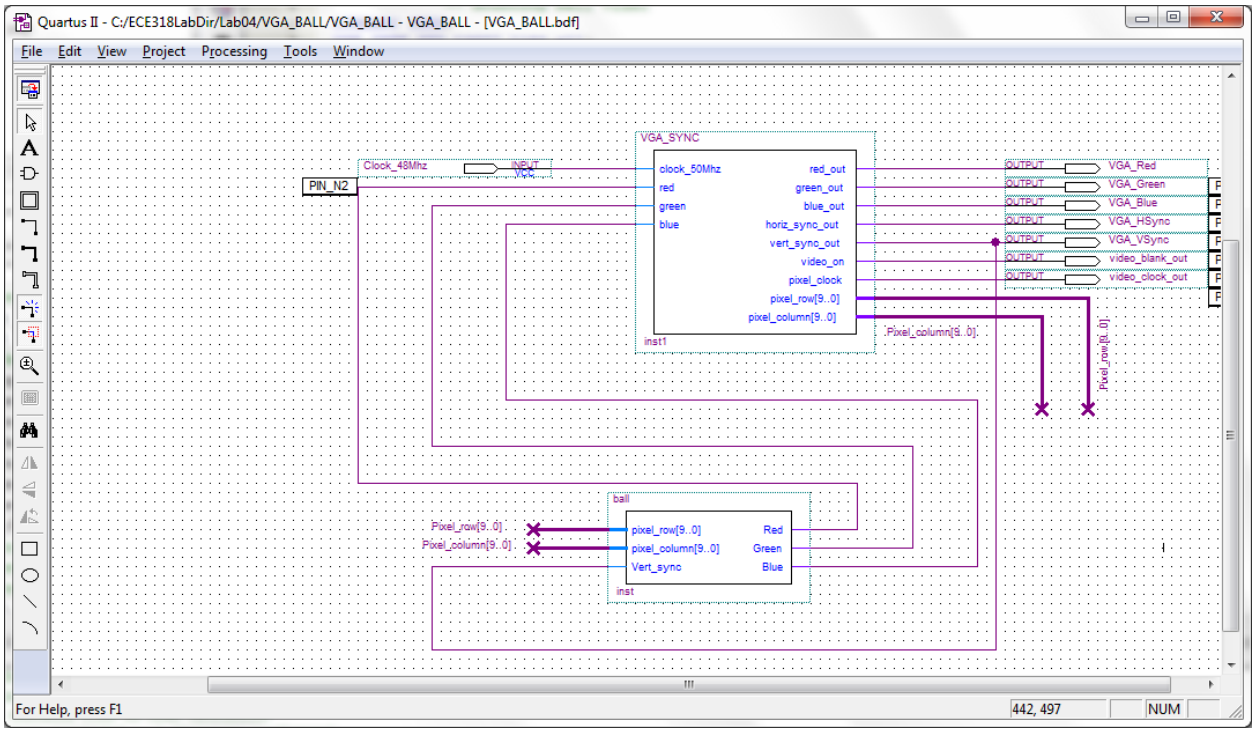

Figure 5

When you compile and download to the board, you should be amazed by a square bouncing ball.

Show your instructor the working demo. In your lab report, detail **in your own words** how this code is working.

Think about how you would complete laboratory exercise 2 on pg. 210. In your lab report, outline a sketch of how you would implement this change. This a potential avenue for projects for the course – once you have a bouncing ball, you can perhaps think of some fun games.

4. Lab Report

Follow the normal lab report format. Be sure to include answers for parts (a) and (b) in your report. Include the code listings for the implementations in parts (c). In addition, include an outline of a solution to exercise 2 on page210 [this is a thought experiment and you do not have to implement the solution; rather, if you were to complete this task, how would you do it?]

Winter 2013