DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF BRITISH COLUMBIA
EECE 353 – Digital Systems Design

Lab 4: Particle Simulator on VGA display

Lab Overview

Marking for this lab consists of two parts:

- Preparation, which is done with your partner and submitted by 11:55pm on the due date
- Performance, which is done **individually** in the lab.

Introduction

The goal of this lab is to build a simple particle simulator that displays the movement of a particle in a box. The particle is drawn as a single pixel, or ball, moving on the VGA screen. You will learn how to slow down the clock to a reasonable value to control the motion. You will also learn how to control the movement of the particle depending upon different conditions.

The function of producing the VGA output signals is taken care of by a VGA adapter, which is provided to you as a black-box component. This means you do not need to know what is inside the VGA adapter, but you do have to know how to use it.

Overview of the VGA Adapter

To draw pixels on a screen, the VGA adapter must continuously send signals to the monitor. It redraws the entire screen every 60th of a second. During this 60th of a second, all of the pixel values are read from a memory inside the VGA Adapter and sent out to the VGA monitor. It then starts all over again. To save on the limited memory on DE2 board, the adapter has been setup to display a grid of 160x120 pixels. The grid is shown in Figure 1. The VGA Adapter starts by drawing one horizontal line at a time, starting from the top-left at position (0,0).

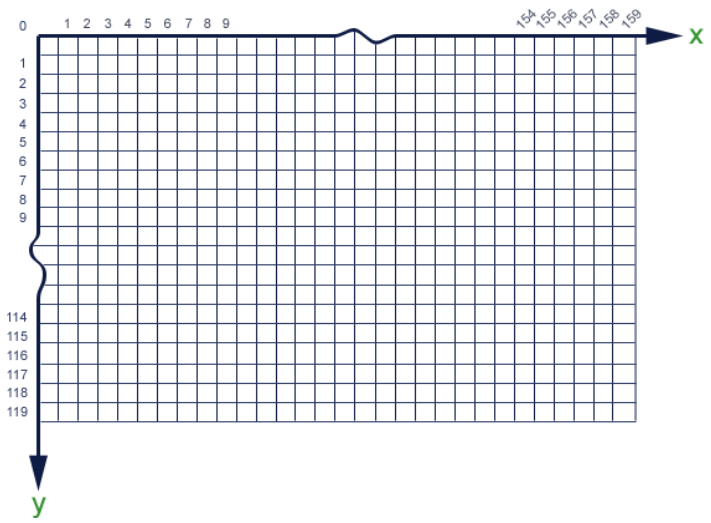


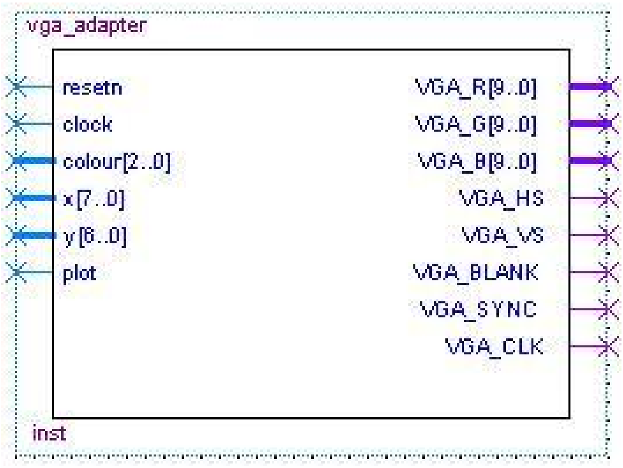
Figure 1: VGA adapter's display grid

The pattern of pixels to be drawn is stored in memory internal to the VGA Adapter. To change the contents of one pixel on the screen, you simply write a new pixel colour to the correct memory location inside the VGA adapter. The VGA adapter will remember the value written, and ensure the new pixel is drawn each time the screen refreshes.

The VGA adapter is written in Verilog. It consists of 4 Verilog files: vga_adapter.v, vga_controller.v, vga_pll.v and vga_address_translator.v. The first file is the top-level file. Open up the file into a text editor and see what Verilog looks like. However, don't let the differences scare you; you don't have to know Verilog in this course. You also don't have to learn the inner workings of the VGA adapter. However, you will learn to use it as a black box, where you use the inputs and outputs properly without knowing the internals. The inputs and outputs of the VGA adapter are shown in Figure 2 and described in the following tables.

More details about the VGA adapter can be found on University of Toronto's web page:
http://www.eecg.utoronto.ca/~jayar/ece241_07F/vga

Inputs
From your circuit
(You will be working with these)



Outputs
To DAC & Monitor
(You do not need to control these)

Figure 2: VGA adapter as a black box

Inputs:

resetn	Active low reset signal (reset when this signal is '0'). Digital circuits with state elements should always contain a reset.
clock	Clock signal. { CLOCK_50 } The VGA core must be fed with a 50MHz clock to function correctly.
colour(2 downto 0)	Pixel colour (3 bits). Sets the colour of the pixel to be drawn. The three bits indicate the presence of Red, Green and Blue components for a total of 8 colour combinations.
x(7 downto 0)	X coordinate of pixel to be drawn (8 bits) – supported values $0 \leq x < 160$.
y(6 downto 0)	Y coordinate of pixel to be drawn (7 bits) – supported values $0 \leq x < 120$.
plot	Active high plot signal. Raise this signal to cause the pixel at (x,y) to be set to the specified colour on the next rising clock edge.

Outputs:

`VGA_CLK`	VGA clock signal.

`VGA_R(9 downto 0)`
`VGA_G(9 downto 0)`
`VGA_B(9 downto 0)`
	Red, Green, Blue components of display (10 bits).
	These signals are connected to the Digital-to-Analog Converter (DAC) on the DE2 board before transmitting to the monitor.

`VGA_HS`
`VGA_VS`
`VGA_SYNC`
`VGA_BLANK`
	VGA control signals.

Testing the VGA Adapter

The Verilog files describing the VGA adapter can be included into Altera QuartusII just like VHDL files. If you read them, you will notice that the "module" definition in Verilog is similar to "entity" in VHDL. This means you can instantiate modules described in the Verilog files as components in your VHDL files.

An example is provided to you as the lab4_test.vhd file, which is part of lab 4 package available from eLearning website. This file provides the following ***input*** mappings (Note: you must use your DE2 pin assignments like the previous labs).

resetn	`KEY(3)`
clock	`CLOCK_50` This is an internal 50MHz clock that is generated by the DE2 board.
colour(2 downto 0)	`SW(17 downto 15)`
X(7 downto 0)	`SW(7 downto 0)`
Y(6 downto 0)	`SW(14 downto 8)`
plot	`KEY(0)`

Note that the ***outputs*** of the VGA core are passed straight through the lab4_test entity without modification.

Using the switches and pushbutton keys, you can change the colour of pixels one at a time. Be sure you try this test and understand how to control VGA Adapter before proceeding with the lab preparation.

Procedure

You need your DE2 board with a USB cable, a VGA cable, PC or laptop with QuartusII software installed and lab_4 package containing: vga_adapter.v, vga_controller.v, vga_pll.v, vga_address_translator.v and lab4_test.vhd. Add these files to a new QuartusII project, compile and program it on to your DE2 board. Connect the DE2 VGA output to a VGA display (most new LCD or LED displays have a 2nd input). The PCs in MCLD358 all have a spare VGA cable connected to their 2nd input. You can switch between the inputs using the display's input select button. Note: the VGA connector on your laptop is an **output**, so don't connect your laptop VGA to the DE2 VGA!!!

Incremental steps:

1. Modify lab4_test.vhd and add your code to plot some pixels at different (x,y) coordinates.
2. Use a state machine to plot a sequence of pixels. Note, you do not need to randomize things, use the switches to set the x/y coordinates (e.g. you could keep one constant and vary the other).
3. Draw a single-pixel particle and have it move in a constant direction, travelling either in a horizontal line or a vertical line. For it to appear to "move", you must first erase the pixel by setting it to the same colour as all of the background pixels, then you draw the pixel at the new location by setting it to the desired colour. Try to avoid a "blinking pixel" effect by only erasing and redrawing the pixel if the x,y position has changed.
4. When the particle hits the wall (edge of the VGA screen where x=0, x=119, y=0 or y=159), have it reflect off of the edge with a perfect reflection. Notice that when the ball hits a wall, the magnitude of its speed (Δx, Δy) does not change. However, it must continue in a mirror image of its previous direction (e.g. when it hits the ceiling or the floor its Δx does not change but its Δy changes direction but with the same magnitude)

Hint: at a given time, your ball will have some position **p** = [x,y] and will be moving by some velocity vector, **v** = [Δx, Δy], to its next location. Each time step (eg, 1/60th of a second), you should move the particle by addition, i.e.: **p** = **p** + **v**.

You can initiate each particle to start from a fixed point; for example from the center of the display (59,79). The <u>larger</u> the Δ**x** the <u>faster</u> the **ball** moves horizontally and the <u>larger</u> the Δ**y**, the <u>faster</u> it moves vertically.

Note 1: Once you have done with initial tests (step 1 and some of 2) you have to control the VGA adapter's X/Y input dynamically (not with slider switches).

Note 2: If you use the push button switches to trigger the ball's movement, you have to press it very fast. And if you use one of the clocks, you have to divide it down significantly so the pixels do not appear/disappear in less than 1/30th of second (precision of human eye). With a fast clock, you will need a wide counter to slow things down (clock divider). To slow things down, you can generate a 1-cycle pulse every time a counter reaches a certain value – use this pulse to enable forward progress in your state machine. **Do NOT GENERATE a second clock signal. Be sure to keep using <u>the same clock signal</u> throughout your VHDL code.**

Prelab [5 marks]

You must hand in a **zip file** implementing what is described in this section.

Modify your step 4 to create 2 balls bouncing from the walls, with ball1 at (x1, y1) and ball2 at (x2, y2), as shown in Figure 3. Name your file **lab4_2balls.vhd**. Please put comments in your VHDL files starting with **BOTH** your Names and Student Numbers and the description of your code and your design method. Please also submit a **short video** of your DE2 board and the VGA display showing the working prelab. **Be sure to clearly identify yourselves in the video**.

You do not need a high-end camera. A low resolution 15 frames/sec video captured by most cellphones is enough, as long as your pixels are not moving faster than 1/15th second. Put all your files in one zip file (lab4.zip) and submit on the eLearning WebSite assignment for your "lab 4 Preparation".

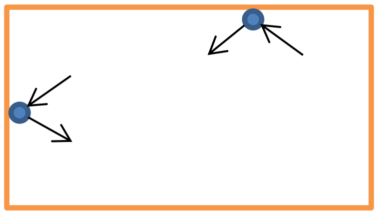


Figure 3: Prelab sample image of 2 moving particles.

Performance Requirements [5 marks]

In the lab you will receive the in-lab requirement sheet from the TA. After demoing your working preparation to the TA, you can do the required modification. Please keep a copy of your preparation code for the demo. You can work on the performance part while waiting to do the demo.

Procedures

The Performance segment will operate as follows:

- **Phase 1**: You have only **2hr 15min** to satisfy the Performance Requirements. During this time, the TAs will provide help (in the lab test, they will not help you).
- **Phase 2**: You have another **15min** to upload your solution to the eLearning website, to finish your final QuartusII compilation, and to **set up your DE2 board for demonstration** to a TA.
- If you finish early, you must wait for Phase 3, but you can ask to be marked first.
- **Phase 3**: During the last **30min**, all students will step away from their computer and wait to be called one-by-one for grading by the TA. Once your name is called, you must give your demo and be graded within 3 minutes. **You will not be given any additional time**.