COE4DS4 – Lab #4 Report

Group 18
Khalid Asad (1147299) and Vishal Kharker (1140837)
asadk@mcmaster.ca, kharkevp@mcmaster.ca
February 11, 2016

EXERCISE 1:

With this exercise, we partitioned the display area into 10x15 rectangles of size 64x32 pixels each. By declaring an array to hold all the values of the various colours, we can play around with the colours and keep those colours in memory. The array was initialized with values from 0 and decrementing as we go right. The cursor has two variables for the x and y co-ordinate. If the cursor is at the top left corner, *cursor_position_x* is equal to 16 and it is 32 pixels wide up to 48, whereas *cursor_position_y* is equal to 8 and it is 16 pixels long up to 24. When the cursor moves in any direction, the cursor positions will be incremented or decremented by the width/length. If the cursor position goes beyond the boundaries, it will be set to the value of the opposite boundary. For example, if it is in the top left corner, and key 0 (up) is pressed, then *cursor_position_y* will be set to 480 – 24 = 456 (ranging from 456 to 472). If key 4 or 5 are pressed, the current square will increment or decrement colours respectively in the array. Following that key press, the value of the array is checked, and if it is 7 before it is incremented, the value becomes 0, and vice versa for decrementing. Then a function is called to check if all the colours in any given row are all the same. This is done by a nested for loop that checks if the first column of a row is equal to all of the other columns in that row, incrementing a counter each time it is true. If the counter reaches the max, print the row and colour. Finally, the function to draw the bars is called, simply writing the values of the array to the LCD for each square.

EXERCISE 2:

During this exercise exercise it was crucial to create new registers in order to store values that are being passed from NIOS. To implement these registers we made changes to the following files:

Nios_LCD_Camera_Component.sv

Nios_Imageline_Interface.v

Filter_Pipe.v

Experiment.c

The interface.v file was used to extract the data that was passed through the interrupts and store them in registers. The buffers for these registers were then created in the camera_comppnent file this allowed us to update filter and passed the values to filter_pipe.v. The coefficient calculations were carried out in filter_pipe.v we simply used a right shift by either 0, 3 or 4 bytes depending on the requested weight sum. Instead of implementing loops, we had blocks of code to define our set0 to set6, all having individual values for themselves.