# COE 4DS4 – Lab #4 Report

Group Number: 33
Moshiur Howlader / 001316948 / howlam@mcmaster.ca
Ryan Ganeshan / 001322407 / ganesr3@mcmaster.ca
February 7th, 2018

**Exercise 1:**

This exercise was a direct extension of experiment2 in-lab. Achieving the functionality specified was done by first implementing a flag that will detect key 3. When key 3 is detected, we first do a Y conversion and then call our 2d edge detect function (it is stated that this edge detect function should operate on a grayscale image). The 2D edge detect function was implemented in a similar way to the LPF filter function that was done in lab. The challenge was to keep track of multiple line data, instead of single line. We took care of this by implementing a buffering system. We had buffers for each previous, current and next line contents of grayscale data. We had a for loop i with an inner loop j. The i loop would iterate through lines and the inner j loop would iterate through columns within the line. We would write the new 2d-edge detect filtered pixels at the end of every line iteration. We would read in a new line at the beginning of every new line, and would shift “up” grayscale line data in a shift register fashion. We used the “memcpy” function to take care of memory copying between our array buffers. Finally, border cases were taken care of within both the i loop and inner j loop. The first line and last line were part of the border case (i==0 and i==numlines-1), in addition to the right and left-most columns of the panel (j==0 and j==LineLen-1). In these cases, the border pixel value was replicated, as instructed. We were able to see working edge detect filter once complete.

**Exercise 2:**

Main challenge of this exercise was to pass the set of coefficient values for the filter pipe from Nios in real time. The files we changed were *experiment3.c*, *Nios_Imageline_Interface.v*, *Nios_LCD_Camera_component.sv*, and *Filter_Pipe.v*. The main architectural flow that was used was from NIOS -> Imageline -> Filter_Pipe to pass the filter coefficients. The filtered equation from the pipe would be displayed on the LCD panel. The important change made in experiment3.c was the iterator to cycle through and pass the six sets of coefficient values when key 3 was pressed. The coefficients were packed into 32 bit representation, with the coef_set_x[0] packing the first four values of the set, then coef_set_x[1] packing the next four values of the set, then finally coef_set_x[3] packing the last two elements of the set (with zero padding on the 16 most MSB). We have arbitrarily decided to write these coefficient values into register 6 to 8 (since they do not conflict with other existing registers for the Imageline peripheral). Before writing to these registers, we ensure to write filter_config as 5 so that later down the flow, we know that key 3 has been pressed, and we need to apply the coefficients for the given filter coefficient set. For Nios_Imageline_Interface.v, since we are writing the coefficients from NIOS directly to it, we simply read the write data, and write that into the new output coefficient ports that will later be passed down into the filter pipe circuit. Some changes in the Nios_LCD_Camera_component.sv were also required. Since our implementation needed to be real time filtering, we needed to update the coefficients that we passed into the filter_pipe at V_sync edge. Hence, we used buffers to buffer them in at V_Sync edge. Some additional wire connections for the ports were made. Finally, in Filter_Pipe.v, we have computed the filter equation as well as the threshold logic using signed combinational arithmetic to produce the edge detect saturation that can update real-time with the press of key 3.