Minilab 2 Report - Horizontal and Vertical Edge Detection

Verilog Files Written and Used

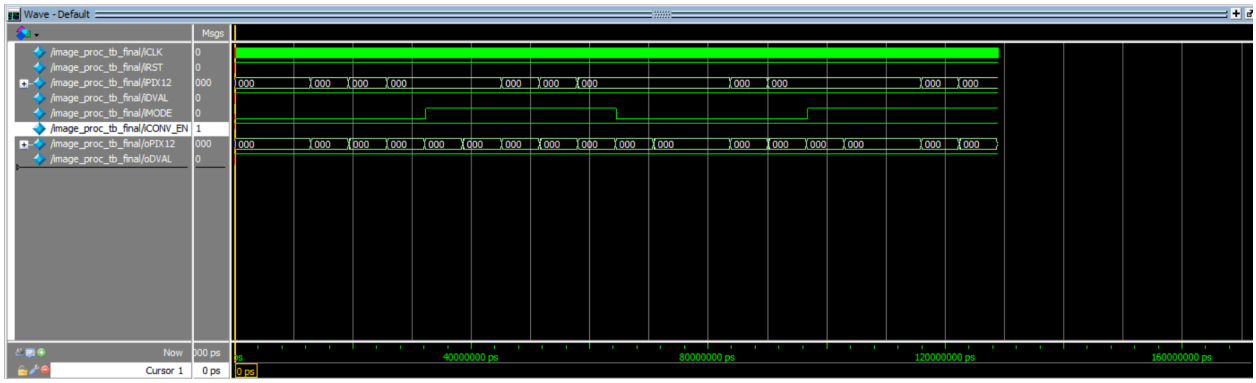
Image_processing.sv: This module processes a 3x3 convolution filter for 12-bit grayscale videos. This processing module utilizes our gray line buffer which was created using an IP for a memory based shift register to access y1 and y2 rows. After building the 3x3 window, a signed filter multiply is done with unsigned pixel handling to ensure no faulty sign extension is done. The computed magnitude is an absolute value of the convolution sum, which is used if valid.

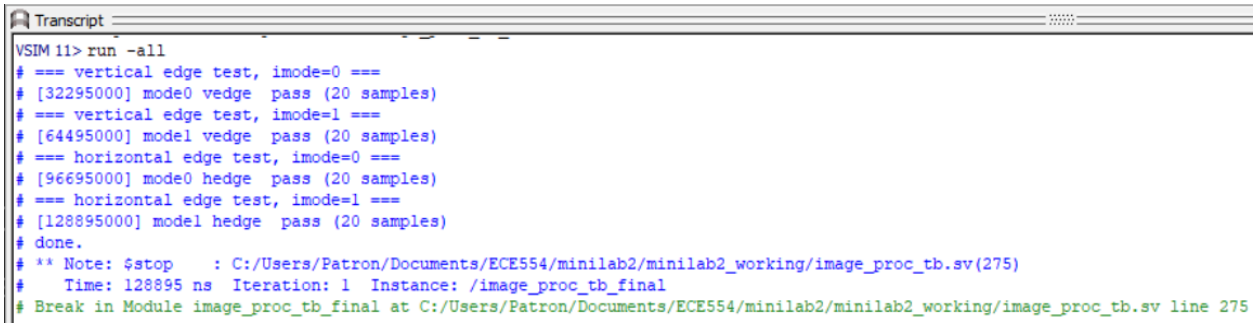
Gray_Line_Buffer_640.v (IP): This is the IP we used for the gray scaling buffer that was 12 bits wide, 640 bit distance, and two taps. We had each tap have its own output for easier pipelining and use of data.

DE1_SoC_CAMERA.v (modifications): Very little was modified in the top level. A selector MUX was added to determine which data is written to the Sdram_Control u7 module. If no switches were turned on, then the regular colored output is selected. SW[1] enabled would change to the gray scaled data. SW[2] would demonstrate convolution, and SW[3] would determine whether horizontal or vertical edge detection is shown. SW[3] = 0 would mean horizontal edge detection is shown, and 1 would mean vertical edge detection is on.

The image processing module was instantiated to obtain the edge detection data, and the gray scaling was calculated manually inside of the top level.

Testbench





image_proc_tb.sv: Builds the expected arrays of data for both vertical and horizontal edge detection and only tests our image processing module. Test data is sent to the image processing module and is compared to the expected results.

Flow Summary and Utilization

Flow Summary.rpt and DE1_SoC_CAMERA-Resource Usage Summary.rpt: Our flow summary showed only 2% logic utilization (710/ 32,070) which seems very small for large data convolution. Our total DSP blocks used was 9%, which seems high compared to the logic utilization. This could be due to the resources allocated for the convolution being allocated to the DSP blocks, due to the high performance required. Total block memory bits was 74,656 and 1220 dedicated logic registers were used. This makes sense due to the large buffers and us having RGB, grayscaling, horizontal, and vertical edge detection data was stored, and we simply chose the data to write to memory. This could definitely be optimized.

Analysis & Synthesis Resource Usage Summary	
Resource	Usage
Estimate of Logic utilization (ALMs needed)	767
Combinational ALUT usage for logic	1184
-- 7 input functions	12
-- 6 input functions	239
-- 5 input functions	135
-- 4 input functions	160
-- <=3 input functions	638
Dedicated logic registers	1220
I/O pins	226
Total MLAB memory bits	0
Total block memory bits	74656
Total DSP Blocks	8
Total PLLs	4
-- PLLs	4
Maximum fan-out node	sdram_pll:u6|sdram_pll_0002:sdram_pll_inst|altera_pll:altera_pll_i|outclk_wire[0]
Maximum fan-out	581
Total fan-out	11170
Average fan-out	3.68

Problems Encountered and Solutions

Amer: We encountered an issue in our image processing module that would make the processing module only output the gray scale data, and after some debugging, we found that

each bit stream was “invalid”, so our selection would always choose the gray scale output. We found a better implementation that enabled or disabled the convolution output internally, which solved some difficult timing issues.

Chance: Figuring out how many taps to use for the buffer was tricky, but after looking at the top level and how the RGB image was formed onto the screen, we figured the correct configuration (2 taps 640 bits apart).

Anirudh: Understanding the top module seemed really hard at first but after figuring out the data path for the camera, it made the top level data flow much easier to understand. This made us more confident in implementing a camera feature for our project proposal.

Munasib: Testing the image processing module seemed easy, but coming up with a clever way to generate test data was hard. After fully processing the convolution, and understanding how vertical and horizontal differ in terms of calculation, and just how the bit streams looked made it easier.