M Sadman Sakib, Jingyu Liu

Minilab 2 Report

Implementation:

Our image processing module implementation follows the provided RAW2RGB module in the beginning. We are also using the provided IP module Line_Buffer. The bayer pixels captured by the camera are received as an input to the image processing module. When the pixels received are valid, we take the first two pixels of two rows (like a square) and average their RGB values by summing them up and dividing by four or right shifting by two. That gives us the greyscale pixel values. We then store those greyscale pixels in two buffers using Line_Buffer so that we have access to two rows of greyscale pixels. Then we have defined a 3x3 convolution window array where we shift in the greyscale pixels from the buffer. This array values later gets multiplied by the desired sobel operators. We have two different convolution output based on vertical edge detection and horizontal edge detection. In the top-level module, we can switch between RGB, horizontal edge detect, and vertical edge detect using FPGA switches. We did not have much problem implementing the module. After running it the first time, we found a small mistake that we did not connect the new greyscale valid signal at top level properly. Once we fixed that, the demo worked fine.

Testbench:

We have a simple image processing testbench that generates two rows of bayer pixels which are sent to the image processing module, and we match the output pixel values with expected values. When we ran the test bench it passed normally.

Quartus Flow report:

From the report we can see that 2 % of ALM, 3% of BRAM and 0% of DSP were used. The report makes sense as there was not any heavy calculation used in this project. Convolutions also had very small coefficients which did not need DSP to compute. There is not much data being stored either as most data is being shifted out as output in real time.

Github link: https://github.com/jy2247/minilab2.git