DSP - Final Project

Hasith Perera

May 2023

1 Introduction

Today image processing is commonly used in science and engineering from medical imaging to astronomy. This project was focused on implementing an Edge detection algorithm for 2D images. A modified version of the Canny edge detector which is often named as a *Canny-Deriche edge detector* was chosen due to its design and suitability to be implemented in a FPGA.

1.1 Canny-Deriche edge detector

Mathematically the process can be described using the following set of equations. It should be noted these represent two passes of the image from left to right (equation 1) and right to left (equation 2). As far as the implementation is concerned the only difference between *Stage 1* and *Stage 2* are the filter coefficients and the same function (in the case of a CPU implementation) can be reused to process the data.

A second pass is needed on the second dimension. This could be reinterpreted as passing the image through the same stages with a 90° flip. These symmetries makes implementing the full detector boil down to implementing equation 1 through 3 and calling them as needed. These filters can be run simultaneously on multiple rows if needed and this makes the algorithm attractive since it can be easily parallelized to get a significant speed up.

Note: Filter coefficients are tabulated as an appendix at the end

- Stage 1

$$y_j^1 = a_1 x_j + a_2 x_{j-1} + b_1 y_{j-1}^1 + b_2 y_{j-2}^1 \tag{1}$$

$$y_j^2 = a_3 x_{j+1} + a_4 x_{j+2} + b_1 y_{j+1}^2 + b_2 y_{j+2}^2 \tag{2}$$

$$\theta_j = c_1(y_j^1 + y_j^2) \tag{3}$$

- Stage 2

$$y_j^1 = a_5 \theta_j + a_6 \theta_{j-1} + b_1 y_{j-1}^1 + b_2 y_{j-2}^1 \tag{4}$$

$$y_j^2 = a_7 \theta_{j+1} + a_8 \theta_{j+2} + b_1 y_{j+1}^2 + b_2 y_{j+2}^2 \tag{5}$$

$$\Theta_j = c_2(y_j^1 + y_j^2) \tag{6}$$

Typically an additional stage is required to select a suitable threshold to get a binary image for the detected edges.

2 Implementation

2.1 CPU

A python code was developed (available here) to understand and evaluate the performance of the filter. Figure 1 shows the input image, x sweep , y sweep and the final output

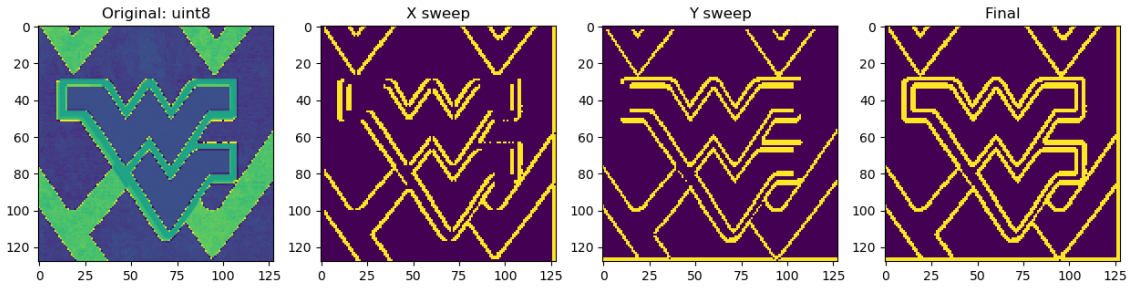

Figure 1: A sample 128×128 image processed using the Canny-Deriche algorithm with $\alpha = 5$

2.2 FPGA

The signal processing stage is shown in figure 2 and this is directly implementing equation 1 using a tapped design.

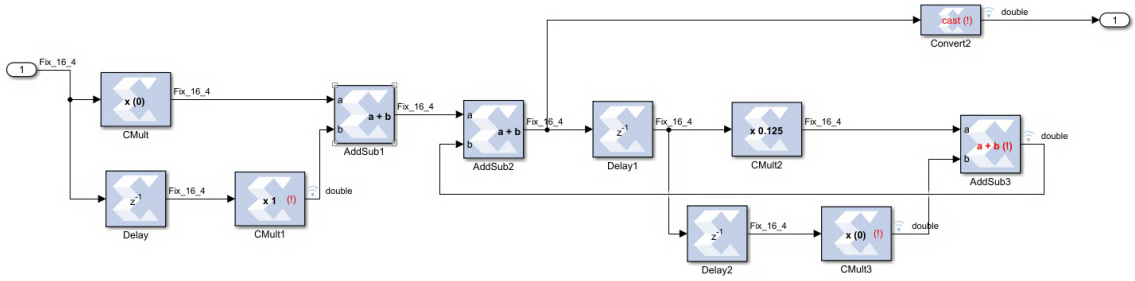

Figure 2: Internal operation of the signal processing block in Simulink

The test system was designed with limited functionality as listed below

- Input image was loaded into a RAM block in Simulink.
- The same image was sent out repeatedly via Ethernet after processing during testing.

The Most recent Simulink project available here

3 Results and discussion

Using the CPU implementation a suitable value for α was experimentally found. It was also noted the 2^{nd} stage had little impact on the actual edges detected in these small-scale images. This was intended to be a smoothing stage originally. It was decided to omit the second stage entirely from the implementation given the final results were not significantly changed and that would cut down the execution time in half.

With the data loaded into a RAM block in Simulink accessing them via Dual port RAM was selected which would facilitate a pipeline architecture to allow the best utilization of the time while the algorithm needs to wait for data to be processed to be moved on to the next stage. This operation was verified using Simulink internal probes from different stages of the pipeline which is shown in figure 3

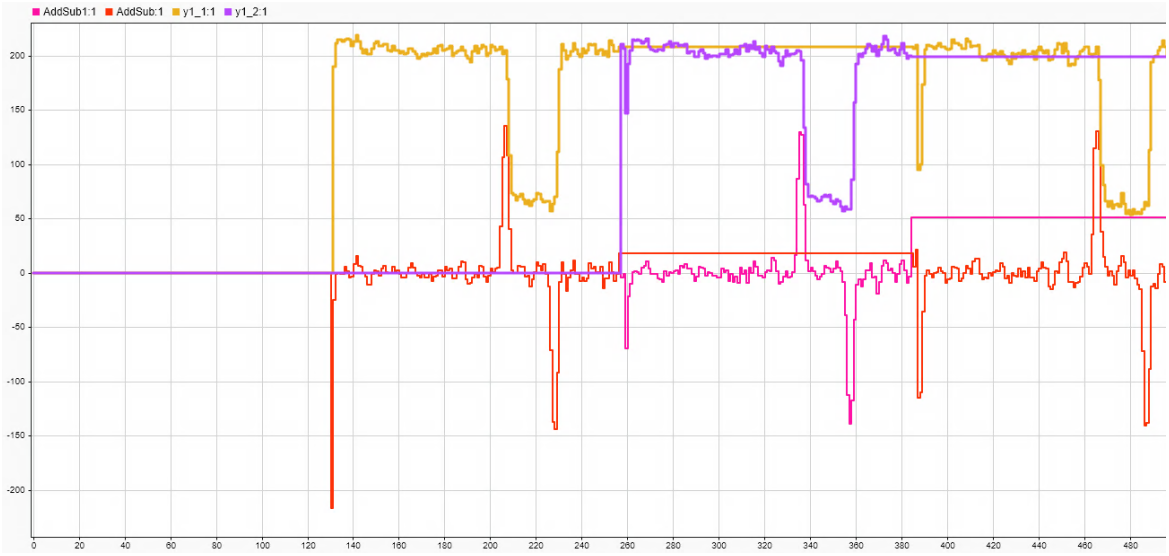

Figure 3: Internal simulated signals from Simulink.

At the time of writing the Simulink project completes the implementation stage with significant timing errors. Some investigation was done to determine the cause of this and found that the use of the signal type Double (Float 11_53, 64 bits) makes the design massive and spread out across the FPGA even though the actual operation was pretty straightforward in theory. Reducing this to a fixed point 16 bit (Fix 16_4) improved the timing discrepancy. More investigation is needed to find the cause and make appropriate changes. It should be noted that a two-time slot design was implemented to utilize the inherent delay caused due to the nature of the algorithm used. (equation 3 can only be executed only when all the data is available)

4 Conclusion

In conclusion, the Canny-Deriche edge detection algorithm is efficient in implementing on an FPGA and can be done so that you would get significant speedups. The multiplication operations could be optimized to use the *DSP48* units and this would significantly reduce the space used by the processing stage.

5 Appendix - I

Definition of the filter coefficients. α is a positive constant and can be tuned to do optimize the edge detection process.

$$k = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} \tag{7}$$

$$a_1 = 0 \tag{8}$$

$$a_2 = 1 \tag{9}$$

$$a_3 = -1 \tag{10}$$

$$a_4 = 0 \tag{11}$$

$$a_5 = k \tag{12}$$

$$a_6 = ke^{-\alpha}(\alpha - 1) \tag{13}$$

$$a_7 = ke^{-\alpha}(\alpha + 1) \tag{14}$$

$$a_8 = -ke^{2\alpha} \tag{15}$$

$$b_1 = 2e^{-\alpha} \tag{16}$$

$$b_1 = -e^{-2\alpha} \tag{17}$$

$$c_1 = -(1 - e^{-\alpha})^2 \tag{18}$$

$$c_2 = 1 \tag{19}$$