rFPGA Sobel Filter

Sobel Filter Based on SMID instructions.

Reese S. Russell

Contents

[**Objective** 2](#_Toc457786366)

[1.1 Design Goal 2](#_Toc457786367)

[1.2 Implementation Notes 2](#_Toc457786368)

**Objective**

1.1 Design Goal

The primary focus of this design is to create the optimal combination of throughput and area of a sobel filter using a Xilinx Kintex 7 Series FPGA. This can be done by looking at optimal halo size of the kernel operator to operate on and take into account parallel kernels to maximize throughput.

A halo is defined a cluster or organization of pixel that will be operated on by a single kernel in one operation. Most sobel examples will use a single pixel to be convolved at a given time. I would like to calculate the efficiency of possibly operating on a 2\*2 to N\*N area to reduce the total calls to memory requesting data.

Another element that will be looked into is the use of DSP48E1 elements provided by Xilinx to allow for SIMD add instructions as well as reducing the total path length. This in turn would lead to a higher *f* value and lower area usage is a long as the wrapping logic does not exceed the amount of LUTs and logic that would be used to create the adders and shifters necessary for the kernel to operate without a built in DSP block

1.2 Implementation Notes

The first goal of this project is the calculate the most efficient halo size or at least determine an efficiency slope. this will be based off of the number of requested bytes (*pixels*) called per op.

A typical sobel kernel for the X and Y directions of convolution are defined as in Figure 1

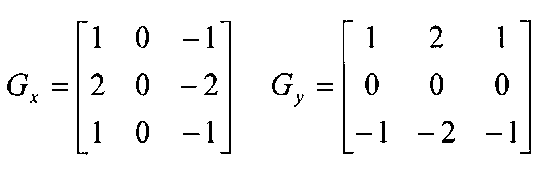


Figure 1: Sobel Kernels X and Y.

At minimum there are 6 operation of addition that must take place for either direction. Assuming this is the exact kernel used. There are only 2 inputs that are scaled. These scaled inputs are scaled by 2. This means no special multipliers are needed. Bit shifts can be used instead. In total there are 8 operations used (6 adds and 2 shifts).

Next assume that the first pixel has been completed, new data must be shifted in for the kernel to operate on. This is typically done in a left to right fashion as if reading a book. This means that the next pixel will consume 3 new items, and destroy 3 items.

How to handle these new items, first the kernel needs to know the order in which to operate on data. So where is the start point? Should all data in the convolved set be moved so that the first item be a specific coordinate or should there be an index pointer that moves based on where the shifted data went? These questions will be answered later on in this document but for now the primary objective is to determine the function of efficiency.

Firstly, how is efficiency defined? Initially we will assume additional resources are free and just measure a function of operations to completion divided by calls to memory. Table 1 will contain all the necessary variables to begin solving this equation.

|  |  |
| --- | --- |
| *Variable Name* | Description |
| Ximg | Image Width |
| Yimg | Image Height |
| Xker | Kernel Width |
| Yker | Kernel Height |
| Mret | Elements Retrieved Per Pixel |

Table 1: Variables and Definitions