Fast Sobel Edge Detection Using Parallel Pipeline-based Architecture on FPGA

Mohammad Shokrolah Shirazi, Brendan Morris
Department of Electrical & Computer Engineering, University of Nevada, Las Vegas

Abstract
Implementing image processing algorithms on FPGA has recently become more popular since it provides high speed in comparison with software-based approaches. In this paper, we have presented fast pipeline-based architecture for one of the most popular edge detection algorithms called Sobel edge detection. The objective of our work is to present two fast pipeline-based architectures for Sobel edge detection on FPGA benefiting one and two way parallelism. We used Verilog language to implement our designs and we synthesized each one for Cyclone IV FPGA. Experimental results show that our pipeline-based architectures perform edge detection process more than 379 and 751 times faster than software-based approach using MATLAB.

Real-Time Sobel Edge Detection

✓ Real time edge detection is obtained by implementing edge detection algorithm over some hardware like FPGA
✓ By passing convolution kernels over intensity image, image gradients like \( G_x \) and \( G_y \) are obtained.
\[
\begin{align*}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{align*}
\]
✓ The magnitude of the gradient is computed by (1)
\[
|G| = \sqrt{G_x^2 + G_y^2} \equiv |G_x| + |G_y|
\]
✓ Direction of gradient is computed by (2)
\[
\tan^{-1}\left(\frac{G_y}{G_x}\right)
\]

Fast FPGA-based Edge Detection Architecture

✓ Accelerating edge detection process by designing pipeline architecture
✓ Fetch
✓ Convolution
✓ Computing \( |G_x| \) and \( |G_y| \)
✓ \(|G| = |G_x| + |G_y|\)
✓ One way parallel method
✓ Two way parallel method

Experimental Results

✓ Image intensity values were written into txt file using Matlab.
✓ Two proposed architectures were implemented by Verilog language and simulated with Modelsim 10.1.d.
✓ Edge detection was done over Cameraman and Lenna images with 128 by 128 pixels.
✓ 15880 and 8005 clock cycles are needed for one way and two way parallel architectures respectively.
✓ Filtered image intensity values are reconstructed to form the images using Matlab.

✓ In one way parallel method, eight intensity values regarding to each image pixel are fetched by each clock cycle.
✓ Thanks to our pipeline, image gradients \( G_x \) and \( G_y \) are obtained in parallel at same clock cycle.

✓ Two image gradients \( G_x \) and \( G_y \) are obtained at same clock cycle.

Experimental Results

✓ Two edge detection architectures are synthesized using Quartus II 9.1 for cyclone IV FPGA.
✓ Total consumed memory bits are identical but two way parallel architecture consumes more logic elements.
✓ Our fast FPGA-based edge detection architectures are more than 379 and 751 times faster than software-based approach using MATLAB.

<table>
<thead>
<tr>
<th>Table 1. Device utilization summary</th>
<th>Total Logic Elements</th>
<th>Total Memory Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA1 (One-way Method)</td>
<td>735 (1048576 bits)</td>
<td>1%</td>
</tr>
<tr>
<td>FPGA2 (Two-way Method)</td>
<td>666 (1048576 bits)</td>
<td>1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Run-time of the Soble edge detection processes</th>
<th>FPGA1 (One-way Method)</th>
<th>FPGA2 (Two-way Method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenna</td>
<td>0.0005815</td>
<td>0.0000473</td>
</tr>
<tr>
<td>Cameraman</td>
<td>0.000732</td>
<td>0.0000073</td>
</tr>
</tbody>
</table>