International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 1, January- 2013

FPGA based Area optimized and efficient Architecture for NMS and Thresholding used for Canny Edge Detector

[1,]Chandrashekar N. S.,[2,] Dr. K. R. Nataraj
[1,]Department of ECE, Don Bosco Institute of Technology, Bangalore.
[2,] Department of ECE, SJB Institute of Technology, Bangalore.

***Abstract*—— In this paper, we present an architecture for Non Maximal Suppression used in Canny edge detection algorithm that results in significantly reduced memory requirements decreased latency and increased throughput with no loss in edge detection. The new algorithm uses a low-complexity 8-bin non-uniform gradient magnitude histogram to compute block-based hysteresis thresholds that are used by the Canny edge detector. Furthermore, an FPGA-based hardware architecture of our proposed algorithm is presented in this paper and the architecture is synthesized on the Xilinx Virtex 5 FPGA. The design development is done in VHDL and simulates the results in modelsim 6.3 using Xilinx 12.2.**

***Index Terms*— Canny Edge detector, Distributed Processing, Non-uniform quantization, FPGA**.

I. INTRODUCTION

The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries. There is certainly a great deal of diversity in the applications of edge detection, but it is felt that many applications share a common set of requirements [1]. The Canny edge detector is used in many real-world applications due to its ability to extract significant edges with good detection and good localization performance. Unfortunately, the canny edge detection algorithm contains extensive pre-processing and post-processing steps and is more computationally complex than other edge detection algorithms, such as Roberts, Prewitt and Sobel algorithms [2].

In a recursively implementable edge detection algorithm is suggested and optimized using retiming techniques. But its performance is quite poor in images with low SNRs. The approach of [3] combines the derivative and smoothening operations of the Canny algorithm into a single mask to reduce computations. The pipelined implementation is a block-based approach with a block-size of 2 rows of pixels. It overcomes the dependencies between the blocks by fixing high and low thresholds to a constant value. In both of these approaches gradient thresholds are not adapted to the image characteristics, and their performance is not guaranteed for blurred images and images with low SNRs [3]. Canny edge detector a parallel architecture [4] of simultaneous 4-pixel calculation is proposed, which increases the throughput of the design without increasing the need for on-chip cache memories. This design has been synthesized for low-end and high-end Xilinx FPGA. However, in [3], the hysteresis thresholds calculation is based on a very finely and uniformly quantized 64-bin gradient magnitude histogram, which is computationally expensive and, thereby, hinders the real-time implementation. In this paper, a method based on non-uniform and coarse quantization of the gradient magnitude histogram is proposed. In addition, the proposed algorithm is mapped onto reconfigurable hardware architecture. This architecture exploits the parallelism and pipelining of the proposed algorithm, and therefore, yields significant speedup in running times.

This paper is organized as follows. Section 2 gives a Brief analysis of the Canny edge detector

algorithm. Section 3 presents the proposed Canny edge detection algorithm which includes a novel method for the hysteresis thresholds computation based on a non-uniform quantized gradient magnitude histogram. Simulation results are presented in Section 4. A conclusion is given in Section 5.

II. CANNY ALGORITHM ANALYSIS

The block diagram of the Canny algorithm is demonstrated in Fig. 1. The Canny algorithm first smoothes the image to eliminate noise by using a smoothing filter such as the Gaussian Convolution. The Gaussian smoothing is performed by using a mask (matrix) which is sled over the image, manipulating a square of pixels at a time. The bigger the dimensions of the mask are, the lower sensitivity the detector has to noise. A mask is a common choice for the size of a Gaussian filter. The output smoothed image is denoted as I (x, y).
After smoothing the next step is the calculation of the gradient of the image. The gradient calculation leads to the detection of the possible edge strength and direction. This is executed by another convolution with a gradient operator. The most commonly used gradient operators are the Prewitt and the Sobel operators. Both operators perform a 2-D spatial gradient measurement on an image. Calculating the horizontal gradient $G_x(x, y)$ and vertical gradient $G_y(x, y)$ at each pixel location by convolving the image $I(x, y)$ with partial derivatives of a 2D Gaussian function. Computing the gradient magnitude G(x, y) and direction θ(x, y) at each pixel location. Gradient magnitude and orientation following equations:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad -- \quad (1)$$

$$\theta = \arctan (G_y/G_x) \quad -- \quad (2)$$

Arctan and the division can be eliminated by simply comparing G_x and G_y values. If they are of similar length, we will obtain a diagonal direction, if one is at least 2.5 times longer than the other, we will obtain a horizontal or vertical direction.

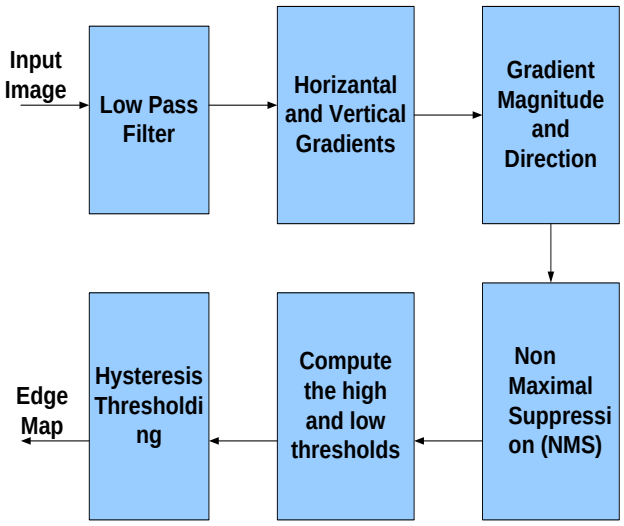

Fig.1 Block diagram of the Canny edge detection algorithm

After the edge directions are known, non-maximum suppression is applied. Non-maximum suppression is used to trace along the gradient in the edge direction and compare the value perpendicular to the gradient. Two perpendicular pixel values are compared with the value in the edge direction. If their value is lower than the pixel on the edge then they are suppressed i.e. their pixel value is changed to 0, else the higher pixel value is set as the edge and the other two suppressed with a pixel value of 0. Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also

selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1 [1].

III. IMPLEMENTATION OF THE PROPOSED CANNY EDGE DETECTOR

In this section, we describe the hardware implementation of our proposed canny edge detection algorithm on the Xilinx virtex-5 FPGA.
In the proposed architecture, it consists of the following 5 units.

- Smoothening unit using Gaussian filter.
- Vertical and horizontal gradient Magnitude calculation unit.
- Directional non-maximum suppression unit.
- High and low threshold Calculation unit.
- Thresholding with hysteresis unit.

a) Image Smoothening:

The input image is smoothened using a 3×3 Gaussian mask. The Gaussian filter is separable and, thus, the implementation of the 2-D convolution with the 3×3 Gaussian mask is achieved using row and column 1- D convolutions.

b) Gradients and Gradient Magnitude Calculation:

This stage calculates the vertical and horizontal gradients using convolution kernels. The kernels vary in size from 3 × 3 to 9×9, depending on the sharpness of the image. The whole design is pipelined, and thus the output is generated every clock cycle. This is input to the magnitude calculation unit which computes, at each pixel location, the gradient magnitude from the pixel's horizontal and vertical gradients.

c) Directional Non Maximum Suppression:

Fig. 2 shows the architecture of the directional non-maximum suppression unit. In order to access all the pixel's gradient magnitudes in the 3 × 3 window at the same time, two FIFO buffers are employed. The horizontal gradient G_x and the vertical gradient G_y control the selector which delivers the gradient magnitude (marked as M(x, y) in Fig. 2) of neighbors along the direction of the gradient, into the arithmetic unit. This arithmetic unit consists of one divider, two multipliers and one adder, which are implemented using the Xilinx Math Functions IP cores. The output of the arithmetic unit is compared with the gradient magnitude of the center pixel, and the pixel that has no local maximum gradient magnitude is eliminated.

d) Calculation of the hysteresis thresholds:

Since the low and high thresholds are calculated based on the gradient histogram, we need to compute the histogram of the image after it has undergone directional non-maximum suppression. As discussed in Section 2, an 8-step non-uniform quantizer is employed to obtain the discrete histogram for each processed block. The block-based hysteresis thresholds (high threshold ThH and low threshold ThL) are computed.

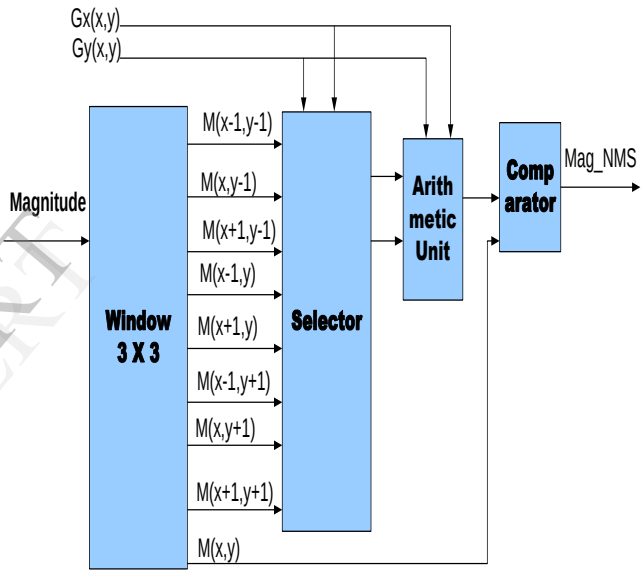

Fig.2 Directional Non Maximum Suppression Unit

e) Thresholding with hysteresis:

Since the output of the non maximum suppression unit contains some spurious edges, a method of thresholding with hysteresis is used. Two thresholds, high threshold ThH and low threshold ThL, which are obtained from the threshold calculation unit, are employed. Let f(x, y) be the image obtained from the non maximum suppression stage, f1(x, y) be the strong edge image and f2(x, y) be the weak edge image. Fig. 3 illustrates the pipelined design of this stage.

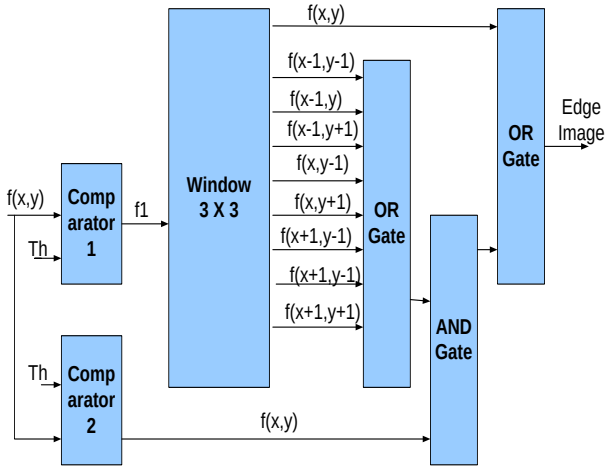

Fig.3 Pipelined architecture of the Thresholding Unit

IV. SIMULATION RESULTS AND ANALYSIS

The Non Maximum Suppression Unit and Thresholding Unit results in canny edge detection system implemented in FPGA using Device Virtex 5 XC5VTX240T and Package FF1759. Figure 4(a) shows design utilization summary of Non Maximum Suppression Unit and Figure 4 (b) and 4(c) shows RTL top module schematic and simulation results of Non Maximum Suppression Unit respectively.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	11443	149760	7%
Number of Slice LUTs	26223	149760	17%
Number of fully used LUT-FF pairs	3440	34226	10%
Number of bonded IOBs	23	680	3%
Number of BUFG/BUFGCTRLs	3	32	9%

Fig.4 (a) Design Summary of Non Maximum Suppression Unit.

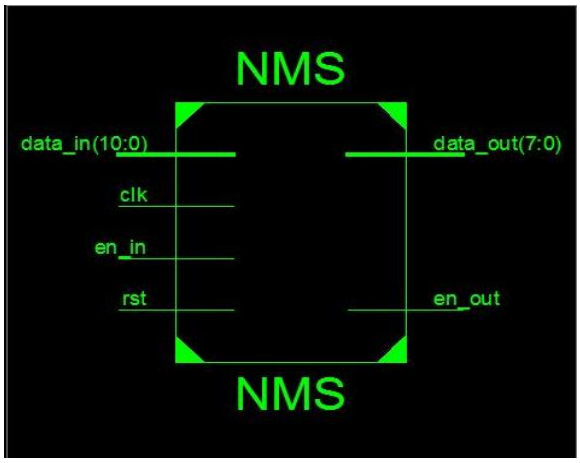

Fig.4 (b) RTL Top module Schematic of Non Maximum Suppression Unit

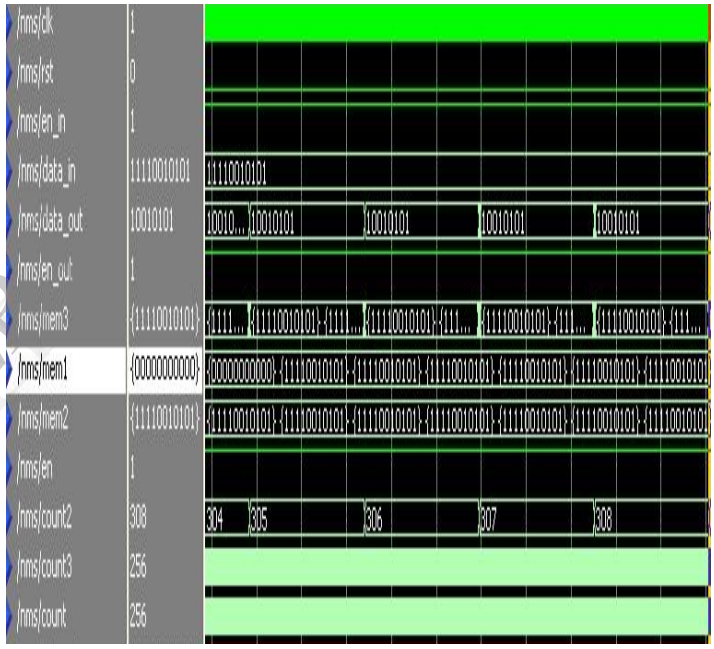

Fig.4 (c) Simulation Results of Non Maximum Suppression Unit.

Figure 5(a) shows design utilization summary of Thresholding Unit and Figure 5 (b) and 5(c) shows RTL top module schematic and simulation results of Thresholding Unit respectively.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	193	149760	0%
Number of Slice LUTs	271	149760	0%
Number of fully used LUT-FF pairs	163	301	54%
Number of bonded IOBs	30	680	4%
Number of BUFG/BUFGCTRLs	2	32	6%

Fig.5 (a) Design Summary of Thresholding Unit

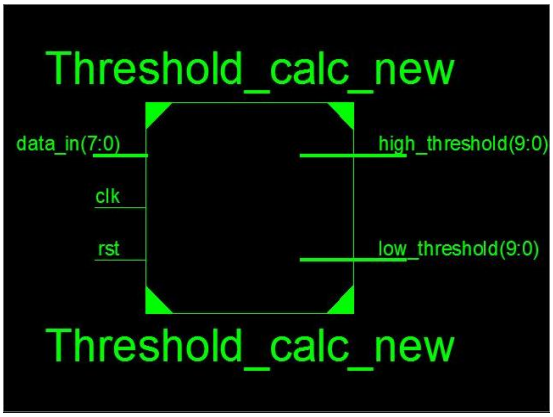

Fig.5 (b) RTL Top module Schematic of Thresholding Unit

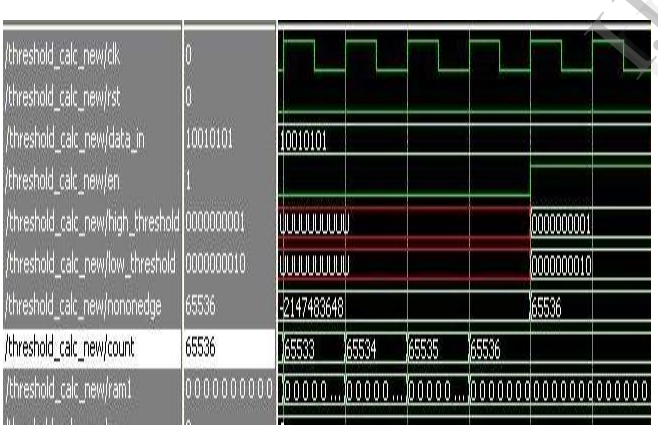

Fig.5(c) Simulation Results of Thresholding Unit

V. CONCULSION AND FUTURE WORK

We proposed a novel nonuniform quantized histogram calculation method in order to reduce the computational cost of the hysteresis threshold selection. As a result, the computational cost of the proposed algorithm is very low compared to the original Canny edge detection algorithm. The algorithm is mapped to onto a Xilinx Virtex-5 FPGA platform and tested using ModelSim. It is capable of supporting fast real-time edge detection for images and videos with various spatial and temporal resolutions including full-HD content.

REFERENCES

[1] J. Canny, "A computational approach to edge detection,"*IEEE Trans. PAMI*, vol. 8, no. 6, pp. 679 –698, Nov. 1986.

[2] *Qian Xu, Chaitali Chakrabarti and Lina J. Karam,"*A Distributed Canny Edge Detector and Its Implementation On FPGA", Arizona State University, Tempe, AZ.

[3] *Qian Xu, Chaitali Chakrabarti and Lina J. Karam,"*A Distributed Canny Edge Detector and Its Implementation On FPGA", Arizona State University, Tempe, AZ.

[4] *Christos Gentsos, Calliope-Louisa Sotiropoulou and Spiridon Nikolaid*i,"Real- Time Canny Edge Detection Parallel Implementation for FPGAs," Thessaloniki, Greece

[5] *W. He and K. Yuan*, "An improved Canny edge detector and its realization on FPGA," *WCICA*, pp. 6561 –6564,Jun. 2008.

[6] *D. V. Rao and M. Venkatesan*, "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," *ITCC*, vol. 2, pp. 843 – 847, Apr. 2004.