LUNG CANCER DETECTION USING DBRAM-CED METHOD ON FPGA VIRTEX 6

Chandrashekar N.S.* and Dr. K.R. Nataraj **
*Associate professor, Department of ECE, Don Bosco Institute of Technology, VTU, Bangalore
**Professor, Department of ECE, SJB Institute of Technology, VTU, Bangalore

ABSTRACT: In the Digital Image processing, discovering the boundaries of the objects in the image is one of the important stage for computer vision applications. There are different kinds of Edge Detection (ED) techniques available such as Canny, Laplacian, Log Operator, Sobel etc. From the analysis, It is found that Canny Edge Detector (CED) provides better results in comparison with other edge detectors. In this paper, architecture for CED has been introduced, which is named as DBRAM-CED. In order to improve the ED performance, Dual port Block Random Access Memory (DBRAM) is used. The hardware analysis of the performance parameters such as LUT, flip flop, slices and frequency of FPGA Virtex 6 are measured.

KEYWORDS: Lung Image, Dual port Block Random Access Memory (DBRAM), Canny Edge Detection (CED)

I INTRODUCTION

Edge Detection is a preprocessing step of the image processing. It is used for defining the object boundaries of the image. CED is used for achieving high accuracy and less hardware consumption in ED [1]. In CED, different sigma and threshold values are given to the various parts of the image instead of processing the whole image with single sigma and threshold value to analyze the ED [2]. The brain tumor detected from the image based on the combination of CED and Harris corner detector and the edges of the MRI brain tumor images are determined by using Sobel ED [3-4]. In CED, the dynamic weighted smoothing filter (adjusted median filter) is replaced by Gaussian filter for removing the speckle noise in ultrasound images. Circle search has been applied with CED to detect the iris of an image [5-6]. The image frames first divided into blocks, and then the CED is applied to each block for reducing the time and memory consumption [7]. CED is introduced as a parallel architecture for calculating the 4 pixel values of the image simultaneously. It is a pipelined architecture which employs on-chip BRAM memories to cache the data among the various stages [8]. Linear smoothing filter is replaced by median filter for removing the salt and pepper noise. The processing speed of the CED is improved by adopting the shifting LUT based direction calculation [9].

Offloading of CED is targeted from processing system to programmable logic CED for improving the computation time [10]. An adaptive threshold algorithm is used in CED for detecting the edges based on the block type. Latency increases when the loss of edges occurred in the high detailed regions [11]. CED and multi-step algorithm detects the edges along with the noise and it suppressed simultaneously. If the operation of smoothening gets slow, it affects the performance of the ED [12]. A 5 by 5 sliding window is introduced in the CED for conducting the image smoothing and gathers the gradient at the same time. It operates at the maximum frequency up to 132.4 MHz [13]. All these existing methods of ED using CED have some constraints like high delay and area. To solve these problems, DBRAM-CED method is introduced in this paper for reducing the hardware utilization. In Gradient & Magnitude Calculation unit (GMC), single DBRAM will be used instead of using two BRAM. Finally, FPGA Virtex 6 performances like LUTs, flip flops, slices, frequency is improved in DBRAM-CED method than existing method.

II LITERATURE REVIEW

Vasilios Kelefouras *et al.* [14] introduced the canny and Hough transform for speeding up the edge and line detection respectively. Instructions of arithmetic and amount of load/store were reduced as well as the memory size required for storing the input and output arrays were decreased by the data reuse and algorithm parallelism. Huge numbers of registers are needed to store the elements for data reuse.

Paulo Ricardo Possa *et al.* [15] presented the CED and Harris corner detection for detecting the edge and corner respectively and the architecture comprises of neighborhood extractors and threshold operators. These components were parameterized at the runtime. In order to detach the output frame from the input frame a circular buffer is placed between the Bayer filter and canny/Harris detectors. By using these types of architectures in edge and corner detection, the computational complexity, memory requirements and latency are reduced. The flexibility level increases when the reconfigurable interconnection is provided between the building blocks.

Qian Xu *et al.* [16] presented the distributed CED algorithm which computes the ED threshold depends on the block type and local distribution of the gradients in the image block. A finely quantized magnitude histogram was developed to compute the low and high level hysteresis thresholds. This distributed CED gave better ED performance and also it significantly reduced the latency. It needs more memory for processing the images in the ED process.

Fuqiang Zhou *et al.* [17] introduced the modified Line Segment Detector (LSD) on FPGA to detect the lines in the images. This modified LSD contains the Gaussian filter and CED. The Gaussian filter was used for smoothening the image then the adjusted CED was employed to achieve an edge map at single pixel width. The on-chip resources were saved by using the fewer amount of FIFOs for storing the intermediate values and also it improved the accuracy and reliability. The time consumption of the desired method depends on the size of the image.

Juseong Lee *et al.* [18] presented the energy efficient hardware architecture of the CED which has three major sub blocks such as image filter, a gradient magnitude & direction calculator and adaptive threshold value selection module. The energy savings and the area of the CED were greatly reduced. The computational complexity of the overall architecture becomes high.

III DBRAM-CED METHODOLOGY

The DBRAM-CED method uses distributed CED for detecting the lung cancer. The block diagram of the DBRAM-CED method is given in the Figure 1. Both the MAT-LAB and Verilog are used for the detection of lung cancer through the images. READ and WRITE operations are performed by using the MAT LAB as well as the ED process of the lung cancer is performed in the Verilog analysis. The performance of DBRAM-CED is analyzed by using the following parameters like LUT, slices, Flip-Flops (FF), frequency.

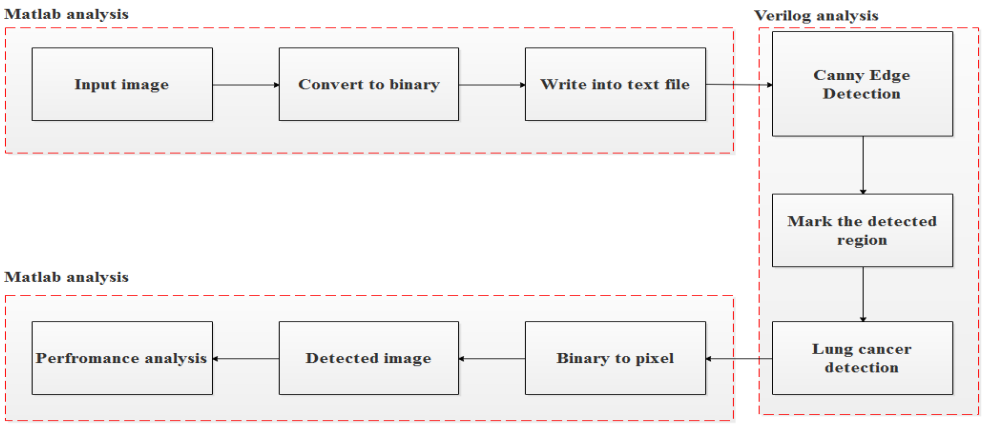

Figure 1. Block diagram for DBRAM-CED methodology

Preprocessing

Initially, the images are converted into binary form in MATLAB, because the Verilog does not accept the image file directly. The text file is created in the MATLAB and it is given as input to the Verilog analysis for detecting the edges of the lung cancer images by using distributed canny edge detector.

Distributed CED

The CED function on the entire image is to detect the edges in the Verilog analysis. The latency of the CED is directly proportional to the size of the image. The latency increase, when the size of the image becomes high. In order to overcome these constraints, the distributed CED process is used for removing the inherent dependency among the several blocks. Hence, the image is divided into blocks and each block is processed in parallel. The block diagram for the distributed CED is given in the Figure.2.

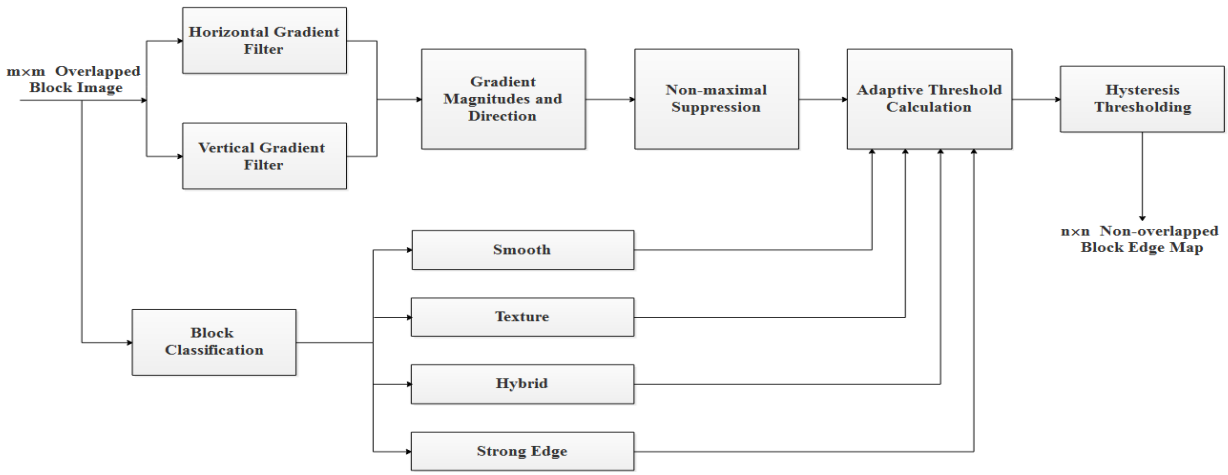

Figure 2. Block diagram for distributed CED

The distributed CED algorithm comprises the following steps.

1. Block classification unit.
2. Horizontal gradient and vertical gradient calculation at each pixel location, it is computed by convolving the image with partial derivatives of a 2D Gaussian function and direction of gradient calculation at each pixel location.
3. Non-Maximum Suppression (NMS) is applied to get thin edges.
4. The high and low level hysteresis Threshold calculation.
5. Edge map is calculated by performing hysteresis thresholding.

6. Each and every output steps are shown using MATLAB and Verilog.

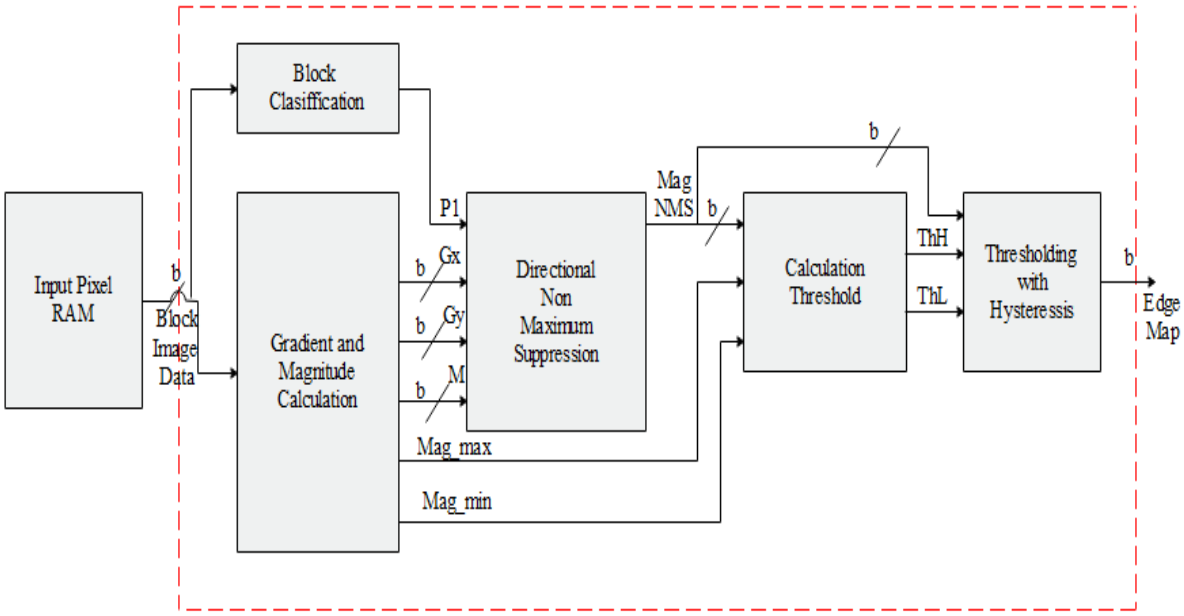

Figure 3. Architecture of of CE (computing engine for ED)

1. Block Classification Unit

Initially, the image is divided into $n * n$ non-overlapping blocks and it is classified into six types such as uniform, uniform/texture, texture, edge/texture, medium edge, and strong edge block. Based on these types, the variance of each pixel is described by using a local window of size 3×3 . This window is placed in the center around the respective pixel for labeling the edge of uniform pixel or texture types. Percentage of each pixel is computed by the variance of pixels, and it is helpful in determining the adaptive threshold calculation. The mathematical expression of the block classification unit is given in Equation 1.

$$var = \frac{1}{8}\sum_{i=1}^{9}(x_i - \acute{x})^2 \quad (1)$$

Where, is x_i the pixel intensity and $\acute{x}$ is the mean value of the 3 × 3 local neighborhood. Thus, the pixels in the 3 × 3 windows are fetched from the local memory and stored in one FIFO buffer to compute the local variance. The computation is done by using one adder, two accumulators, two multipliers and one square (right shift 3 bits to achieve multiplication by 1/8). The architecture of the block classification unit is shown in the Figure 4. Then, the local variance is compared to uniform threshold and edge threshold in order to define pixel type. The output of $COMP$ 1 and $COMP$ 2 is given to $COMP$ 4 and $COMP$ 5. Next, the output of $COMP$ 1 is given to $COMP$ 3 (here $COMP$ 1 is compared to the 0.3 × total pixel (239×202). The output of $COMP$ 3 and $COMP$ 5 is given to MUX 2 and $COMP$ 4 and $COMP$ 5 is given to MUX 2 and that outputs are employed as the control signal of MUX 1 and MUX 1 to determine the value of $P1$, which is a percentage of the total pixels in the lung cancer image that would be considered as strong edges. The $P1$ and enable signal are the outputs for the BC unit and are stored in the registers for thresholding calculation. The latency between the first input and the output P1 is $m \times m + 12.749\, ns$ clock cycles and the total execution time for the BC component is $m \times m + 12.749\, ns$.

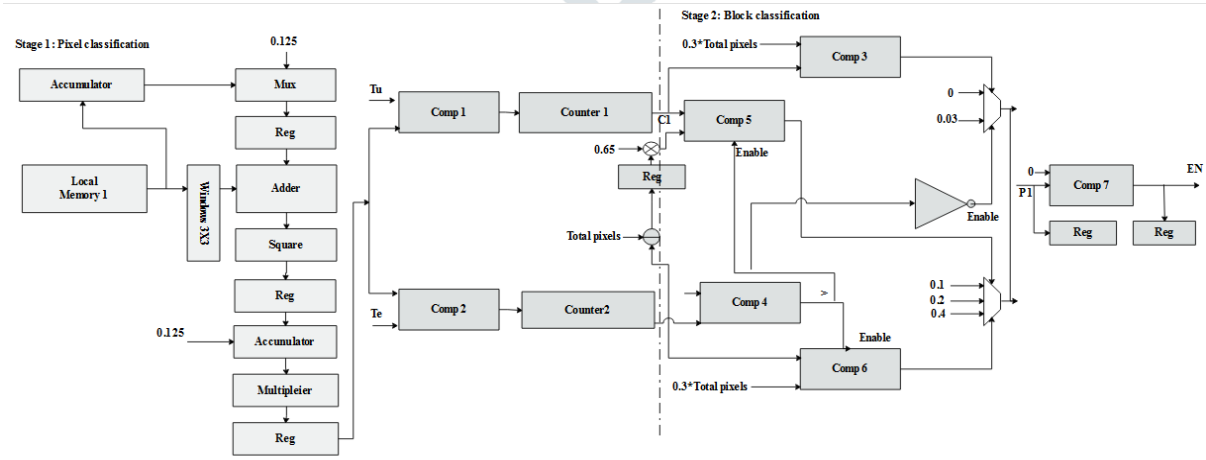

Figure 4. Architecture of the block classification unit

2. Horizontal gradient and Vertical gradient and Magnitude Calculation

The convolution kernel is used for determining the vertical and horizontal gradients and the kernel size is 3x3. Figure 5 shows the architecture for the calculation of gradient and magnitude. It consists of three computation parts such as horizontal gradient computation, vertical gradient computation, and magnitude computation, one address and time controller, which provides the address and control signals to co-ordinate the calculation. The first order derivatives are used for determining the 2D horizontal gradients and vertical gradients and it is given in the following Eqns. 2 and 3.

$$G_x = F_x(x,y) = \frac{-x}{\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} = \left(-xe^{\frac{-x^2}{\sigma^2}}\right)\left(\frac{1}{\sigma^2} e^{\frac{-y^2}{\sigma^2}}\right) \quad (2)$$

$$G_y = F_y(x,y) = \frac{-y}{\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} = \left(-ye^{\frac{-y^2}{\sigma^2}}\right)\left(\frac{1}{\sigma^2} e^{\frac{-x^2}{\sigma^2}}\right) \quad (3)$$

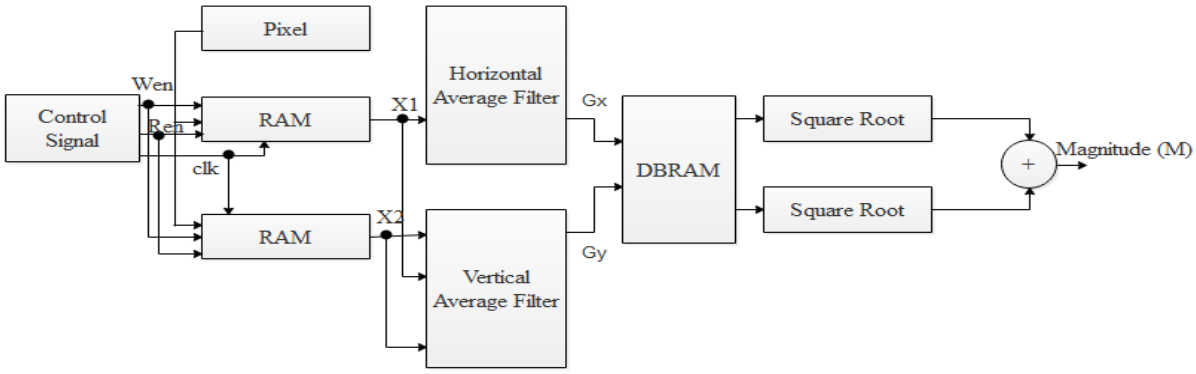

Figure 5. Architecture for Gradients and magnitude calculation

Where, G_x and G_y are the horizontal gradient and vertical gradients respectively. σ is the variance at each pixel location. x *and* y are the coordinates of each pixel. In the architecture of distributed CED, the dual port BRAM (Blocked RAM) is used instead of two BRAMs. The size of the architecture is optimized by using dual port BRAM in the magnitude and gradient calculation unit. The total execution time of the magnitude calculation is $m \times m + 8.213\ ns$. As well as horizontal and vertical execution time is $m \times m + 8.615\ ns$.

3. Directional Non maximum Suppression (NMS)

NMS receives the following inputs such as magnitude, horizontal gradients and vertical gradients. In order to access all the pixels' gradient magnitudes in the 3 × 3 window at the same time, two FIFO buffers are employed. The localization of an image is developed by reducing the edge thickness. NMS is applied over thin edges and it is used for calculating the gradient direction at each pixel. There are 8 possible directions such as 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. The gradient magnitude of this pixel is compared with two of its immediate neighbors along with the gradient direction and the gradient magnitude is set to zero if it does not correspond to a local maximum. The interpolation occurred when the gradient directions does not coincide with the eight directions (0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°) and this interpolation detects the neighboring gradients. The Architecture of directional NMS unit is shown in the Figure 6. The proposed work has used MUX instead of the selector.

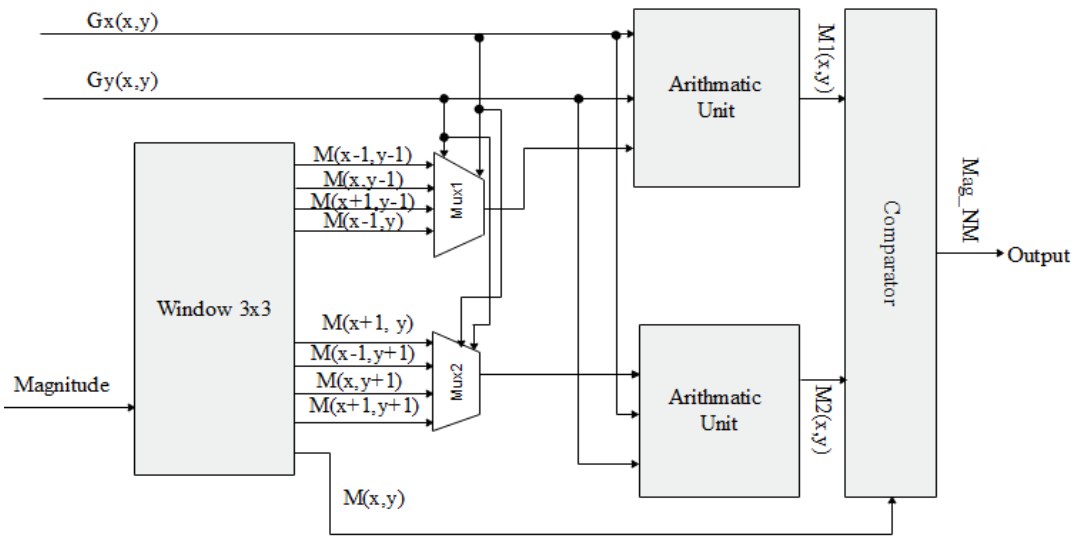

Figure 6. Directional NMS unit Architecture

4. The high and low level hysteresis Threshold calculation

The P1 value defined by the BC unit, $mag - max, and\ mag - min$, determined by gradient and magnitude computation unit, are the inputs for computation of threshold unit. Since the low threshold (LT) and High Thresholds (HT) values are calculated based on the gradient histogram, it is necessary to compute the histogram of the image after it has undergone directional non-maximum suppression. An 8-step non-uniform quantizer is employed to obtain the discrete histogram for each processed block. The block-based hysteresis thresholds (HT Th_H and LT Th_L) are computed. In this threshold calculation unit, area efficient optimal adders are used instead of the normal adder The block diagram of the arithmetic threshold calculation unit is shown in figure 7. The execution time of the threshold calculation is $m \times m \times 7.21\ ns$.

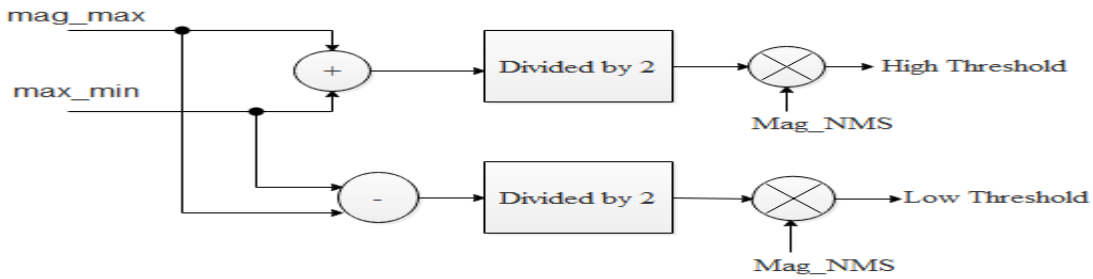

Figure **7.** Threshold calculation unit

5. Thresholding with Hysteresis

The gradient magnitude of each pixel after directional NMS is fetched from used as input to the thresholding unit. Meanwhile, the high and low threshold, which are determined by the threshold calculation unit, are also the inputs for this unit.The output of this block gives final edge map of the lung cancer cell.

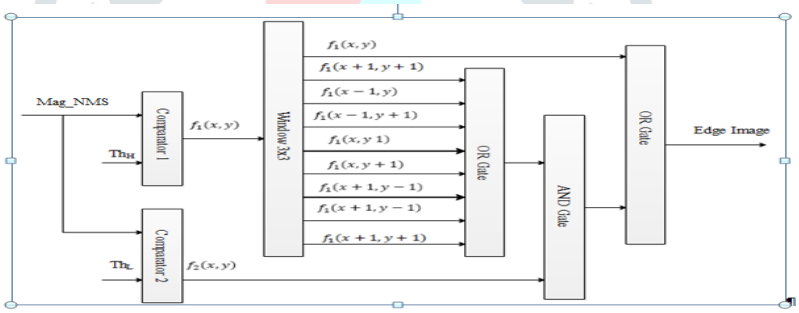

Figure 8**.** Pipeline architecture of thresholding with hysteresis

IV Results and Discussion

The DBRAM-CED design timing diagram is verified in Modelsim 10.5c using Verilog code. RTL schematic is taken from pro tool. FPGA performance is analyzed for the device Virtex-6 by using Xilinx ISE tool.

Initially The DBRAM -CED method read the Lung CT scanned image by using Matlab, which is converted into Black and white (Binary) format which is show in the figure 9. The Binary image is converted into the text file (for example shown in the figure 10) for Verilog Analysis.

V FPGA Synthesis

This FPGA synthesis is implemented in Xilinx tool for the device Virtex-6. From this tool, the performance will be calculated such as LUT, flip flop, Slices, and Frequency.

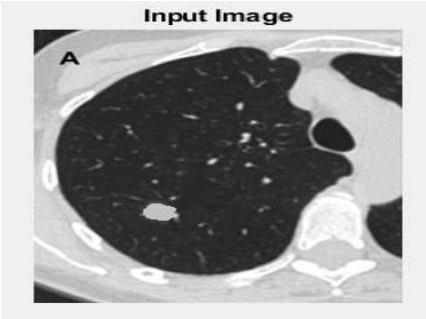

Figure 9. Input image

Figure 10. Image binary value

Table 1. Performance comparison of existing and DBRAM-CED method

Target FPGA	Circuit	LUT	Flip flop	Slice	Frequency (MHz)
Virtex6 xc6vcx75t	Existing [16]	122/46560	87/93120	57/11640	227.749
	DBRAM- CED	69/46560	36/93120	44/11640	287.017

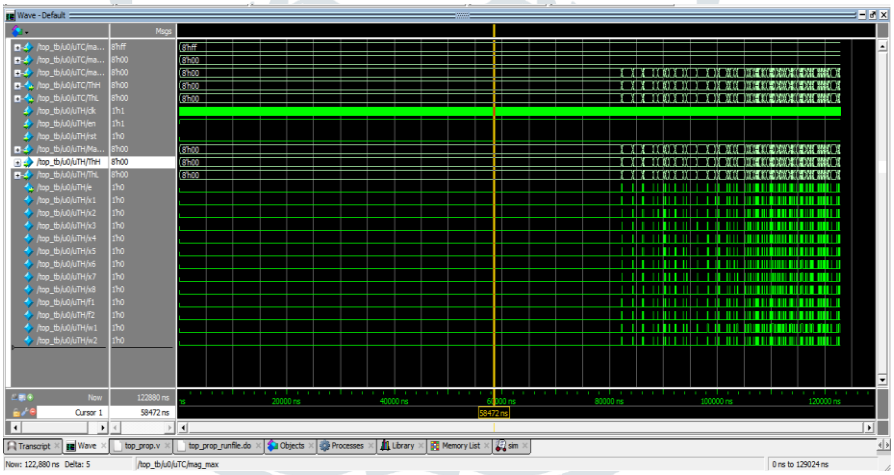

Figure 11. output waveform of the DBRAM -CED architecture

The Verilog lung cancer detection consist of four stages such as Gradient in X direction stage, Gradient in Y direction stage, Magnitude of Gradient stage and Mass NMS stage which is shown in the figure 12. The output images cannot be shown in Verilog thus a text file is created for displaying the output in Matlab and the performance are measured.

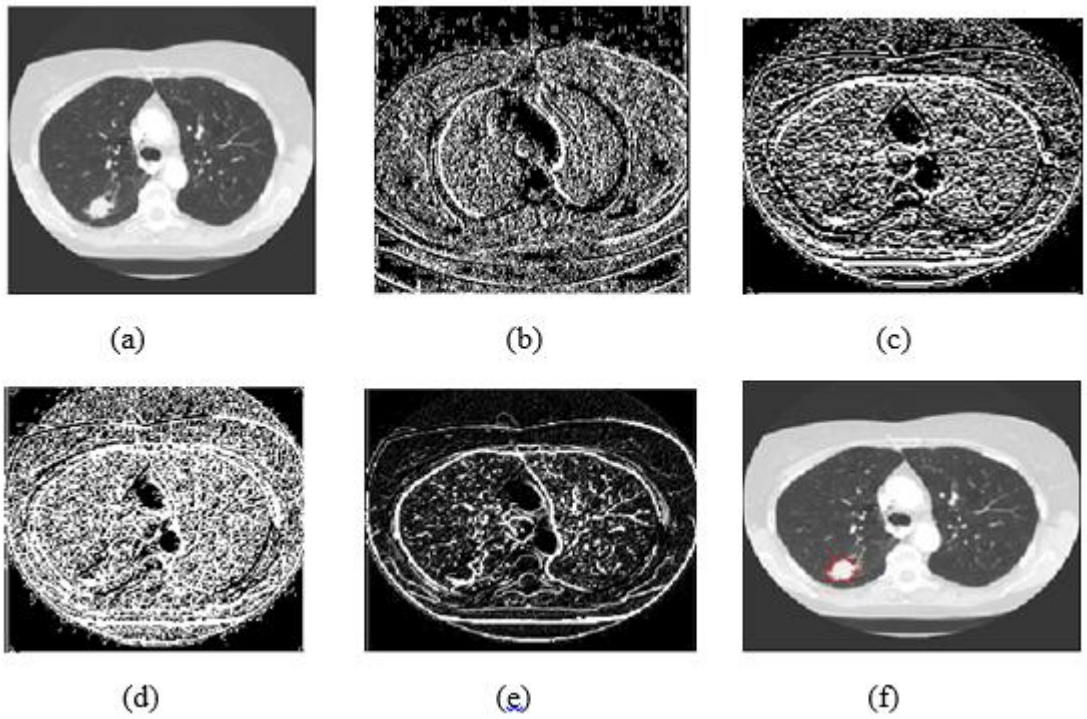

Figure 12. lung cancer image (a) Input Image (b) Gradient in X direction (c) Gradient in Y direction (d) Magnitude of gradient, (e) NMS and (f) Output image

VI Conclusion

In this paper hardware architecture for CED has been introduced which is named as DBRAM-CED. To improve the ED performance, Dual port Block Random Access Memory (DBRAM) is used. By using this DBRAM-CED technique provide better performance. To analyze the FPGA performance, parameters like LUT, flip flop, slices, frequency are measured.

References

[1]. Sangeetha, D., and P. Deepa. "An Efficient Hardware Implementation of Canny Edge Detection Algorithm." 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference on. IEEE, 2016, 457-462.

[2]. Shokhan, M. H. "An efficient approach for improving canny edge detection algorithm." International Journal of Advances in Engineering & Technology 7.1, 2014, 59.

[3]. Ramya, G., S. Bhuvaneshwari, and Faculty Student. "FPGA Based Implementation of Edge and Corner Detection in MRI Brain Tumor Image." International Journal of Engineering Science and Computing, 2015, 1046-1051.

[4]. Nikolic, Marina, Eva Tuba, and Milan Tuba. "Edge detection in medical ultrasound images using adjusted Canny edge detection algorithm." Telecommunications Forum (TELFOR), 24th. IEEE, 2016.

[5]. Ngo, H., Shafer, J., Ives, R., Rakvic, R. and Broussard, R., 2012, July. Real time iris segmentation on FPGA. In Application-Specific Systems, Architectures and Processors (ASAP), 23rd International Conference on IEEE, pp. 1-7.

[6]. Othman, Z., Haron, H., Kadir, M.R.A. and Rafiq, M., 2009. Comparison of canny and Sobel edge detection in mri images. Computer Science, Biomechanics & Tissue Engineering Group, and Information System, 133-136.

[7]. KABADE, AMRUTA L. "Canny edge detection algorithm." International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) 5.5, 2016, 1292-1295.

[8]. Gentsos, C., Sotiropoulou, C.L., Nikolaidis, S. and Vassiliadis, N., 2010, December. Real-time canny edge detection parallel implementation for FPGAs. In Electronics, Circuits, and Systems (ICECS), 17th IEEE International Conference on IEEE, 2010, 499-502.

[9]. Li, Xiaoyang, Jie Jiang, and Qiaoyun Fan. "An improved real-time hardware architecture for Canny edge detection based on FPGA", Intelligent Control and Information Processing (ICICIP), Third International Conference on IEEE, 2012, 445-449.

[10]. Abdelgawad, Hanaa M., Mona Safar, and Ayman M. Wahba. "High level synthesis of canny edge detection algorithm on Zynq platform." Int. J. Comput. Electr. Autom. Control Inf. Eng, 9.1, 2015, 148-152.

[11]. NEETHU .P R,"CANCER CELL DETECTION USING DISTRIBUTED CANNY EDGE DETECTOR" International Research Journal of Engineering and Technology (IRJET), Volume: 02 Issue: 03, 2015, 1224-1226.

[12]. Sana'a khudayer Jadwa "Canny Edge Detection Method for Medical Image Retrieval" International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-8, August 2016, 54-59.

[13]. Jang, Yunseok, Junwon Mun, and Jaeseok Kim. "Resource-efficient FPGA architecture of canny edge detector." SoC Design Conference (ISOCC), International, IEEE, 2016, 299-300.

[14]. Kelefouras, Vasilios, Angeliki Kritikakou, and Costas Goutis. "A methodology for speeding up edge and line detection algorithms focusing on memory architecture utilization." The Journal of Supercomputing, 68.1, 2014, 459-487

[15]. Possa, P.R., Mahmoudi, S.A., Harb, N., Valderrama, C. and Manneback, P., "A multi-resolution fpga-based architecture for real-time edge and corner detection", IEEE Transactions on Computers, 63(10), 2014, 2376-2388.

[16]. Xu, Q., Varadarajan, S., Chakrabarti, C. and Karam, L.J., "A distributed canny edge detector: algorithm and FPGA implementation. IEEE Transactions on Image Processing", 23(7), 2014. 2944-2960.

[17]. Zhou, Fuqiang, Yu Cao, and Xinming Wang. "Fast and Resource-efficient Hardware Implementation of Modified Line Segment Detector." IEEE Transactions on Circuits and Systems for Video Technology, 2017, 1-11.

[18]. Lee, Juseong, Hoyoung Tang, and Jongsun Park. "Energy Efficient Canny Edge Detector for Advanced Mobile Vision Applications." IEEE Transactions on Circuits and Systems for Video Technology, Volume 28, Issue 4, 2018, 1037-1046.