# FPGA IMPLEMENTATION FOR REAL TIME SOBEL EDGE DETECTOR BLOCK USING 3-LINE BUFFERS

**[1]RONNIE O. SERFA JUAN, [2]CHAN SU PARK, [3]HI SEOK KIM, [4]HYEONG WOO CHA**

[1,2,3,4]CheongJu University
E-maul: [1]engr_serfs@yahoo.com, [2]dachang57@nate.com, [3]khs8391@cju.ac.kr, [4]hwcha@cju.ac.kr

**Abstract**— In this Paper, an efficient method of FPGA based design and implementation of Sobel Edge detector block using 3-Line buffers is presented. The FPGA provides the proper and sufficient hardware for image processing algorithms with flexibility to support Sobel edge detection algorithm. A pipe-lined method is used to implement the edge detector. The proposed Sobel edge detection operator is a model using Finite State Machine which executes a matrix mask operation to determine the level of edge intensity through different pixels on an image. This approach is useful to improve the system performance by taking advantage of efficient look up tables, flip-flop resources on target device. The proposed Sobel detector using 3-line buffers is synthesized with Xilinx ISE 14.2 and implemented on Virtex II xc2vp-30-7-FF896 FPGA device. The proposed edge detector shows good performance of edge detection with the following results; resource of utilization, obtained a 238.687 MHz on maximum clock frequency, and using MATLAB software; it shows better PSNR results in terms of 3-Line buffers utilization.

**Index Terms**—FPGA, Sobel Edge Detector, Sobel Mask, 3-Line Buffer

## I. INTRODUCTION

In the field of Image Processing, camera of stereoscopic image method can be used for object recognition. Also it has been used in different fields of applications such as high definition television.

Image processing systems requires a high performance digital processing and it should be easier to implement. Xilinx Field Programmable Gate Array (FPGA) implementation is a platform that provides to meet these needed requirements. This tool is a direct and efficient method for processing from a PC-based model simulation to a real-time FPGA based hardware implementation. Furthermore, it should support flexibility for Sobel edge detection algorithm and it has a good characteristics and satisfactory performance even on images with noise corruption. In image processing it needs to create suitable images for easy human viewing and image identification. Along with it, an image should be automatically recognize and easy to understand by a computer.

Edge of an image is the key feature of an image and it contains significant information. Edge detection is the foundation tool used in most image processing applications in obtaining this significant information. Edge detection's objectives are; to produce an outline feature of a scene from an image and to extract important features like corners and curves that can be used for high level object recognition algorithms.

The Sobel operator is widely used for edge detections of image and in video processing. Sobel mask is the most well-known method for extracting the outline. The method is to sharpen the contour extraction, reduce interference caused by noises suitable to simplify the correction image. The most commonly used edge detection methods are Roberts Detection, Prewitt Detection, Sobel Detection, and Canny Detection [1]-[4] and software implementation of these methods are available. Edges characterize boundaries and are therefore a problem of fundamental importance in image processing [5]. In this work, Sobel Edge detection algorithm is implemented in hardware as a part of line-scan edge detector for real-time application. Line-scan edge detector consists of different modules but for Sobel Analysis certain specific requirements the system must meet [6].

The rest of this paper is organized as follows. Section II described the discussion of theory of the Sobel Edge Detection Techniques. In Section III, Description of the designed hardware method that utilizes the 3-Line Buffer Method. Section IV, Comparison of Peak Signal to Noise Ratio (PSNR), synthesized results and operating method of Rajesh method [7] to the proposed method 3-Line Buffer. Section V, Discussed the conclusions of this paper.

## II. SOBEL EDGE DETECTION

The Sobel operator is a classic first order edge detection operator computing an approximation of the gradient of the image intensity function [8]. At each point in the image the result of the Sobel operator is the corresponding norm of this gradient vector. The Standard Sobel operators, for a 3×3 neighborhood, each simple central gradient estimate is vector sum of a pair of orthogonal vectors [5]. Each orthogonal vector is a directional derivative estimate multiplied by a unit vector specifying the derivative's direction. The vector sum of these simple gradient estimates amounts to a vector sum of the 8 directional derivative vectors. The point on Cartesian grid and its eight neighboring density values as shown in Fig 1.

| | | |
|---|---|---|
| a | b | c |
| d | e | f |
| g | h | i |

**Fig. 1. "a" point and its neighboring values**

The Sobel operator only considers the two orientations which are 0° and 90° convolution kernels as shown in Fig. 2. The Sobel operator has the advantage of simplicity in calculation but the accuracy is relatively low as its disadvantage. Because it only used two convolution kernels to detect the edge of image [9].

$$M_X = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad M_Y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 2 & 1 \end{bmatrix}$$

(a) Horizontal operator (b) Vertical operator

**Fig. 2. Sobel Mask Pattern**

Equation (1) shows convolution of input image with horizontal mask and Equation (2) shows convolution of image with vertical mask.

$$G_X = \{f(x+1, y-1) + 2f(x+1, y) + f(x+1, y+1)\} - \{f(x-1, y-1) + 2f(x-1, y) + f(x-1, y-1)\} \tag{1}$$

$$G_Y = \{f(x-1, y-1) + 2f(x, y-1) + f(x+1, y-1)\} - \{f(x-1, y+1) + 2f(x, y+1) + f(x+1, y+1)\} \tag{2}$$

Size of gradient can be calculated from gradient vectors as shown in Equation (3).

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{3}$$

Gradient can be written as shown below in Equation (4)

$$G(x, y) = |G_X| + |G_Y| \tag{4}$$

If Sobel operator is used to detect the edge of image f, the horizontal template $M_X$ and vertical $M_Y$ shown in Fig. 2 (a) and (b) must be used to get convolution with image, without taking into account the border conditions. It may get the same size of two gradient matrixes $G_X$ and $G_Y$ as the original image. Then the total gradient value G can be obtained by adding the two gradient matrices. Finally, the edge can get by applying threshold.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \tan^{-1}(G_X / G_Y) \tag{5}$$

In [5] a convolution mask used is usually much smaller than the actual image. As a result, the mask is slid over an area of the input image, changes that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row. It then starts at the beginning of the next row. However it did not consider real time application of large window size. Hence the proposed pseudo-codes for Sobel Edge Detection using 3-line buffers as follows:

```
Input: Sample Image
Step 1: Accept the input image.
Step 2: Apply the input image.
Step 3: Apply Sobel Edge Detection Algorithm and the gradient.
Step 4: Masks manipulation of $G_X$, $G_Y$ separately on the input image using 3-line buffer.
Step 5: Results combined to find the absolute magnitude of the gradient.
Step 6: The absolute magnitude is the output edges.
Output: Detected edges.
```

## III. THREE-LINE BUFFER METHOD

1. Image Data Control

Image control denoted as Finite State Machine (FSM) controller was designed to provide a Sobel arithmetic operation of the address signal (Write_Addr) and to provide a signal from the arithmetic operation to Sobel mask filter in Fig.3. It also provides Write_enable input for Sobel Start Point Check block.

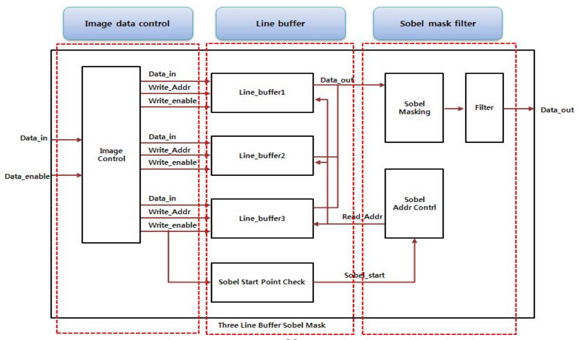


**Fig. 3. The 3-Line Buffer Sobel Mask Block**

2. Line Buffer

From the sequential output of image controller to the three pixel line that performs a function as a temporary storage. Once the activation of the address signal is being fed from the image controller, each picture store three pixel line from the image controller, as shown in the state diagram in Fig. 5 and temporarily stores it to the three pixel lines from first pixel line to the second pixel line to the third pixel line buffer.

Each individual line buffer is measured from an 8 bit (Gray scale) by a depth of M in M by N size of the image. The purpose of Sobel Start Point Check block is send the data to Sobel Address Control Block that sends read address to each buffer line.

## 3. Sobel Mask Filter

Sobel mask filter has a function of receiving the image frame from a single pixel line in clockwise order direction from the pixel line performed that by Sobel operation. This feature extracts the Sobel edge data.
The existing hardware design of Rajesh algorithm and the Line buffer when storing the entire image, it is possible read that memory that contains one image in Sobel mask, from the initial to up to the last scanned image frame pattern but without changing the mask pattern as shown in Fig. 4.
However, saving the entire image will increase the clock consumption and the size of the memory, Because of this occurrence designing additional correction block will constraints the memory size of the FPGA.

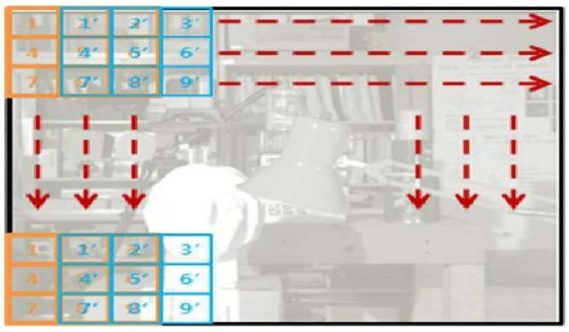


**Fig. 4. Scanning process of Sobel mask**

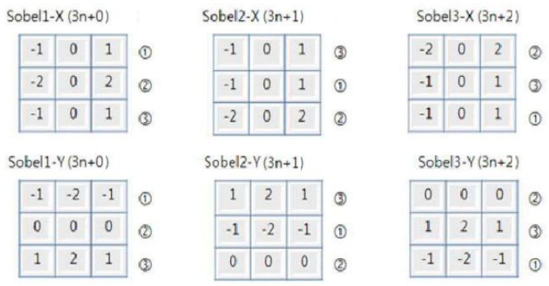


**Fig. 5. Three-Line Buffer Sobel Mask Pattern**

In order for the real time Sobel edge extraction to perform on a high speed mode, reducing the amount of memory of the designed 3-line Image buffer is necessary. When reading the gray level image data of the 3-line, an order from 1 to 2 to 3 then repeats the reading from line 1 to the next step. Sobel mask pattern should be changed to suit the three types of sequence because it is stored in one line buffer as shown in Fig. 7.
In Fig. 5 and Fig. 7 shows the information of image when changing the vertical mask and horizontal mask pattern to be changed from 3n+0, 3n+1, to 3n+2 Sobel mask will be changed after every run by line. Based on Line buffer the image information stored in Sobel mask filter performs synchronized image information. Therefore, reading the information of the Line buffer and writing separately is possible at the same time; Because Sobel edge extraction operates in real time synchronized mode.

## 4. 3-line Buffer Controller

Fig. 6 shows the state diagram of the FSM of the 3-Line buffer. Saving the individual image information of the line to states B, C, D then at status D, it will start to switch from state B, to D state again and it will continue to the address status in E state, if more than three addresses will enter this state, it will be filtered by Sobel mask. The operating input orders of the Line buffer is from B, C, to D while at states G, H and I its output are executed simultaneously. Then it will be fed to state J. The operation of each state is shown in Table I.

Table I. Lists of States

| No. | States | Remarks |
|---|---|---|
| 1 | A | Idle |
| 2 | B | Write A_Line Address & Data |
| 3 | C | Write B_Line Address & Data |
| 4 | D | Write C_Line Address & Data |
| 5 | E | If (C_Line Address> 3) Masking Start |
| | | Else if (C_Line_full) B State Check |
| 6 | F | Masking mode check, G, H, I Sobel calculation |
| 7 | G | Sobel Pattern calculation of 3N + 0 |
| 8 | H | Sobel Pattern calculation of 3N + 1 |
| 9 | I | Sobel Pattern calculation of 3N + 2 |
| 10 | J | Sobel X, Y calculation |
| 11 | K | Sobel Data absolute |

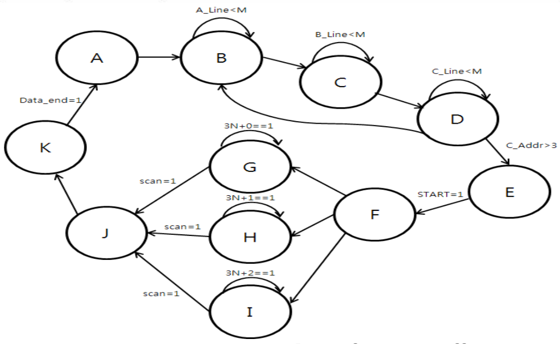


**Fig. 6. Finite State Machine of 3-Line Buffer**

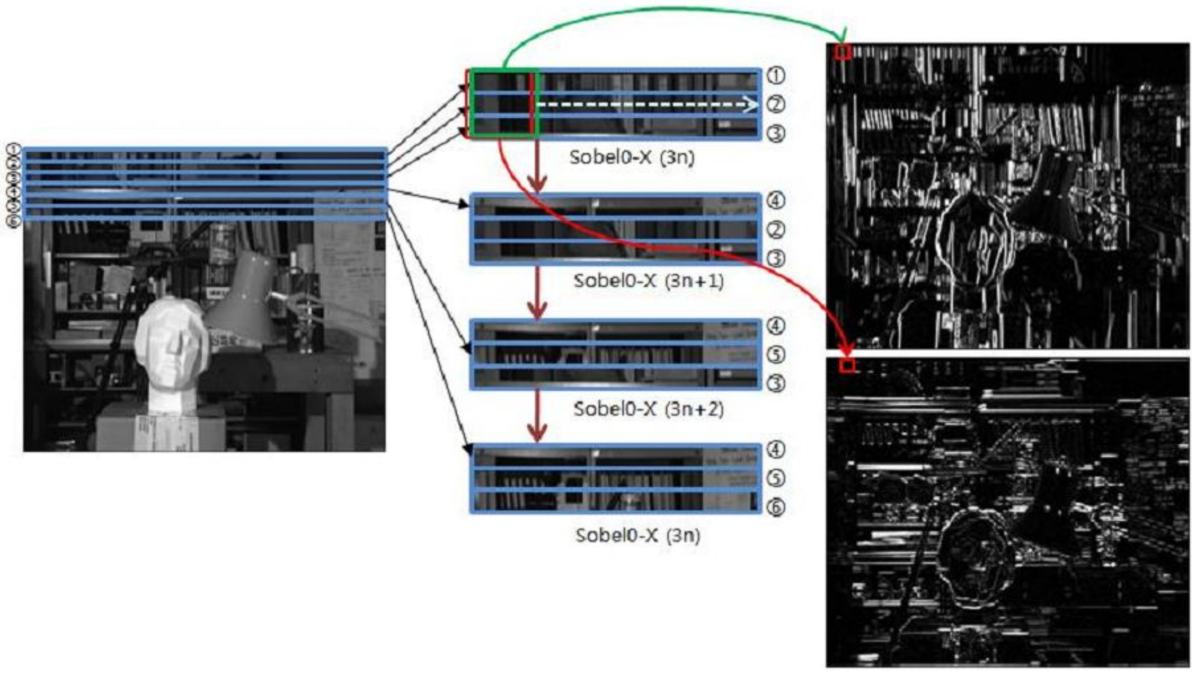


**Fig. 7. Data input sequence for 3-Line Buffer**

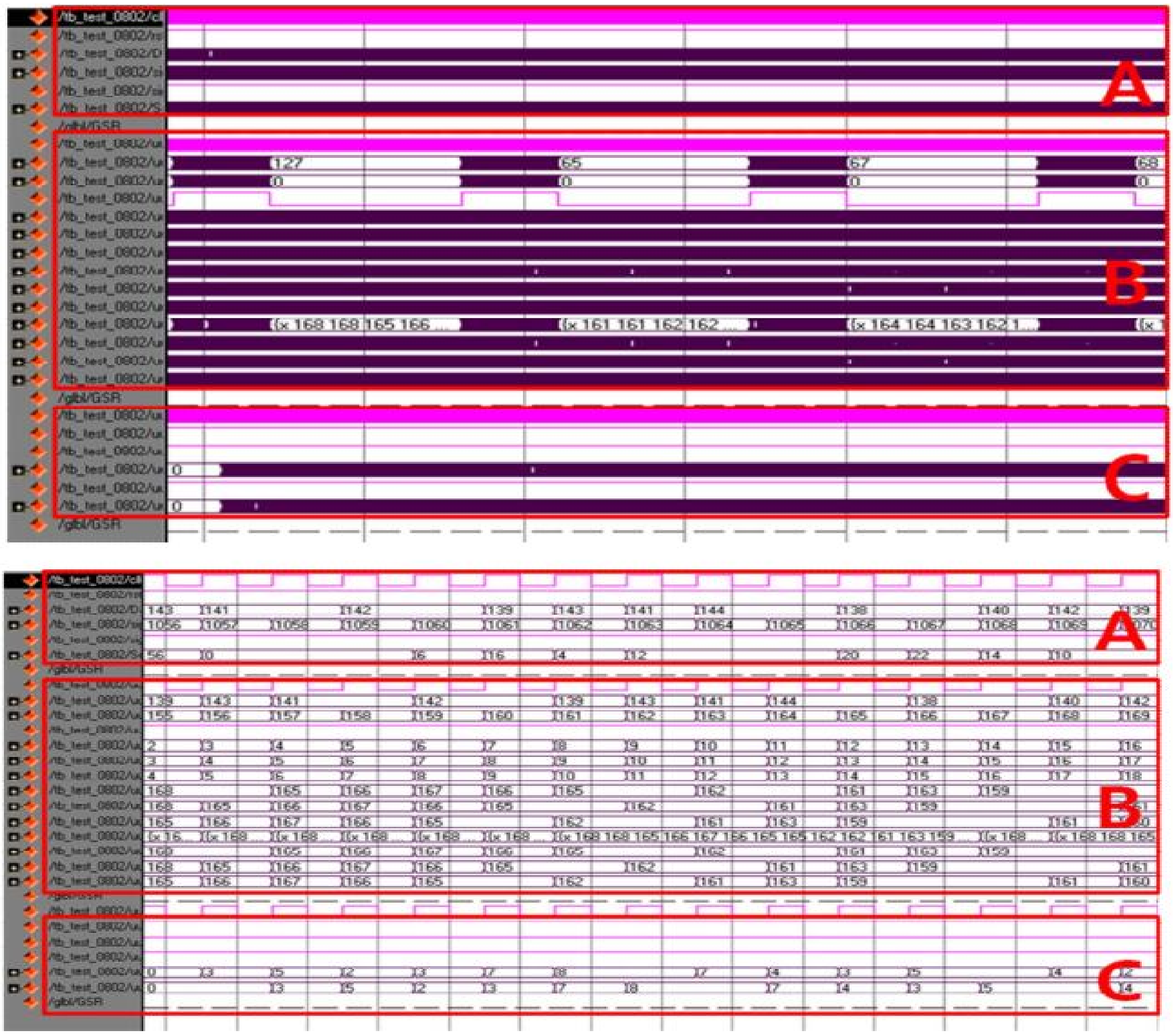


**Fig. 8. Simulation results of Three-Line Buffer**

## IV. EXPERIMENTATION

In this paper, the operating speed of the hardware region and the combine synthesis results of the proposed real-time Sobel edge extraction are verified using Virtex II xc2vp-30-7-FF896 FPGA-based Xilinx 14.2.
In Addition, in order to verify the performance of the loss and the difference of the image edge information extracted in the Sobel edge image, the PSNR results are compared between Rajesh method and the 3-line buffer method.

The proposed number of slice Flip Flop and the proposed total number of LUTs to the conventional method were reduced to 34% and 80% respectively. It can be verified from Table II that some parts of the Designed memory architecture of the proposed method is much lower than the proposed Rajesh method.

Table III shows the speed performance synthesis results of Rajesh and proposed method hardware. The minimum period of Rajesh and proposed method are 6.751ns and 4.190ns respectively. It is verified that the minimum period is reduced to 62%. Data processing delay was reduced because the whole operation period of interval differences were increased.

Fig. 8 shows the simulated results of the proposed 3-line buffer. This figure shows the overall flow simulation; in part "A", the image pixel value is fed and each pixel address is extracted to the Line buffer. Image pixel's value is stored to 3-line buffer by using FSM image controller and in part "B", image pixel's data is fed for Sobel operation.

Finally, part "C" shows the overall data are performed by Sobel operation for real time application. It can be confirmed that in part A, B and C; all of its inputs and outputs are all CLK synchronized.
In Table IV, MATLAB is utilized to the extracted edge of the reference image and the individual PNSR results of the full image of Rajesh method and the proposed 3-line Buffer method are compared.

In case of Lena image, a result shows an increase of 16.5% compare with the other method and shows an average of 6.7% using the proposed method. The extracted image of the proposed 3-line buffer method compared to Rajesh method shows similar performance.

The edge extraction results between MATLAB Software, Rajesh method and the proposed 3-Line Buffer method in accordance with the reference image is shown in Table IV.

Table II. Synthesis result of the Rajesh method and the proposed 3-Line Buffer method

| Resources | | Used | Available | Utilization |
|---|---|---|---|---|
| Number of Slice Flip Flops | Rajesh | 482 | 27392 | 1% |
| | Proposed3-Line Buffer | 167 | | 1% |
| Number of occupied Slices | Rajesh | 371 | 13696 | 2% |
| | Proposed3-Line Buffer | 369 | | 2% |
| Total Number of 4 input LUTs | Rajesh | 353 | 27392 | 1% |
| | Proposed3-Line Buffer | 283 | | 1% |
| Number of bonded IOBs | Rajesh | 14 | 556 | 2% |
| | Proposed3-Line Buffer | 20 | | 3% |
| Number of BUFGMUXs | Rajesh | 1 | 16 | 6% |
| | Proposed3-Line Buffer | 1 | | 6% |

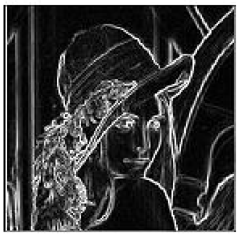
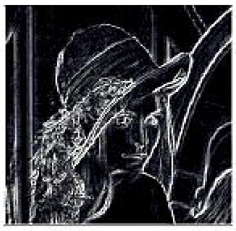
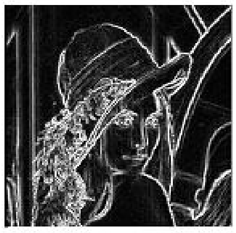
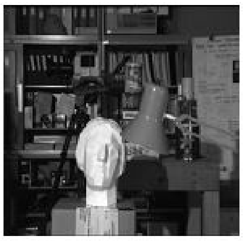
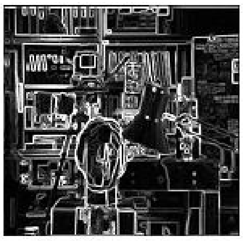
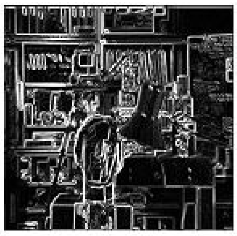
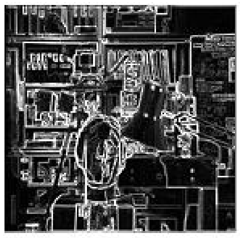
Table III. Speed Performance of the Rajesh method and the proposed 3-Line Buffer method

| Parameter | | Value |
|---|---|---|
| Minimum Period | Rajesh | 6.751 ns |
| | Proposed | 4.190 ns |
| Maximum Frequency | Rajesh | 148.133 MHz |
| | Proposed | 238.687 MHz |

Table IV. MATLAB utilized PSNR results of the Rajesh method and the proposed 3-Line Buffer method Table

| Image | Size | Rajesh [dB] | 3-Line Buffer [dB] |
|---|---|---|---|
| Baby | 640 X 480 | 30.38 | 32.99 |
| Doll | 640 X 480 | 29.04 | 29.43 |
| Lena | 640 X 480 | 28.31 | 33.00 |
| Tsukuba | 640 X 480 | 31.86 | 31.98 |

V. Comparison results of MATLAB software Rajesh method and the 3-Line Buffer method

| Image | Reference | MATLAB (Software) | Rajesh Method (Hardware) | Proposed 3-Line Buffer Method (Hardware) |
|---|---|---|---|---|
| Lena | |  |  |  |
| Tsukuba |  |  |  |  |

## CONCLUSION

In this paper, hardware correction method was used for processing a real-time image based on the proposed 3-Line buffer. Each pixel line is performed by the designated Sobel mask pattern operation. The designed image controller is customized to save the sequential input pixel in each of these line buffers.
Also, all blocks performed in real-time CLK synchronization. FSM state of Sobel mask sequential logic algorithm is applied in order to prevent the generation of clock delay phenomenon. The extracted Sobel edge information from the image experimented results were synthesized then the computing speed as well as PSNR results are compared and obtained an excellent results.
For future research, we intend to make an actual high-speed edge extraction that utilized rotation and an algorithm for rotating image correction system.

## REFERENCES

[1]. N. Senthilkumaran and R. Rajesh, "Edge Detection Techniques for Image Segmentation – A Survey of Soft Computing Approaches," Int. J. Recent Trends Eng., vol. 1, no. 2, pp. 250-254, 2009.

[2]. JM. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of Edge Detectors: A Methodology and Initial Study," Comput. Vis. Image Understanding, vol. 69, pp. 38-54, 1998.

[3]. R. Maini and H. Aggarwal, "Study and Comparison of Various Image Edge Detection Techniques," Int. J. Image Process., vol. 3, no. 1, pp. 1-12, 2009.

[4]. M. Sharifi, M. Fathy, and M. T. Mahmoudi, "A classified and comparative study of edge detection algorithms," in Proceedings of International Conference on Information Technology: Coding and Computing, pp. 117-120, 2002.

[5]. S. Gupta, and S. G. Mazumdar, "Sobel Edge Detection Algorithm," International Journal of Computer Science and Management Research., vol. 2, issue no. 5, February 2013.

[6]. S. Kabir, and ASM A. Alam, "Hardware Design and Simulation of Sobel Edge Detection Algorithm," I.J. Image, Graphics and Signal Processing., no. 5, pp. 10-18, April 2004.

[7]. Rajesh Mehra and Rupinder Verma, "Area Efficient FPGA Implementation of Sobel Edge Detector for Image Processing Applications," International Journal of Computer Applications., vol 56, no.16, October 2012

[8]. Dr. A. M. Khidhir, and N. Y. Abdullah, "FPGA Based Edge Detection Using Modified Sobel Filter," International Journal for Research and Development in Engineering., vol. 2, Issue. 1, pp. 22-32, June-July 2013.

[9]. O. R. Vincent, and O. Folorunso, "A Descriptive Algorithm for Sobel Image Edge Detection," Proceedings of Infoming Science & IT Education Conference., pp. 97-107, 2009.

★★★