International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 07, July-2015

Parallel Block-Based Architecture for Improved Edge Detection in Verilog

S. Neethu Raj
Embedded Systems
Sree Buddha College of Engineering,
Pattoor, India.

Alex. V
Electronics and Communication
Sree Buddha College of Engineering,
Pattoor, India.

***Abstract*— The purpose of edge detection is to reduce the amount of information in an image, while preserving the structural properties to be used for further image processing. Most often edge detection algorithms are implemented in software tools .The advancements in VLSI technology made the hardware implementations a good alternative. Exploiting the parallelism in algorithms and assigning complex task to hardware yields significant speedup in running times. The Canny edge detector is widely used in many applications due to its ability to extract significant edges with better detection, localization and performance for noise contaminated edges. But the frame level processing of image makes the computation of Canny algorithm complex and increases its latency. Hence a block based design with a block classification and adaptive threshold unit was considered. The latency for the block based algorithm now becomes the function of block size instead of frame size. In addition to the above design, the input image will be given a contrast boost with help of histogram equalization, in order to detect more edges. Further-more, hardware architecture of the proposed algorithm is also presented. The architecture design is also capable of processing blocks in parallel and supports fast edge detection of images and videos. The proposed design is coded in Verilog hardware description language and synthesized on Xilinx Virtex-5 FPGA using Xilinx ISE design suite 14.2. Simulation results are presented to illustrate the performance of block-based edge detector.**

Keywords*— *Canny Edge detector; Block-based Processing; Histogram equalization; FPGA.

I. INTRODUCTION

The most important task in any vision based system is the detection of proper edges in digitized images. A good edge detector responds to true edges and should be insensitive to noise. The realization of edge detector should be computationally efficient and effective. The most modern image processing applications requires more power and storage space. Edge detection is one among the preprocessing steps in many computer vision applications. It is used to detect sharp discontinuities in an image. Typical edge detection algorithms are implemented using software tools.

The advancements in Very Large Scale Integration (VLSI) technology made the hardware implementations, a better alternative for real-time applications. Edge detection identifies and locates sharp discontinuities in the image. The discontinuities include abrupt changes in pixel intensity which defines the boundaries of objects in an image. Existing edge detection algorithms involve, convolution of the image with edge detector operator, which is sensitive to maximum gradients in the image and returns values of zero in uniform and smooth regions. There is large number of edge detection operators available, each designed to be sensitive to peculiar type of edges.

Canny edge detector is considered as a standard edge detector because of its superior performance. The high performance is due to the hysteresis thresholding concept which computes high and low thresholds based on the whole image statistics. Unfortunately, the specified feature makes the Canny algorithm more computationally complex and additional pre-processing steps needs to be done on the entire frame image. Hence, direct implementation of the Canny algorithm has high latency and cannot be used in real-time applications. Canny algorithm is highly dependent on a correct setting of the threshold. They will miss some edges or detect some spurious edges when the threshold is not set a proper value. Hence, this algorithm is not good for mobile robot vision system in which all of the operation should be done by the robot controller and the environment changes rapidly. Another disadvantage of the commonly used Canny algorithm is the high computation cost. To solve the problems of Canny algorithm, an adaptive threshold selection unit, which calculates high and low thresholds for each block, was utilized. Each block can be executed in parallel, thereby decreasing the latency involved. The block based architecture allows distributed Canny edge detector to be pipelined with other block-level based codecs, and improves the timing performance of video/image processing applications.

Recently there has been an increased demand for good quality images in various research areas. Most of the applications such as image registration, region separation, and segmentation use edge detection as a pre-processing step for feature extraction. The real challenge is to find a way to enhance the noisy images and thereby extracting edges properly. The proposed design consists of mainly two steps. The first step is to enhance the contrast of the input block image and second is to extract the details of the edges. Image enhancement can be performed using histogram equalization technique.

The Canny algorithm has been implemented on wide variety hardware platforms. The algorithm was implemented using Handle-C software in [3], but the tool couldn't synthesis the algorithm, as it did not provide the synthesis. The implementation in [5] takes more memory access as it works on 4 pixels in parallel. The algorithmic level implementation

of a similar edge detector without the synthesis results were highlighted in [6]. An improved version of Verilog implementation of Canny edge detector without equalization was presented in [7].

The paper is organized as follows. Section II gives an overview of the parallel block based edge detector. Section III explains the FPGA implementation of the parallel block based architecture with histogram equalization. Section IV presents the obtained results, and compared with traditional canny detector, in both Matlab and Xilinx tools. It computes the PSNR (peak signal to noise ratio) values for quality analysis of proposed edge detector and also presents the synthesis results on Virtex 5 FPGA. Finally, conclusions are presented in section V.

II. PARALLEL BLOCK BASED CANNY EDGE DETECTION ALGORITHM

The important steps in the original Canny algorithm includes 1) Calculates the horizontal and vertical gradients (Gx and Gy) after convolving with gradient matrix. 2) The magnitude gradient and direction at each pixel location is computed. 3) A technique of non-maximal suppression is used to convert the blurred edges to sharp edges. 4) The high and low thresholds are calculated for determining the potential edges of the entire image. 5) The final edges are obtained by performing hysteresis thresholding so that the final edges are obtained by suppressing all edges that are not connected to strong edges. The basic block representation of Canny edge detection algorithm is shown in Fig. 1.

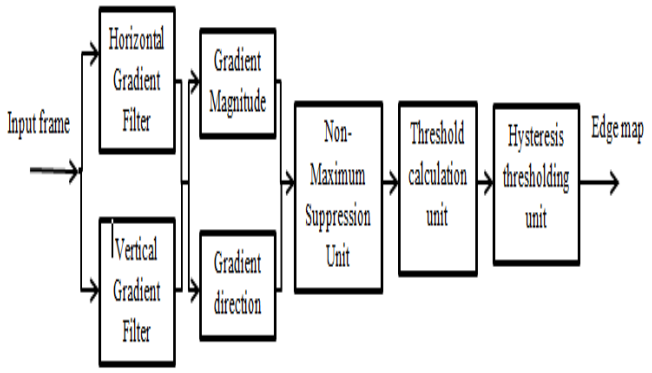


Fig. 1 Canny edge detector

The basic algorithm aims at three criteria as in [1]: detecting edges, locating proper edges, and responses from the input image. Comparison of this algorithm with other edge detectors like Sobel, Prewitt, Roberts etc., shows that Canny algorithm has better response in presence of noise. The thresholds are set on the basis of pixel distribution throughout the image. The Canny algorithm cannot be applied to blocks of image because of the fact that it assumes a percentage P_1 of total pixels in the block as true edges.

In order to improve the performance of edge detector at block level, a distributed Canny edge detector algorithm is considered. The block diagram is shown in Fig. 2. In this version as in [7], the input image is divided into m x m overlapping blocks and the blocks will be independently processed. This design has almost similar steps as the original algorithm except that it is now applied on blocks of image. Adaptive threshold calculation step is modified for parallel processing without affecting the edge detection performance.

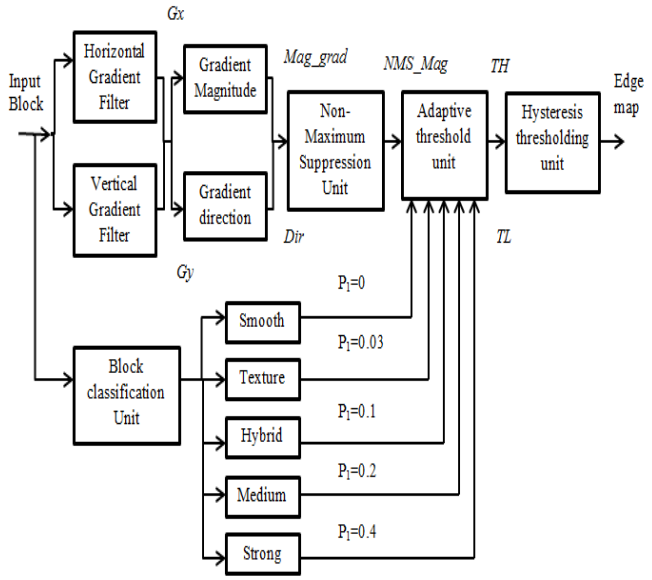


Fig. 2 Distributed Canny edge detector

The percentage value P_1 should be appropriate for threshold calculation and the values are selected, as in [7], for each block as shown in Table 1.

TABLE I
P_1 VALUES FOR VARIOUS BLOCK SIZES

Block size	Block Classification				
	Uniform	*Texture*	*Hybrid*	*Medium*	*Strong*
8×8	0	0.0312	0.1022	0.2183	0.482
16×16	0	0.0307	0.1016	0.2616	0.483
32×32	0	0.0305	0.1117	0.2079	0.485
64×64	0	0.0318	0.1060	0.2218	0.467
128×128	0	0.0302	0.0933	0.2375	0.484
256×256	0	0.0299	0.0911	0.2304	0.489

The computation of thresholds requires an accurately quantized magnitude histogram. The non-uniform quantizer, as proposed in [6], is used to obtain gradient magnitude histogram. 'n' reconstruction levels are computed as shown in [6], R1=(min+max)/2 and Ri+1 =(min+Ri)/2 (i=2,3,…..n), where min and max are the minimum and maximum values of gradient magnitude matrix respectively, and Ri is the reconstruction level.

III. FPGA IMPLEMENTATION OF PROPOSED ALGORITHM

The block diagram of the proposed algorithm is shown in Fig.3. Each block, obtained by dividing the image, is provided as an input to the computation engine shown in Fig.4. Each computation engine executes an m x m

overlapping block and generates the corresponding edge maps. There are six units for computation in the engine: 1) Histogram equalization 2) Block classification unit. 3) Gradient and magnitude calculation. 4) Directional non-maximal suppression unit. 5) Adaptive thresholding. 6) Hysteresis thresholding.

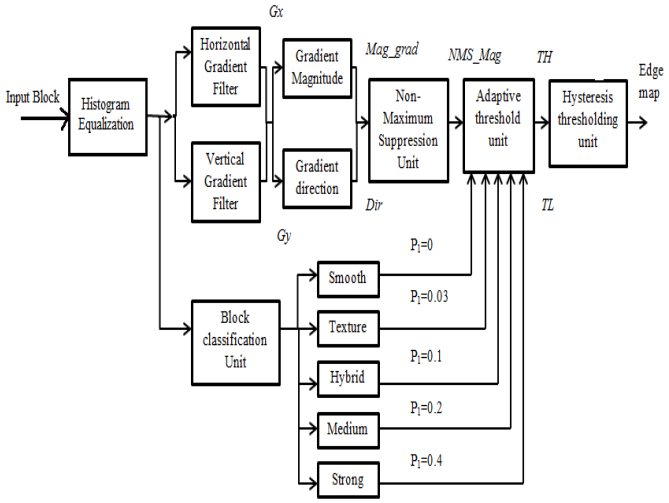


Fig.3 Proposed edge detector

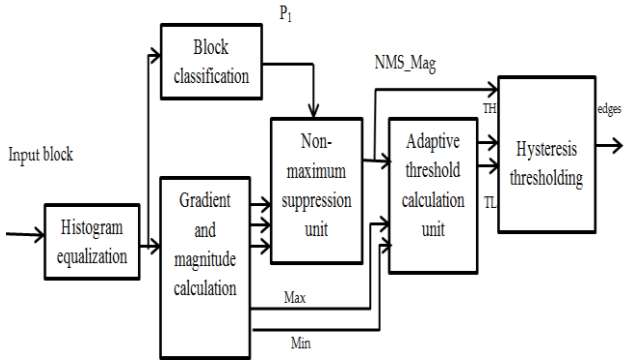


Fig.4 Block diagram of computation engine

A. Histogram equalization

It is a method used in image processing applications for adjusting contrast of the image using the histogram of the image. This method is mainly used for adjusting the image for easy analysis and improving visual quality. Histogram represents the frequency of occurrence of each pixel in an image. The cumulative distribution function is calculated as follows:

$$Cdf(i)=\sum p(j) \text{ for } j=0\ldots..i \qquad (1)$$

where p(j) is the probability of occurrence of a pixel. The histogram equalization can be performed as follows:

$$H(n) = ((cdf(n) - cdf(min))/((M \times N) - cdf(min)) \times 255) \qquad (2)$$

cdf(min) is minimum non-zero value of cumulative distribution function.(M x N) is the total number of pixels in the block of image.

(a) (b)

Fig. 5(a) Input image, (b) Equalized image

Fig.5 represents the necessity of image enhancement for edge detection. The image enhancement technique improves the sharpness of pixel values, and thereby highlights the details, which is an advantage for many edge detection algorithms. When image is represented by same pixel values it causes the degradation of the image quality.

B. Block Classification Unit

The architecture consists of two stages, the pixel classification unit and the block classification unit. For pixel classification, the variance is calculated as follows:

$$Variance=1/8 \sum (xi - mean)\,2 \qquad (3)$$

where xi is the pixel intensity. The local variance is then compared with Tu and Te [2] for pixel classification.

Step:

If (var(x, y) <=Tu)
Pixel type=uniform;
else if(Tu<var(x,y)<=Te)
Pixel type=texture;
else if(Te<var(x,y)
Pixel type=edge;

where var(x,y)= 3 x 3 local variance at each pixel (x, y),
Tu =100, Te=900, [2].

Based on the number of pixels, the input block image is classified into either of 5 types as shown in Table II.

TABLE II
BLOCK CLASSIFICATION

Block Type	Pixel type count
Smooth	Nuniform>=(307*Total_Pixel)/1024 & Nedge=0
Texture	Nuniform<(307*Total_Pixel)/1024 & Nedge=0
Hybrid	Nuniform<(665*(Total_Pixel-Nedge)/1024 & (Nedge>0 & (<307*Total_Pixel/1024))
Medium	Nuniform>=(665*(Total_Pixel-Nedge)/1024 & (Nedge>0 & (<307*Total_Pixel/1024))
Strong	Nuniform<=(716*Total_Pixel)/1024 & Nedge>=307*Total_Pixel/1024

Where, Total_Pixel=Total number of pixels in the block, Nuniform = Number of uniform pixels in the input block, Nedge = Number of edge pixels in the block.

C. *Gradient and magnitude calculation unit*

As these units are independent of each other they can work in parallel. The input is scanned and convolved with the 3 x 3 gradient mask to obtain the horizontal (Gx) and vertical gradients (Gy). The modulus of Gx and Gy are added to get the magnitude as shown in Fig. 6.

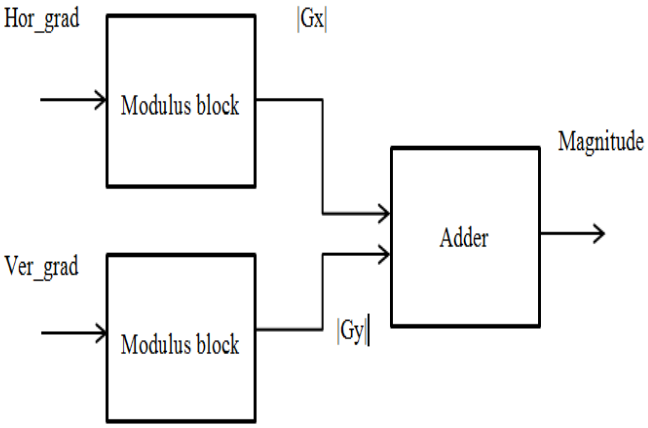


Fig. 6 Block diagram of magnitude calculation

D. *Directional non-maximum suppression unit*

The direction is calculated by dividing Gy by Gx. Here two intermediate gradient values are calculated along the direction and each magnitude pixel is compared with these two values. The gradient of the pixel that has maximum magnitude is passed to final edge map and others are suppressed. Architecture of the unit is shown in the Fig. 7.

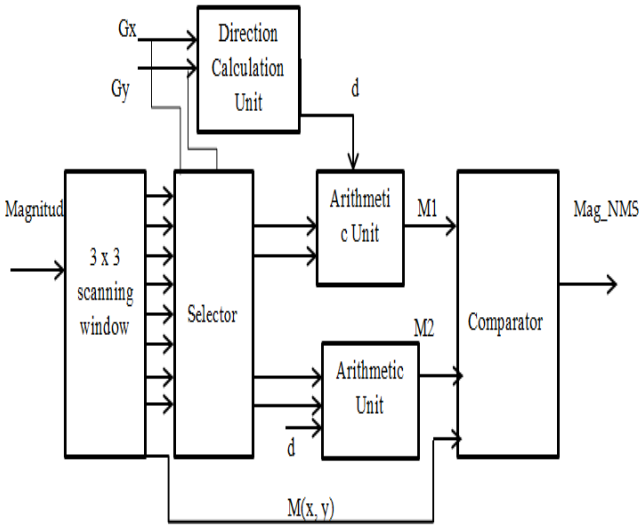


Fig.7 Architecture of Non-maximum suppression unit

E. *Adaptive threshold calculation unit*

A non-uniform quantizer is used to obtain the discrete cumulative distribution function and the thresholds are calculated based on the CDF. The shifters and adders are used to calculate the reconstruction levels Ri and the comparators and counters are used to count the number of pixels, which have magnitude equal to or less than the reconstruction level. The P_1 value will be multiplied by total pixels and the counter is initiated. The pixel count of each reconstruction level is compared with this count and the closest value is used to select level i. With the help of i, max, and min, the arithmetic unit computes the corresponding Ri, which is the high threshold TH. Low threshold is computed as 40% of TH. The pseudo code for adaptive threshold calculation unit is as follows:

Step 1:
P_1 =percentage of pixels in block that would be classified as strong edges.

```
If (smooth block_flag=1)
    P1=0;              //no edges
else if (texture block_flag=1)
    P1=0.03;// only few edges
else if (hybrid block_flag=1)
    P1=0.1;            //some more edges
else if (medium block_flag=1)
    P1=0.2;            //medium number of edges
else
    P1=0.4;            //many number of edges
```

Step 2: Compute non-uniform magnitude histogram and its corresponding cumulative distribution function.
Step 3: Compute high threshold
Step 4: Compute low threshold

The corresponding architecture is shown in Fig.8. Minimum magnitude value and output of shifter from previous stage are the input to the adder units.

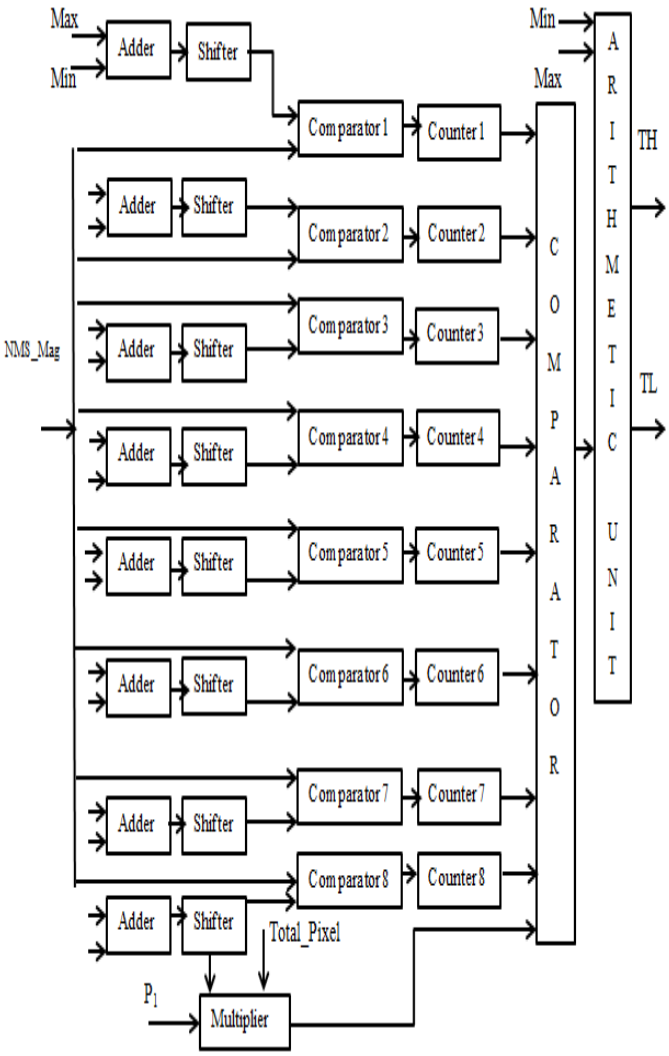


Fig.8 Architecture of adaptive threshold calculation unit

F. Hysteresis thresholding

If the pixels of NMS output matrix is above the upper threshold, it is marked as strong edge else as weak edge. The variable f1 represents strong edge and f2 represents weak edge. If any of the neighboring pixels is a strong edge then center weak pixel is marked as strong edge otherwise considered as non-edge pixel. The architecture is shown in Fig.9.

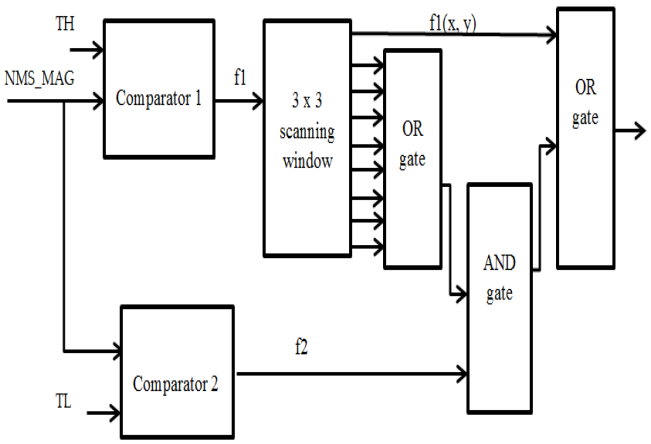


Fig.9 Architecture of Hysteresis Thresholding

IV. RESULTS

The edge detection performance can be evaluated by analyzing the obtained edge maps. Fig. 10 shows the edge maps of original Canny edge detector and Equalized edge detector. The algorithm was applied on a set of images shown below and validated the results. The parameter used for quantitative analysis is peak signal to noise ratio.

(a) (b) (c)

Fig. 10 .(a) Lena image (512 x 512) (b) Gray scale image (c) Canny edge map of original Lena image

(a) (b) (c)

Fig.11 a) Original Lena image b) Enhanced gray scale image c) Equalized edge map

(a) (b) (c)

Fig.12 a) Boat image (256x256) b) Gray scale image c) Canny edge map

(a) (b) (c)

Fig.13 a) Boat image b) Enhanced gray image c) Equalized edge map

(a) (b) (c)

Fig. 14 a) House image b) Gray scale image c) Canny edge map

(a) (b) (c)

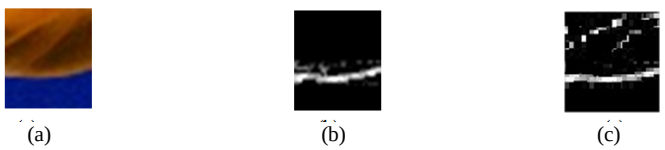
Fig. 15 a) House image b) Enhanced image c) Equalized edge map

(a) (b) (c)

Fig.16 a) Coin image b) Gray scale image c) Canny edge map

(a) (b) (c)

Fig. 17 a) Coin image b) Enhanced image c) Equalized edge map



(a) (b) (c)

Fig. 18 a) 32x32 input image b) edge map of distributed canny c) Equalized edge map

The above comparisons show that the equalized parallel block edge detector detects more edges than the conventional edge detectors. The simulation results were obtained using Matlab software and FPGA hardware simulation tool (Xilinx ISE software).

Peak signal to noise ratio is the ratio of maximum power of the signal to the power of noise that affects the quality of representation of the signal. The peak signal to ratio was computed for the edge detectors as follows:

$$PSNR=10 \log 10 (MAX^2 / MSE) \quad (4)$$

where MAX=maximum pixel value of the image, and MSE is the mean of square of errors. Mean square error is the cumulative squared error between the output image and the original image.

TABLE III
PEAK SIGNAL TO NOISE RATIO

Image Size	Edge detector	MSE	PSNR
256×256	Canny edge detector	0.04	61.99
	Equalized detector	0.03	62.82
512×512	Canny edge detector	0.05	61.59
	Equalized edge detector	0.04	62.66

TABLE IV
PSNR VALUES FOR 5 IMAGES

Image Number	PSNR Values	
	Original Canny	Equalized edge detector
1	56.53	59.96
2	58.33	61.60
3	55.10	55.61
4	55.65	58.77
5	56.99	66.49

Table III shows the mean square error values and the peak signal to noise ratio obtained for the Equalized and original Canny edge detector. From results, it can be observed that the Equalized edge detector has more peak to signal ratio and less mean square error. Thus, the proposed design shows better results and can be widely used in tool condition monitoring, and fault detection applications. In such applications low quality images needs to be processed for fault identification and the proposed design can be a robust alternative.

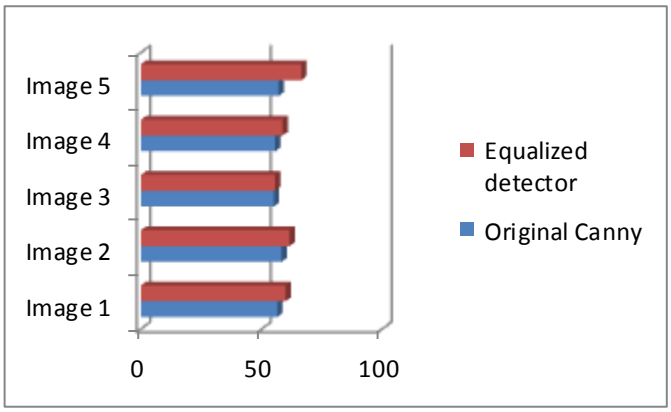


Fig. 19 Graph comparing PSNR values of 5 (512×512) images

A set of 5 test images were taken for computing the PSNR values. The graph shows that, the peak signal to noise ratio is comparatively higher for the Equalized edge detector than the original canny edge detector.

The x-axis of the graph implies the PSNR values and y-axis implies the image number. The FPGA (field programmable gate array) simulation result of the proposed design is shown in Fig. 20.

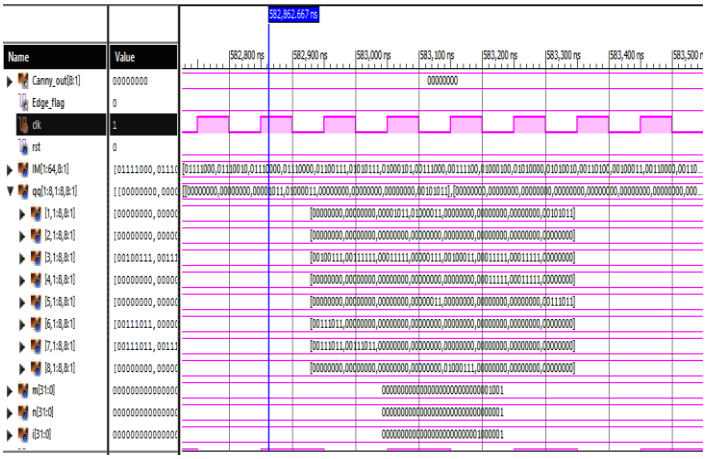


Fig.20. Verilog simulation of proposed edge detector.

The design was synthesized on Virtex 5 FPGA board using Xilinx ISE software tool. The obtained synthesis report for the design is shown in Table V.

TABLE V
SYNTHESIS REPORT

Device utilization summary			
Logic utilization	*Used*	*Available*	*Utilization*
No. of slice registers	14045	69120	20%
No. of slice LUTs	40489	69120	58%
No. of bonded IOBs	12	640	1%
No. of DSP48Es	5	64	7%

V. CONCLUSION

Original Canny edge detection algorithm depends on the entire image frame for threshold calculation and thus the latency is a function of frame size. But latency in the proposed design is the function of block size. In order to enhance the edge detection and reduce the latency involved in the design, the parallel block based architecture with equalized input was presented. The proposed design enhances the quality of the input image block and highlights more details in the block. The edge map of quality enhanced image was found to be more precise and resulted in better performance. The PSNR values also proved the quality of proposed edge detector. Simulation results presented in both Matlab and Xilinx software demonstrated that edges are more effectively detected in proposed design when compared to traditional Canny edge detector.

REFERENCES

[1] J. F. Canny, “A computation approach to edge detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 8, no. 6, pp. 769–798, Nov. 1986.
[2] J. K. Su and R. M. Mersereau, “Post-processing for artefact reduction in JPEG-compressed images,” in Proc. IEEE ICASSP, vol. 3. May 1995, pp. 2363–2366.
[3] D. V. Rao and M. Venkatesan, “An efficient reconfigurable architecture and implementation of edge detection algorithm using handle-C,” in Proc. IEEE Conf. ITCC, vol. 2. Apr. 2004, pp. 843–847.
[4] H. Neoh and A. Hazanchuck, “Adaptive edge detection for real-time video processing using FPGAs,” Altera Corp., San Jose, CA, USA, Application Note, 2005.
[5] C. Gentsos, C. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, “Realtime canny edge detection parallel implementation for FPGAs,” in Proc. IEEE ICECS, Dec. 2010, pp. 499–502.
[6] Q. Xu, C. Chakrabarti, and L. J. Karam, “A distributed Canny edge detector and its implementation on FPGA,” in Proc. DSP/SPE), Jan. 2011, pp. 500–505.
[7] Q. Xu, C. Chakrabarti, and L. J. Karam, “A distributed Canny edge detector:Algorithm and FPGA implementation,”IEEE Transaction on image processing,Vol.23,No.7,July 2014.