Hardware Software Co-Simulation of Canny Edge Detection Algorithm

Kazi Ahmed Asif Fuad
Post-Graduate Student
Dept. of Electrical & Electronic Engineering
American International University-Bangladesh

Shahriyar Masud Rizvi
Assistant Professor
Dept. of Electrical & Electronic Engineering
American International University-Bangladesh

ABSTRACT
Edge detection is a method to detect presence of an object's image- typically this is identified by sharp changes in pixel density. We realized Canny Edge Detection Algorithm, the most optimal edge detector, in FPGA hardware utilizing Hardware-Software Co-Simulation with the help of Simulink (Mathworks) and System Generator (Xilinx). We explored and utilized different edge detection operators, in addition to Sobel, which is the typical such operator, for gradient calculation (the primary edge detection process). After comparative analysis, we found both Sobel and Robert operators among the best with hardware realization of Robert operator utilizing less resources (LUT & Flip-Flops). All the different versions of the algorithm was synthesized for Spartan-6 LX16 FPGAs from Xilinx.

General Terms
Edge Detection Algorithm, Canny Edge Detection, H/W Software Co-Simulation.

Keywords
Canny Edge Detection, Sobel, Robert, Xilinx System Generator, Nexys3- Spartan 6 FPGA Board.

1. INTRODUCTION
Edge detection is used as an initial step for many image processing such as image enhancement, image segmentation, object tracking & motion analysis. Edge detection is a process of identifying an edge. The sharp change in image pixel intensity is considered as the edge of the image. Edges correspond to points in an image where the gray value changes significantly from one pixel to the next pixel. Usually, edge detection techniques are implemented using software but with the advancement in Very Large Scale Integration (VLSI) technology, hardware implementation of edge detectors has become an effective option for real-time applications. In many real-world applications, the use of Canny is predominant due to its ability to extract significant edges with good detection and good localization performance. But unfortunately, the Canny algorithm contains extensive pre-processing such as smoothing and post-processing steps such as NMS and is more computationally complex than other gradient based edge detection algorithms such as Roberts, Prewitt and Sobel algorithms. [1][2] In our research of comparative study of edge detection technique, we designed and implemented Canny Edge Detection Algorithm using Xilinx System Generator and Spartan 6 FPGA (within Nexys3 Board). Detailed Canny algorithm is explained and later on System Generator Implementation technique is discussed. Finally impact of different operator on the gradient calculation of Canny is shown with experimental data and resource utilization summery.

2. CANNY EDGE DETECTION ALGORITHM
The objective of edge detection is to significantly reduce the amount of data in an image, while keeping the original properties of the image to be used for further processing. Amongst several algorithms, this paper focuses on a particular algorithm developed by John F. Canny in 1986. [3][4] Though it is quite old, it has become one of the prominent edge detection methods and it is still in research for further improvement. The aim of Canny was to develop an algorithm that is efficient with regards to the following criteria:

1. Detection: The edge detector should respond only to edges, and should find all of them; no edges should be missed. This corresponds to maximizing the signal-to-noise ratio. [4]

2. Localization: The distance between the edge pixels as found by the edge detector and the actual edge should be as small as possible. [4]

3. Number of responses: An original edge should not result in more than one detected edge. [4]

The algorithm consists of 5 separate steps: **1.** Smoothing: Smoothing the image to remove noise. **2.** Calculating gradients: The edges should be found where the gradients of the image has high magnitudes. **3.** Non-maximum suppression: Only local maxima should be taken as an edge. **4**. Double thresholding: Prospective edges are determined by thresholding. **5.** Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not associated to a very definite edge. [3] [4]

2.1 Smoothing (Noise Reduction)
Generally images taken from a camera contains some amount of noise and to prevent that noise is mistaken for edges, noise must be reduced. That's why the image is first smoothed by applying a Gaussian filter.[5]The Gaussian smoothing filter, a 2-D convolution operator, is used to 'blur' the images and remove details and noise by convolving with the image using a kernel that represents a Gaussian ('bell-shaped') hump shape. As the image is like a collection of discrete pixels, discrete approximation to the Gaussian function is needed before the convolution is performed. [6] The kernel of a Gaussian filter with a standard deviation of $\sigma = 1.4$ is shown in Equation (2) and the Gaussian function is given in equation (1). JFC showed that Gaussian filter is the optimal filter for Gaussian noise. [3]

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

$$\boldsymbol{G(\sigma = 1.4)} = \frac{\mathbf{1}}{\mathbf{115}}\begin{bmatrix} \mathbf{2} & \mathbf{4} & \mathbf{5} & \mathbf{4} & \mathbf{2} \\ \mathbf{4} & \mathbf{9} & \mathbf{12} & \mathbf{9} & \mathbf{4} \\ \mathbf{5} & \mathbf{12} & \mathbf{15} & \mathbf{12} & \mathbf{5} \\ \mathbf{4} & \mathbf{9} & \mathbf{12} & \mathbf{9} & \mathbf{4} \\ \mathbf{2} & \mathbf{4} & \mathbf{5} & \mathbf{4} & \mathbf{2} \end{bmatrix} \quad (2)$$

2.2 Gradients Calculation

The Canny algorithm calculates edges where the grayscale intensity of the image changes the most. Gradients at each pixel in the smoothed image are determined by applying gradient based edge detection operator. There are many gradient based edge detector. Amongst them, Sobel is mostly used in Canny Edge Detection. [3] The gradient magnitudes are determined by applying Manhattan distance measure as shown in Equation (3) to reduce the computational complexity.

$$|G| = |Gx| + |Gy| \quad (3)$$

Where: Gx and Gy are the gradients in the x- and y-directions respectively. In this paper, we analyzed the impact of Sobel, Robert, Prewitt & Robinson operators which are given in figure 1. In a compass operator such as Robinson, total 8 masks are convolved with the image and the max is selected as gradient. [7]

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Operator

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Prewitt Operator

$$G_N = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_{NE} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Robinson Compass Operator

$$G_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Robert Operator

Figure 1: Different Gradient based Edge detection Operator.

2.3 Non-Maximum suppression (NMS)

The main objective of this step is to convert the "smoothed" edges in the image of the gradient magnitudes to "sharp" edges. Basically, this is implemented by keeping all the local maxima in the gradient image, and making others zero. The algorithm is for each pixel in the gradient image:

1. Compare the edge strength of the current pixel with the edge strength of the pixels in both positive and negative gradient direction. **2.** If the edge strength of the current pixel is greater than the neighbor pixels then keep the value of the edge strength otherwise make the pixel value zero.

2.4 Thresholding

The edge-pixels after the NMS are identified with their strength pixel-by-pixel in which many of them will probably be actual edges in the image, but some may be not. The simplest way to discern between these would be to use a threshold. The Canny edge detection algorithm uses double thresholding in which edge pixels greater than the high threshold are identified as strong; edge pixels weaker than the low threshold are deleted and edge pixels between the two thresholds are marked as weak.

2.5 Edge Tracking by Hysteresis

Strong edges is included in the final edge image where weak edges are included if and only if they are associated with strong edges. As noise and other small variations are unlikely to result in a strong edge, thus strong edges will be true edges. The weak edges can either be because of true edges or noise/color variations. Weak edges due to true edges are much more likely to be connected directly to strong edges BLOB-analysis (Binary Large OBject) can be used to find whether they are edge or not. The edge pixels are divided into connected BLOB's using 8-connected neighborhood. BLOB's containing at least one strong edge pixel are then preserved, while other BLOB's are suppressed.

3. H/W SOFTWARE CO-SIMULATION/ IMPLEMENTATION OF CANNY EDGE DETECTION

For or H/W Software Co-Simulation/Implementation, ISE 14.7 (Xilinx) & MATLAB 2012b (Mathworks) have been used. System Generator part of Simulink from MATLAB & Xilinx was the H/W configuration generating tool which is very useful to work in Image processing arena as it gives faster and smarter way to simulate systems close to real life hardware output. Figure 2 shows the full flow diagram of HW/Software environment for the Canny implementation process. As we can see from flow diagram, image data serialization & de-serialization is completed in Software (Simulink) and edge detection is implemented in H/W. Our primary objective is to implement the edge detection algorithm and find a better solution for existing system. H/W Software Co-Simulation Co-Design is a powerful way to realize this. This method allows one to do without any camera or VGA-monitor interfacing which minimizes the complexity of whole system setup. But then again, Xilinx Platform Studio-XPS allows interfacing VGA monitor with Nexyx3 board (not done in this paper). It's useful when system is designed for applications such as Face detection. Figure 4 shows the full H/W Software implementation system Generator diagram. [8]

3.1 Image From File Block

'Image From File' is used to access the input images where image file path is given and data type of image data is selected. If the image is an MxNxP array, then the block outputs a color image in which MxN are the number of rows & columns in each color pan.

3.2 Simulink Image Pre-Processing

Simulink Image Pre-Processing serializes the image data for the H/W as Xilinx System Generator serial Data transfer. 'Color Space Conversion' converts the color image into a grayscale image and the data which is in 2-D is converted into

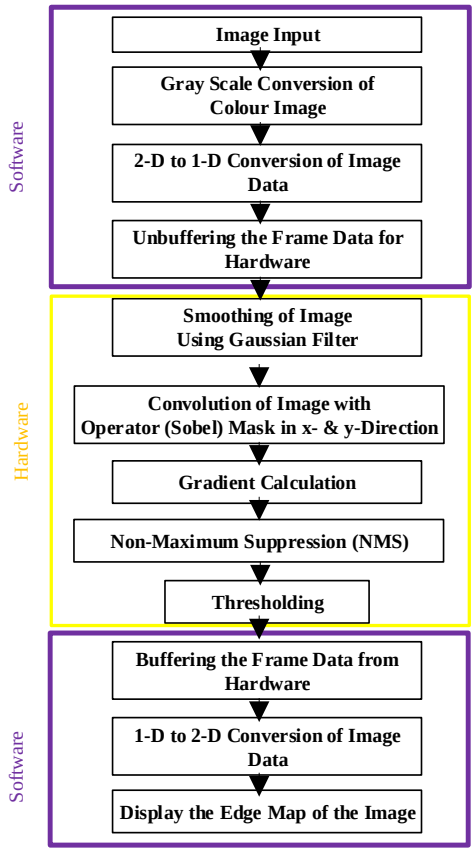

Figure 2: Flow Diagram of H/W-Software Implementation Process of Canny Edge Detection Algorithm.

1-D by 'Convert 2-D to 1-D' Simulink block whereas 'Transpose' block transposes the image data. 'Frame Conversion' & 'Unbuffer' block together sends the 1-D image data frame by frame serially to the H/W input. The 'Unbuffer' block converts the frame to scalar samples output at a higher sampling rate. Figure 3 shows Simulink Image Pre-Processing block.

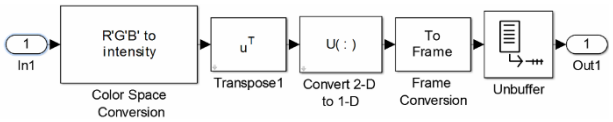

Figure 3: Simulink Image Pre-Processing block

3.3 System Generator System Design Blocks

3.3.1 Xilinx System Generator Token

Xilinx System Generator token is used for FPGA H/W functionality generation. It can generate (i) HDL, (ii) Bitstream, (iii) NGC & (iv) Hardware Co-Simulation from Xilinx System Generator blocks of Simulink. Both VHDL &Verilog can be generated by the token. It also allows 3rd party synthesis tool such as Synplify Pro from Synopsys Inc. [8]

3.3.2 Gateway In/Gateway Out

Gateway In and Gateway Out blocks are one of the basic blocks of System Generator which works as input and output respectively for the H/W design. These blocks are used to convert Simulink data type to Xilinx data type and vice versa. These blocks define top level input & output port respectively in the HDL design generated by System Generator. [8]

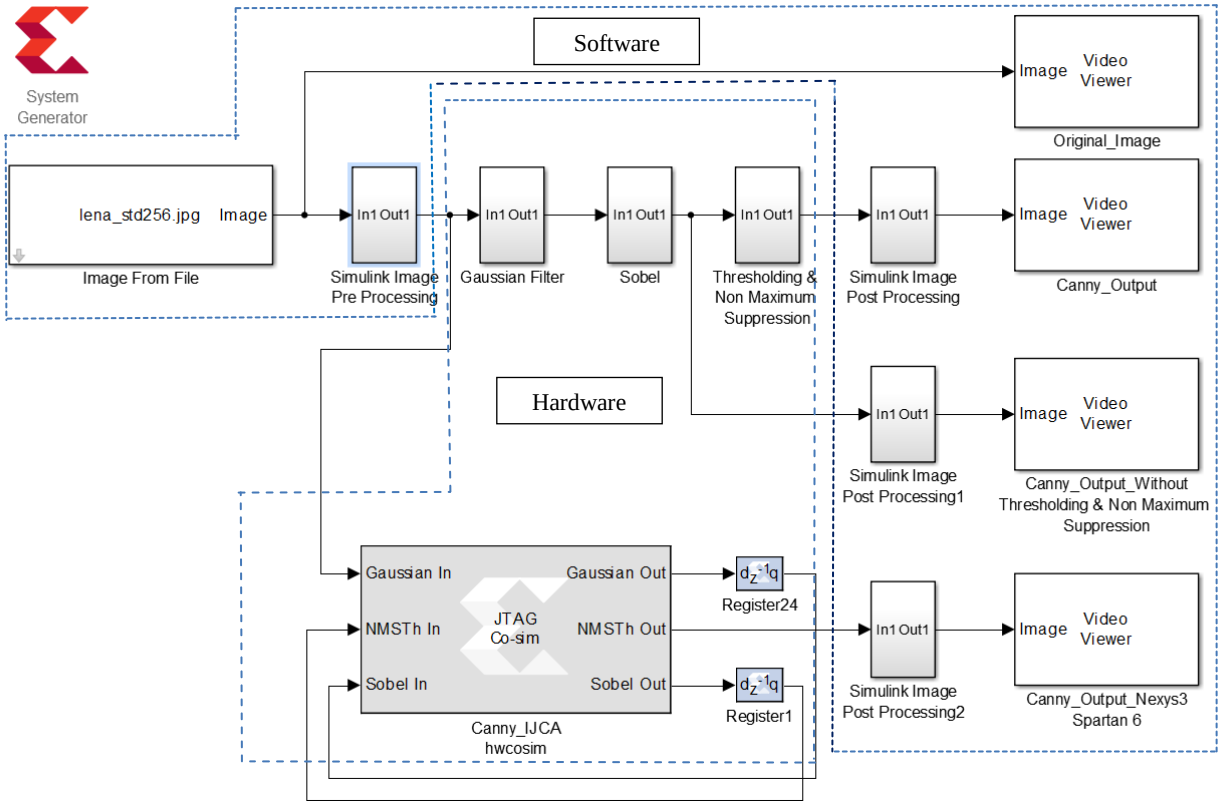

Figure 4: System Generator Block Diagram of H/W Software Implementation of Canny Edge Detection Algorithm

3.3.3 Convolution Blocks

Both Gaussian &Sobel uses convolution. From the study of convolution it is seen that it's actually a FIR filter. So for this purpose basic FIR filter was designed instead of using the System Generator built in blocks. Figure 5 shows how convolution is implemented in hardware. To create 2-D image data 'Virtex2 Line Buffer' is used. It takes data serially and buffers them up to certain numbers. Then each of line data is passed through FIR filter consists of Register (Flip-Flop), Multiplier, Adder and Constants blocks. [9]

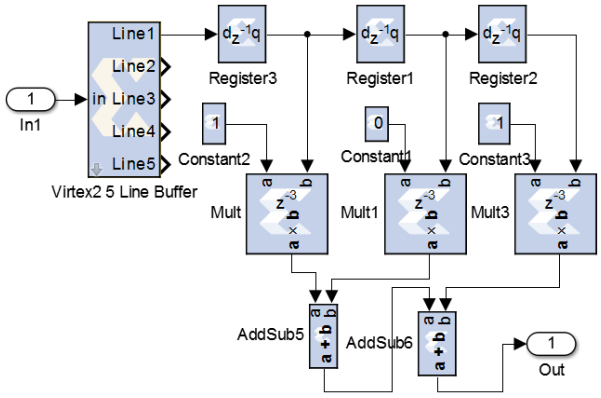

Figure 5: Convolution in Hardware

3.3.4 Thresholding& NMS

Thresholding and NMS, both are implemented using Comparator and Register block. Each of the image data is compared with a certain value if the image data is greater than the comparing value then that image data is transferred otherwise the value is set zero. In NMS, each image data is checked whether it's greater than its adjacent image data or not. If it's true then the image data is passed otherwise it is set zero.

3.3.5 JTAG Co-Sim

JTAG Co-Sim is the Hardware Co-Simulation Block generated by System Generator Token after the simulation of the whole system. This is the block which configures the FPGA, gets its output and displays it in Simulink.

3.4 Simulink Image Post-Processing

Simulink Image Post-Processing, which is opposite of Image Pre-Processing is used to transform the image data back to floating type. To de-serialize the image data, at first, 'Buffer' block is used to convert the scalar samples to frame output at lower sampling rate along with the 'Convert 1-D to 2-D' then the image data is transposed by the 'Transpose' block**.**

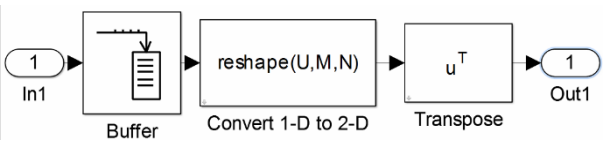

Figure 6: Simulink Image Post-Processing block

4. IMPACT OF DIFFERENT EDGE DETECTION OPERATORS

Output of Canny without Thresholding& NMS and final Canny output along with Krisch operators are given in the figure 7. For comparative study, 'Lena' and 'Cameraman' images were experimented. In Canny Edge Detection Algorithm, edges are mainly found by using gradient based edge detector Sobel operator. So one of our approaches was to find out impact of different gradient operators on the Canny. So, Robert, Prewitt & Robinson Compass operators were implemented into Canny algorithm in addition to Sobel and outputs were checked how they behave on the experimenting images. Device utilization is given in table 1. From visual analysis of images showed in figure 8, it is seen that with the use of Prewitt operator in Canny, the performance of Canny degrades. When only Prewitt operator was applied to the experimenting images, the result was close to the one generated by Sobel alone. But in Canny, Prewitt becomes noise prone as it detects false edges and Canny final output fails to detect original edges. On the other hand, performance of Robert operator in Canny is better in than Prewitt as it does not detects any false edge though it fails to detect low contrast edges. As Robert operator does not detect wrong edges our objective became to find a proper way to fit it into Canny. It's seen that edge detection in high contrast, edge detection using Robert was the same as Sobel. Our visual analysis shows that Canny edge detection without Thresholding & NMS using Robert is better than when only Robert operator was convolved with the image (as noise is reduced). So further analysis was done to get better output from Canny. First approach was to change NMS technique. 8-Point comparison NMS was applied and output result was not satisfactory as compared to 3-Point NMS. Increasing the point only increased delay but not quality. Varying the thresholding value does not improve the output as well. But without thresholding output was satisfactory as Robert itself is the simplest edge detection operator.

Table 1: Device utilization summary

Device Utilization Summary :Nexys3 Spartan-6 FPGA Board					
Selected device **6slx16csg324-3**		Canny Edge Algorithm with Sobel		Modified Canny Edge Algorithm with Robert	
Resource	Available	Used	%	Used	%
Number of Slice Registers:	18,224	3,071	16	2,356	12
Number used as Flip Flops:		2,629		2063	
Number used as AND/OR logics		442		293	
Number of Slice LUTs:	9,112	4,388	48	3,804	41
Number used as logic:	9,112	2,914	31	2,465	27
Number used as Memory:	2716	613	28	548	25
Number used as Single Port RAM:		384		320	
Number of occupied Slices:	2,278	1,472	64	1,254	55
Number of LUT Flip Flop pairs used:		4,968		4,155	

The calculation of Robert operator is simpler than other operators as only adder-subtractor is enough to get the gradient values. Sobel operator offers a lot of data for analysis but our modified Canny Algorithm with Robert operator provides much less data which is shown in figure 9. All the

designs are synthesized and implemented using ISE 14.7. From device utilization summary of selected chip 6slx16csg324-3, it shows that modified Canny Algorithm takes less area than original Canny Algorithm. Original Canny Algorithm uses 3,071 slices but on the other hand Modified Canny Algorithm uses 2,356 slices out of 28,224 slices available in our 6slx16csg324-3 FPGA and 4% slices are saved. 34% less combinational logics are used in Modified Canny Algorithm. Number of 'AND/OR' logic used by Canny Algorithm is 442 and Modified Canny Algorithm is 293. Use of 'Single Port RAM' is also decreased by 17%.

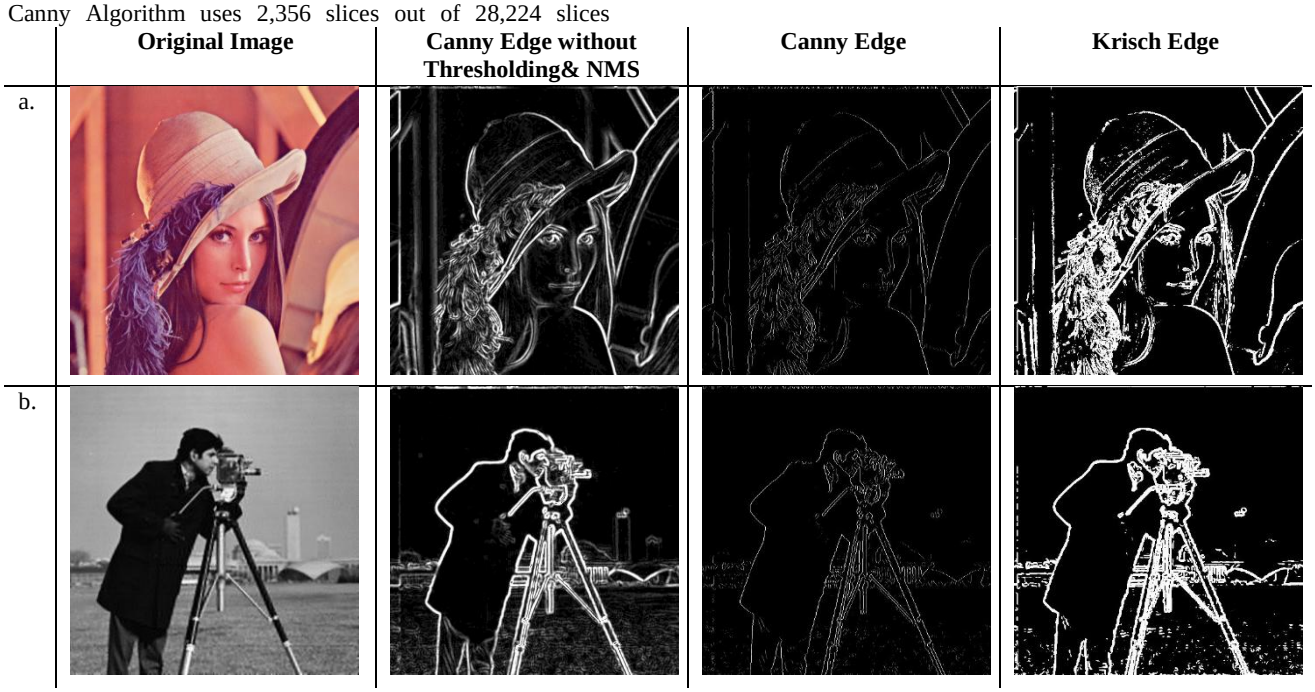

Figure 7: Output of Canny without Thresholding & NMS and final output along with Krisch operator of (a) Lena image (b) Cameraman Image

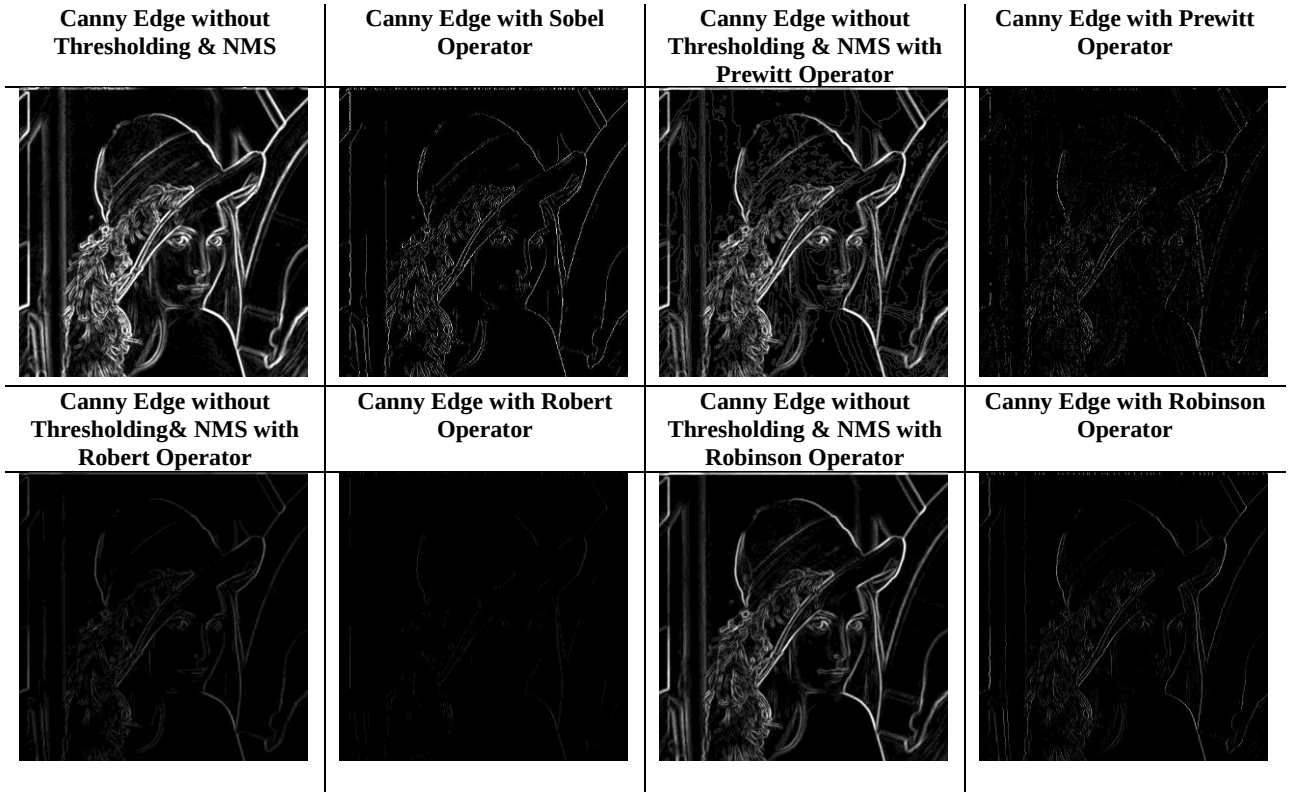

Figure 8: Canny edge detection output of 'Lena' image using different edge detection operator:

Prewitt, Robert & Robinson compass

Canny With Sobel Operator

Canny with Robert Operator and with only NMS

Figure 9: Canny output of 'Lena' image.

5. DISCUSSION

Sobel operator emphasizes on central pixels and that's why the edges are thicker and it can calculate edges at low contrast. On the other hand Robert operator calculates the edges so quickly by only addition and subtraction. Note that, Canny algorithm removes noises and also implement NMS. So combination of Canny algorithm and Robert operator gives a satisfactory output as image is smoothed by Gaussian filter, color changes are more distinct due to Robert operator. So, applications in which medium range contrasted images are primary objective and faster output & less device resources are required our modified Canny algorithm is an optimal edge detection algorithm.

6. CONCLUSION AND FUTURE WORK

After investigating different operators, we conclude that our Modified canny Algorithm is the optimal Algorithm. One drawback of Hardware/Software Co-Simulation is output is retrieved serially and becomes a slow process but to make the system more robust and faster we need to find an effective method. Now a days, Distributed implementation of system has become popular and it might make our Modified Canny Algorithm even more effective and efficient in Real-Time applications.

7. REFERENCES

[1] Qian Xu; Chakrabarti, C.; Karam, L.J., "A distributed Canny edge detector and its implementation on FPGA," Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE , vol., no., pp.500, 505, 4-7 Jan. 2011

[2] Qian Xu; Varadarajan, S.; Chakrabarti, C.; Karam, L.J., "A Distributed Canny Edge Detector: Algorithm and FPGA Implementation," Image Processing, IEEE Transactions on, vol.23, no.7, pp.2944,2960, July 2014.

[3] Canny, John, "A Computational Approach to Edge Detection," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.PAMI-8, no.6, pp.679, 698, Nov. 1986.

[4] Notes of Prof Thomas B Moselund. Available: http://www .cse.iitd.ernet.in/~pkalra/csl783/canny.pdf

[5] (2015) HIPR2. [Online]. [Visited on May, 2015] Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[6] (2015) Autonomous System Lab by William E. Green, Drexel University. [Online]. [Visited on May, 2015] Available: http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

[7] "Digital Image Processing" by Rafael C. Gonzalez, Richard E. Woods, 2nd Ed., Prentice Hall, 2002.

[8] System Generator for DSP User Guide by Xilinx. Downloadable From: http:// www.xilinx.com.

[9] S. Allin Christe, M. Vignesh, Dr. A. Kandaswamy, "An Efficient FPGA Implementation of MRI Image Filtering & Tumor Characterization Using Xilinx System Generator", International Journal of VLSI design & Communication Systems (VLSICS) Vol.2, No.4, December 2011.