Asian Journal of Convergence in Technology **Volume IV Issue II**
ISSN NO: 2350-1146 I.F-5.11

A Robust, Efficient FPGA based implementation of edge detection using Sobel mask

S. Siddhartha Raman
Department of Electrical and Electronics Engineering
BITS Pilani HyderabadCampus,
Hyderabad, Telangana
f20150572@hyderabad.bits-pilani.ac.in

Rahul Gottipati
Department of Electrical and Electronics Engineering
BITS Pilani HyderabadCampus,
Hyderabad, Telangana
f20150957@hyderabad.bits-pilani.ac.in

***Abstract*—Image Processing has traditionally been one of the most popular applications of FPGAs. Usually computationally expensive, image processing algorithms are best implemented on customizable hardware platforms like FPGA boards. The process of convolution has been a very powerful tool in identifying the response of a system given an input. The same thing can be extended to image processing in the sense that it is used for the purpose of finding the output image when acted upon by various filters [1]. In this paper, we have restricted ourselves to finding the resultant image when applied upon by sobel mask (to perform edge detection). The sobel mask is used so as to detect edges and identify whether the edges of an image are spurious. The coefficients of the mask are then convolved with the gray scale images so as to produce desired images. This paper presents a novel implementation of Edge detection using a Sobel mask on a Zynq 7000 board**

***Keywords*— FPGA, Sobel Mask, Edge Detection, Kernel, Vivado HLS**

I. Introduction

With the advancement in the technology in the Very Large Scale Industries, there has been a large increase in the number of hardware devices that have been coming up. One of the major devices that have been used is the Field Programmable Gate Array. It has a large number of LUT's, flip flops which can be used for the purpose of storing information as shown in Fig.1. This has been used for the purpose of implementing edge detection in this paper. The software that has been used primarily for this purpose is Vivado High Level Synthesis(Vivado HLS) wherein the target file is in Verilog language. The major reason behind using this is that a wide range of computer vision libraries are available and have been used for the purpose of detection of edges. For instance, HLS opencv header file has been used in the implementation of edge detection. Edge detection focuses on identifying points of sharp contrast in brightness in digital images. Edges are collections of such points that form curves and segments in an image. Discontinuities in image brightness can be attributed to a variety of scenarios such as a sudden/sharp change in depth or orientation of a surface, differences in illumination of neighbouring points etc. Edge detection helps weed out a lot of information that may not be relevant in many cases, such as when we are only concerned with the structural setup of the scene in an image. This corresponds only to the various surface boundaries of objects and how they are oriented with respect to each other, and excludes information about brightness, colour and other properties of image data points. The resulting image from applying edge detection is significantly smaller in size compared to the original image, and this can be of huge value since the data to for subsequent processing is reduced dramatically.

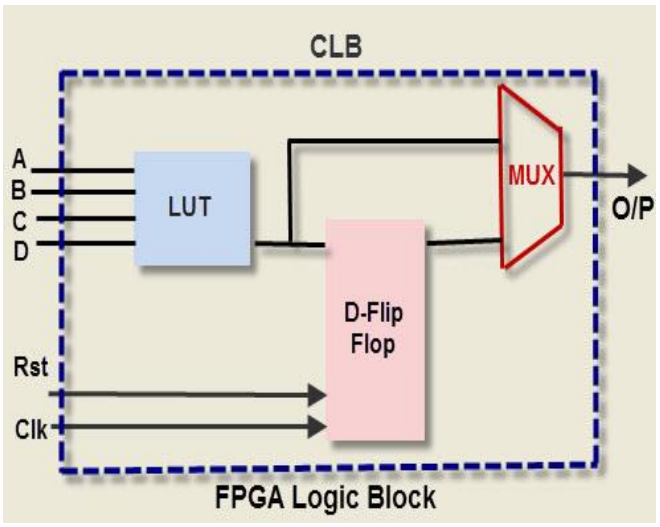
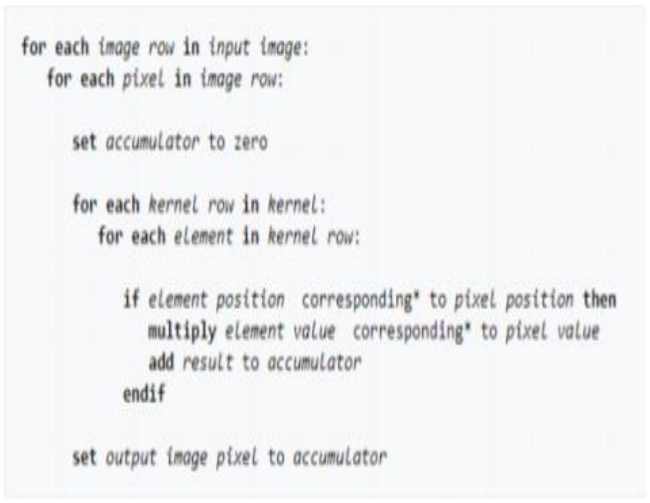


Fig.1 FPGA Block

There are several methods to detect edges in an image, which can be grouped into two broad categories : zero-crossing and search based. Zero crossing method deals with the Laplacian Operator. Laplacian operator is the second derivative of an image. The points where the Laplacian changes its sign (crossing zero) is noted and these points are generally the edges of an image. Search based edge detection is based on detecting edges, as a function of first order derivate of edge strength and then searching for local maxima of magnitude using gradient direction. Sobel mask based edge detection is a search based method, and is one of the most common practical image processing algorithms. Sobel Operator is a kernel based matrix operator which comprises of two 3x3 kernels that are convolved with the original image to yield approximate calculations of the gradient of image strength function - one for vertical edge discontinuities and other for horizontal edge discontinuities. The two can be combined to yield a single mask by appropriate selection of coefficients which will give appropriate focus to all directions of discontinuity in an image by adjusting the weights. This is then convoluted with the input image as shown in Fig.2. The averaging element in the Sobel mask helps in the smoothing out of an image by removing small scale random noise extremities, giving it an advantage over other masks that cannot usually accomplish this extra function. In mathematical form, the process of convolution [1] can be seen as Fig.3


```
for each image row in input image:
   for each pixel in image row:

      set accumulator to zero

      for each kernel row in kernel:
         for each element in kernel row:

            if element position  corresponding* to pixel position then
               multiply element value  corresponding* to pixel value
               add result to accumulator
            endif

      set output image pixel to accumulator
```

Fig.2 2D Convolution – Pseudocode

$$: \text{input} \otimes \text{kernel} = \sum_{y=0}^{\text{columns}} \left(\sum_{x=0}^{\text{rows}} \text{input}(x-a, y-b)\, \text{kernel}(x,y) \right)$$

Fig. 3. Mathematical representation

Where kernel(x,y) is the Mask and input(x,y) is the image.

There are many edge detection algorithms like Sobel edge
Detection, Canny edge detection, etc. The Sobel edge detection algorithm has been chosen because of the innate nature of it being simple. The mask just involves an approximation of finding the gradient of an image by moving the mask over the entire image[1]. One more advantage is that it is possible to identify the orientation of the edges. It is also less sensitive to noise which implies that the Sobel mask does not detect spurious edges which is one of the major advantages over other edge detection algorithms.

The paper starts with describing about the related works that have been performed earlier(Section 2) , the proposed idea (Section 3), results obtained (Section 4), the implementation of the algorithm on FPGA(Section 5), comparison with the results obtained from Santanu.et.al method and the results obtained from the method that we have used(Section 6) followed by the conclusion.

II. RELATED WORK

The edge detection algorithms are very much useful in real time applications like self-driving cars, classification of medical images, etc [2]. The edge detection algorithm just involves finding the edges of the number plates, self- driving cars so as to find the boundaries of the cars, etc. There have been other methodologies that have been proposed earlier like Santanu et.al method [3]. There have also been papers published on the process of convolution of images which forms the basis of most of image processing algorithms [4], [5]. The work that we have performed primarily focusses on detecting the edges using Sobel Masks.

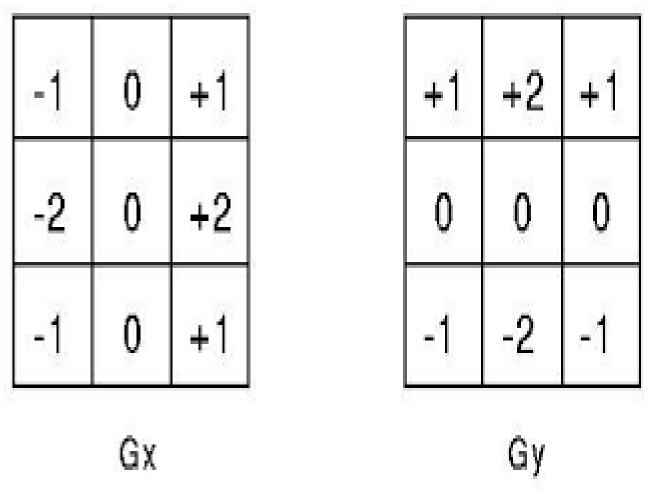


Fig. 4. Sobel Mask kernel

III. PROPOSED IDEA

The process of differentiation [6] in digital domain can be approximated using the following equations:

$$G(y) = f(x + 1, y - 1) - f(x + 1, y + 1) + 2 * (f(x, y - 1) - f(x, y + 1)) + f(x - 1, y - 1) - f(x - 1,y + 1) \quad (1)$$

$$G(y) = f(x + 1, y - 1) - f(x - 1, y - 1) + 2 * (f(x+1, y) - f(x-1, y)) + f(x + 1, y + 1) - f(x - 1,y + 1) \quad (2)$$

This in turn helps in finding the slope of the edge because

$$\tan(Gy/Gx) = \text{Slope} \quad (3)$$

Thus we get the orientation of the edge in the image. The mask shown in Fig.4 is moved on the image so that the differentiation of the pixel values is found along both x axis (Gx) and y axis (Gy). In the above equations, the gray scale value of the center pixel when the Sobel mask is moved is assumed to be f(x,y) and the x is increasing in the conventional x-axis direction whereas y is increasing in the conventional negative y-axis direction.

IV. IMPLEMENTATION OF PROPOSED IDEA

For the hardware implementation we have first used Vivado High Level Synthesis and then performed testing by using Sobel Mask [7]. The output image obtained after performing edge detection was then stored as a bmp file in the same folder as that of the input image, the sources folder of Vivado HLS. Then we have exported core in Vivado and generated an IP (Intellectual Property) using which generation of bitstream was performed. The generated bitstream was used along with the bit converted image, and the obtained values were fed into Xilinx SDK (Software Development Kit), where the process of convolution with the input stream was performed so as to give the output stream of bits. The output stream is then obtained by accessing a particular location on the memory. The values that were accessed are then converted back to the required output using MATLAB [8]. The flowchart is shown in Fig.5.

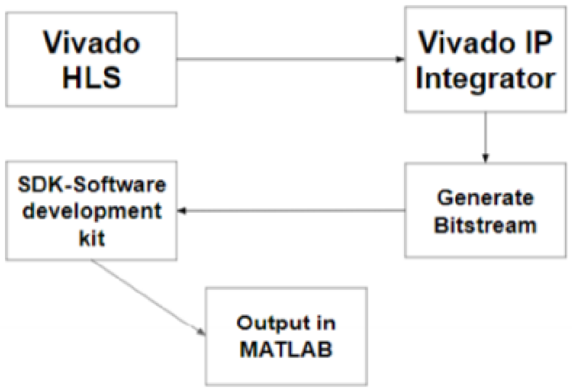


Fig.5 Flowchart of the implementation

A. *Implementation Logic*

The process of convolution requires that we need to use a sampling window of size which is same as that of the sobel mask kernel. This can be done by using a Line Buffer which is used in the implementation for the process of window weights and also that the size of the buffer is one less than the number of columns in the image [9]. The size that is used also determines the latency produced by the buffer. This is taken care by the program written in Vivado HLS. The figure 6 shows the line buffer for a sobel mask of 3*3 size. The process of convolution is just a process of multiplication and accumulation (MAC) [10]. It is obtained by the multiplication of the weights of the values obtained from the sampled window with the corresponding values in the kernel and in turn stored in the memory. The figure 7 shows the MAC.

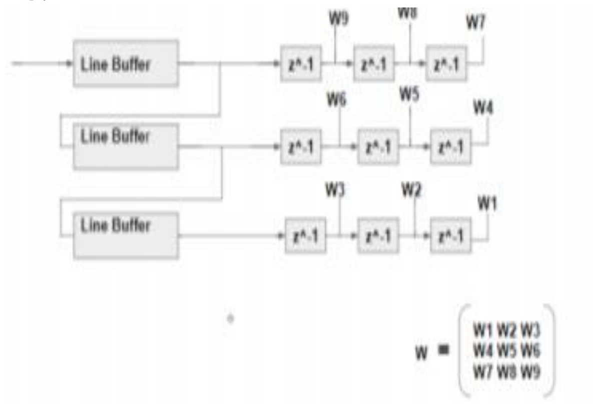


Fig.6 Line Buffer for Sobel Mask

B. *Vivado High Level Synthesis*

Vivado High Level Synthesis (HLS) is a very important tool that allows the users to implement the logic that is need
in high level languages like C++ or C. Before performing the implementation on Vivado HLS, we require buffer logic and window, this was implemented. A test-bench code, a function calling the sobel mask, a function performing the edge detection was used. There were various open CV libraries of Xilinx that were used for the purpose of implementation [12]. For instance HLS opencv header file was used in programming. The input image is shown in the figure 8.

The output image (as shown in Figure 9) obtained after performing edge detection is obtained below. As clearly shown, only edges or areas of high contrast discontinuity can be discerned and the rest are mostly blacked out. In this way ,the size of the output image has reduced drastically compared to the original and further image processing, if required, is made much easier.

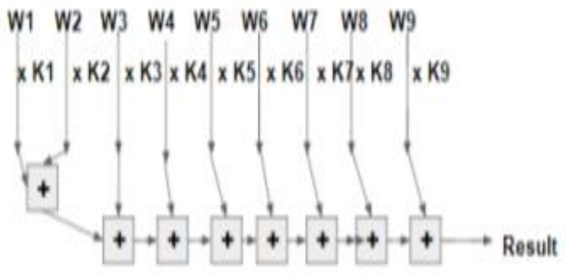

W1 W2 W3 W4 W5 W6 W7 W8 W9
x K1 x K2 x K3 x K4 x K5 x K6 x K7 x K8 x K9
+
Result

Fig.7 MAC

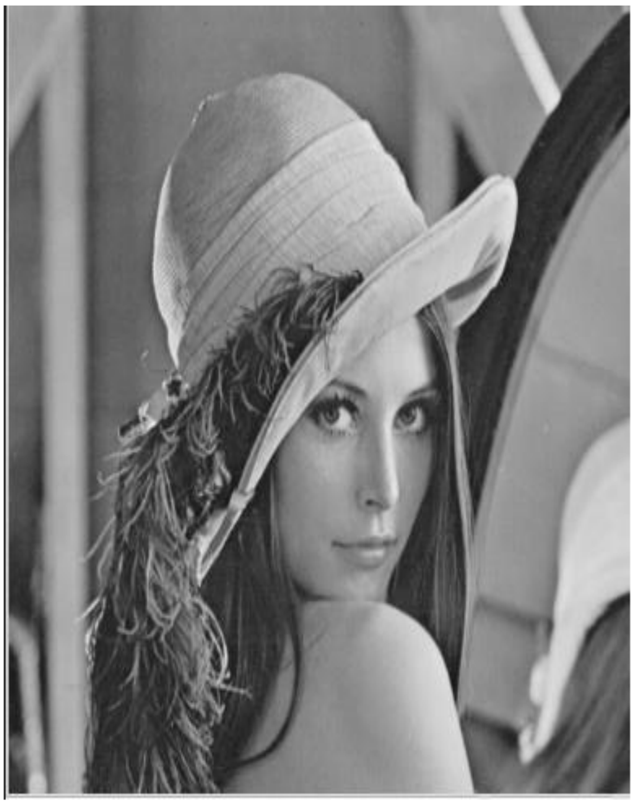


Fig.8 Input/ Original Image

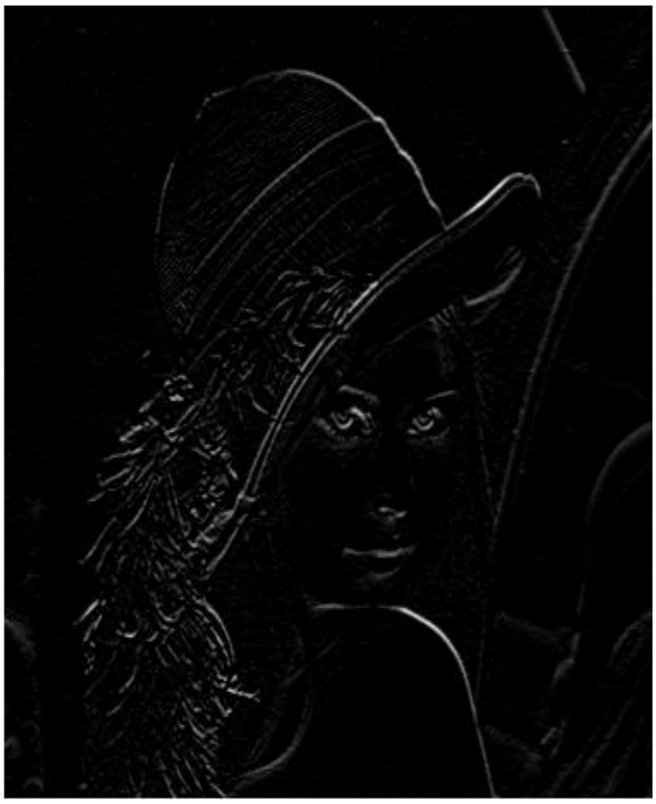


Fig. 9 Image obtained after performing edge detection

C. Vivado IP Generator

After performing the synthesis, running C simulation in HLS, the corresponding IP was exported from HLS. This is done so as to integrate with the Zyng 7000 Zedboard. This also requires Zynq processor IP so as to process the data that is given as input and also to interface the data stream with the AXI channel protocol that is utilized by the hardware. A DMA (Direct Memory Access) IP is also used so as to save the values that were accumulated and also to input the data to the IP core required for the processing of images. Timer was also included so as to determine the time elapsed for the process of execution [11].

The figure 10 shows the IP Generator Implementation. The IP design is validated following which a HDL wrapper is created and then bit stream is generated. The bit stream generation is followed by programming the device so as to check if the implementation is syntactically right. This is then followed by usage of Xilinx Software Development Kit.

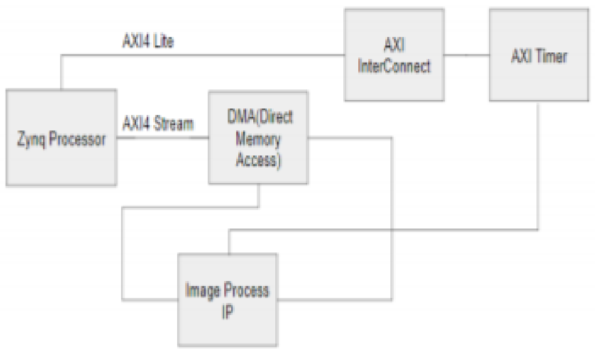


Fig.10 IP Generator

D. *Xilinx Software Development Kit*

Xilinx Software Development Kit has been used to send the input stream and to run the output. The input stream of the image that has been generated from MATLAB serves as an input to the function that has been called in the Xilinx Software Development Kit C++ file [13]. The bit stream generated is then loaded and the output's address location is
obtained from the values stored in DMA and then are accessed via XMD console and is later stored in a log file [14]. This log file is necessary because it contains the required pixel magnitude values in hexadecimal.

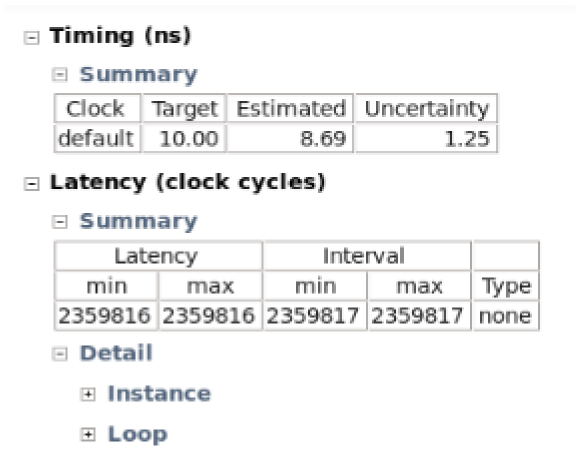
E. *MATLAB*

MATLAB has been used in the project for two main reasons:
The primary purpose is the generation of the input stream by conversion of the input image into a stream of pixels. The secondary purpose is the realization of the output log file (which stores the pixel values in hexadecimal form) which is generated from Xilinx SDK into desired output image. The process of realization of the output from its hexadecimal output requires a converter. This logic iterates in two loops because one is required for looping along the rows and the other is required for looping along the columns. This populates the 2D array with hexadecimal values which is then converted back to an image form.

V. RESULTS AND DISCUSSION

The following tables suggest the utilization results that were
obtained.

A. *Timing Summary*



Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
default	10.00	8.69	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
2359816	2359816	2359817	2359817	none

Detail

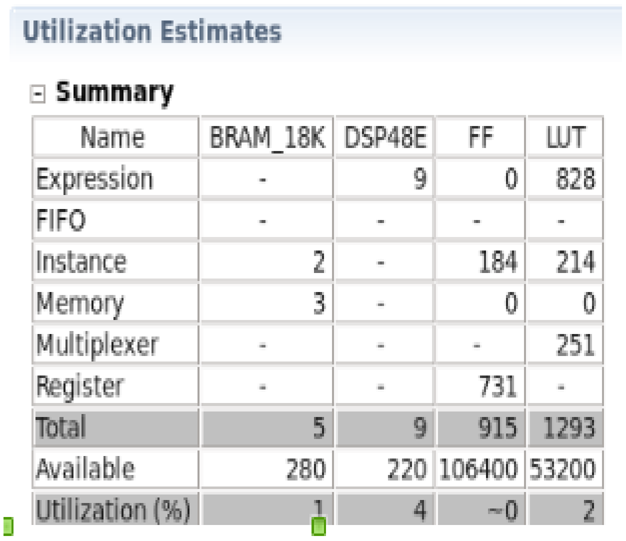
Instance

Loop

Fig.11 Timing summary

B. *Timing Summary*

The following table shows the results of HLS co-simulation.



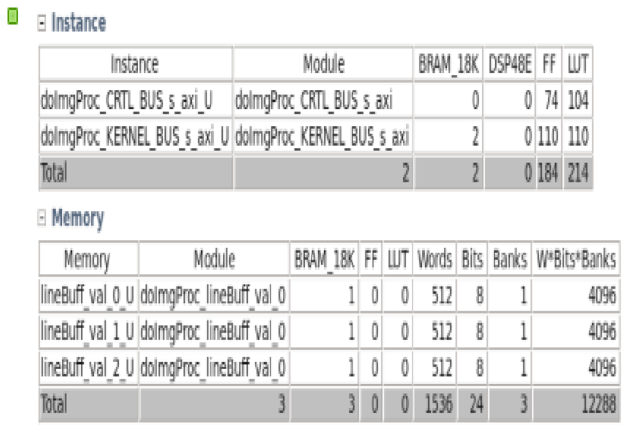
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	9	0	828
FIFO	-	-	-	-
Instance	2	-	184	214
Memory	3	-	0	0
Multiplexer	-	-	-	251
Register	-	-	731	-
Total	5	9	915	1293
Available	280	220	106400	53200
Utilization (%)	1	4	~0	2

Fig.12 Utilization Estimates

C. *Instances Results*



Instance

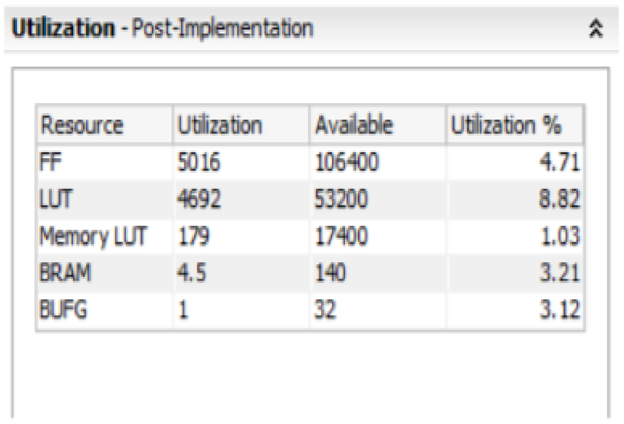
Instance	Module	BRAM_18K	DSP48E	FF	LUT
doImgProc_CRTL_BUS_s_axi_U	doImgProc_CRTL_BUS_s_axi	0	0	74	104
doImgProc_KERNEL_BUS_s_axi_U	doImgProc_KERNEL_BUS_s_axi	2	0	110	110
Total	2	2	0	184	214

Memory

Memory	Module	BRAM_18K	FF	LUT	Words	Bits	Banks	W*Bits*Banks
lineBuff_val_0_U	doImgProc_lineBuff_val_0	1	0	0	512	8	1	4096
lineBuff_val_1_U	doImgProc_lineBuff_val_0	1	0	0	512	8	1	4096
lineBuff_val_2_U	doImgProc_lineBuff_val_0	1	0	0	512	8	1	4096
Total	3	3	0	0	1536	24	3	12288

Fig.13 Instance results

D. *Post Implementation Results*



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
FF	5016	106400	4.71
LUT	4692	53200	8.82
Memory LUT	179	17400	1.03
BRAM	4.5	140	3.21
BUFG	1	32	3.12

Fig.14 Utilization Results

The existing method has the utilization of slice LUTs as 5.625 percent [3] whereas the method that we have used requires only a utilization of 2 percent as shown in Fig.12 which is nearly half the usage of the given method.

The existing method has a maximum frequency of 236 MHz [3] whereas the method that we have used gives a maximum frequency of 432 Mhz as shown in Fig.11 that is nearly twice the frequency of the existing method. This implies the computation is faster when compared to the existing one.

VI. CONCLUSION

The proposed design is able to compute at a faster rate when compared to the existing designs. This in turn leads to operations being performed at a faster frequency than the existing one [3]. Hence, the time to process an image is less comparatively. When this is used for larger computations, the process can show a significant difference. The time complexity of the algorithm that has been used is pretty less than the other algorithms for hardware implementation of Sobel edge detection. The usage of lesser number of LUTs leads to a conclusion that utilization of the resources are more efficient in comparison to the existing technology. This in turn leads to longer lasting of the resources that are existing.

VII. FUTURE SCOPE

One way to move forward would be: The latency in the processing of the images can further be reduced by performing the convolution process in a parallel fashion by breaking down the existing image into different portions so that the processing of each part is done in a separate core and then later make provisions for getting back the parts of the image.

VIII. ACKNOWLEDGEMENT

S. Siddhartha Raman and Rahul Gottipati are thankful to Dr. S.K. Sahoo for giving us the opportunity to explore implementation of algorithms used in computer vision and image processing to FPGA. This has helped us broaden our view of the cutting edge research being done in this area as well as given us practical experience in working with implementing real world research problems on FPGAs.

REFERENCES

[1]B. Draper, R. Beveridge, W. Bhm, C. Ross and M. Chawathe. Implement-ing Image Applications on FPGAs. International Conference on Pattern Recognition, Quebec City, Aug. 11-15, 2002.

[2]Johnston, C. T., K. T. Gribbon, and D. G. Bailey. Implementing image processing algorithms on FPGAs.. Proceedings of the Eleventh Elec-tronics New Zealand Conference, ENZCon04. 2004 .

[3]Santanu Halder, Debotosh Bhattacharjee, Mita Nasipuri, Dipak Kumar Basu. A Fast FPGA Based Architecture for Sobel Edge Detection.. Progress in VLSI Design and Test pp 300-306.

[4]Guo, Zhengyang, Wenbo Xu, and Zhilei Chai. Image edge detection based on FPGA. In Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Sympo-sium on, pp. 169-171. IEEE, 2010.

[5]S.Brown, J.Rose, Architecture of FPGA's and CPLD's ; a tutorial IEEE Des.Test Comput.13(2)(1996)4257

[6]R.O.Duda,P.E.Hart, et al.Pattern classification and scene analysis vol.3, Wiley New York,1973

[7]J.C. Russ, The Image Processing Handbook, CRC press, 2015

[8]R.C. Gonzalez, R.E. Woods, S.L. Eddins, Digital image processing using MATLAB(2004)

[9]A.G. Vicente, I.B. Munoz, P.J. Molina, J.L.L. Galilea, Embedded vision modules for tracking and counting people,Instrum,Meas. IEEE Trans. 58(9)(2009) 30043011

[10] J. W. Pierre, A novel method for calculating the convolution sum of two finite length sequences in IEEE Transactions on Education, vol. 39, no 1, pp. 77-80, Feb 1996.

[11] M. Boo, E. Antelo, J. Bruguera, Vlsi implementation of an edge detector based on sobel operator, EUROMICRO 94. System Architecture and Integration. Proceedings of the 20th EUROMICRO Conference. IEEE, 1994, pp. 506512.

[12] Guobo Xie and Wen Lu, Image Edge Detection Based On Opencv, International Journal of Electronics and Electrical Engineering Vol. 1, No. 2, June 2013.

[13] M. Balaji, S. Allin Christe, FPGA Implementation of Various Image Processing Algorithms Using Xilinx System Generator, Computational Intelligence in Data Mining – Volume 2 pp 59-68

[14] Nazma Nausheen, Ayan Seal, Pritee Khanna, Santanu Halder, A FPGA based implementation of Sobel edge detection, Elsevier 04, vol.27, no.2.