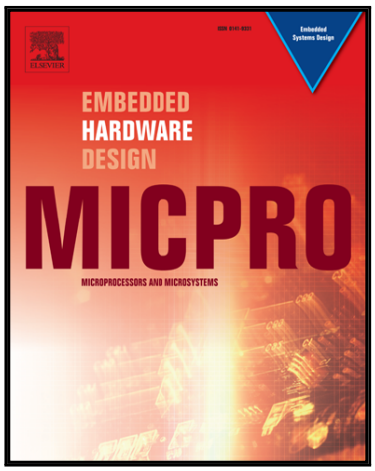

A FPGA based Implementation of Sobel Edge Detection

Nazma Nausheen, Ayan Seal, Pritee Khanna, Santanu Haldar

PII: S0141-9331(16)30228-9
DOI: 10.1016/j.micpro.2017.10.011
Reference: MICPRO 2629

To appear in: *Microprocessors and Microsystems*

Received date: 26 September 2016
Accepted date: 26 October 2017

Please cite this article as: Nazma Nausheen, Ayan Seal, Pritee Khanna, Santanu Haldar, A FPGA based Implementation of Sobel Edge Detection, *Microprocessors and Microsystems* (2017), doi: 10.1016/j.micpro.2017.10.011

Highlights

- A FPGA based architecture for Sobel Edge Detection algorithm is proposed.
- An 8-bit architecture is proposed to retrieve the addresses of pixels involved in convolution process.
- The proposed architectures reduce the time and space complexity compare to two existing architectures.

Noname manuscript No.
(will be inserted by the editor)

A FPGA based Implementation of Sobel Edge Detection

Nazma Nausheen · Ayan Seal · Pritee Khanna · Santanu Haldar

the date of receipt and acceptance should be inserted later

Abstract This paper presents an architecture for Sobel edge detection on Field Programmable Gate Array (FPGA) board, which is inexpensive in terms of computation. Hardware implementation of the Sobel edge detection algorithm is chosen because hardware presents a good scope of parallelism over software. On the other hand, Sobel edge detection can work with less deterioration in high level of noise. A compact study is also been done based on the previous methods. The proposed architecture uses less number of logic gates with respect to previous method. Space complexity is also reduced using proposed architecture.

Keywords FPGA · Edge detection · Sobel operator · Hardware architecture

1 Introduction

Edges are basically the noticeable variation of intensities in an image. Edges help to identify the location of an object and the boundary of a particular entity in the image. It also helps in feature extraction and pattern recognition. Hence, edge detection is of great importance in computer vision. So far, most of the researchers have chosen software for implementation of basic edge detection algorithms and their variations [2]. But, it has been established that it is not an efficient approach for real time applications. Implementation of edge detection algorithms on hardware platform is more efficient for real time applications. With the advancement of VLSI technology, hardware implementation

Nazma Nausheen · Ayan Seal · Pritee Khanna
Computer Science and Engineering
PDPM Indian Institute of Information Technology, Design and Manufacturing Jabalpur
Madhya Pradesh, India, Pin-482005 E-mail: ayan@iiitdmj.ac.in

Santanu Haldar
Department of Computer Application Kalyani Government Engineering College
Kalyani, Nadía, West Bengal-742101, India

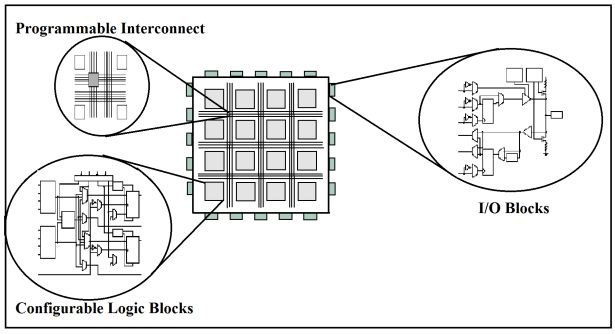

Fig. 1 Architecture of FPGA board adopted from [4]

presents a scope to parallelize subroutines in a program. Hence, hardware implementation provides much faster alternative as compared to software. In 1994 Boo et al. [3] have proposed hardware implementation of edge detection using Sobel operator using VLSI technology within application specific integrated circuit (IC). In the past few years extensive work has been done in the area of Field programmable Gate Array (FPGA) based implementation of edge detection algorithm which in turn implements image processing in real time. In 2009 real time algorithms have been designed to detect edges on FPGA board [12]. Parallelism of any algorithm is possible due to integration of large number of transistors (10k-100k) on a single silicon chip using VLSI technology [10]. All embedded systems are designed and implemented on Application Specific Integrated Circuit (ASIC) or FPGA. FPGA is an IC with Configurable Logical Blocks (CLBs) [5]. CLBs are interconnected using routing channels on a silicon board based on the desired operation. The internal connections can be configured using Hardware Description Languages (HDL) like VHDL [9] or Verilog. The silicon board consists of input/output ports on the boundary to take the input and provide the output as shown in figure 1. These HDLs provide a medium of simulation of the designed IC to see if anything can go wrong. These programmed ICs are emulated on FPGA board after correct simulation. In 2007, T.A.Abbasi et al. [1] presented an architecture for Sobel edge detection on FPGA. Later in 2012, this architecture was found to be inefficient with respect to space and time complexity by S.Halder et al. [8]. Architecture proposed by them saved time and took lesser space than the architecture proposed by T.A.Abbasi et al. [1]. However, this architecture also had disadvantages of redundant storage of pixels and also there is scope of reduction in the architecture. The motivation of proposed work is to overcome these shortcomings of architecture proposed in [8]. The contributions of the present work are as follows:

- An 8-bit architecture has been proposed to retrieve the addresses of pixels involved in convolution process for reducing the space complexity in the convolution process.
- A modified architecture by replacing some components in the architecture proposed by [8] for reducing the time complexity of Sobel edge detection algorithm.

The organization of the paper is as follows. Section II gives a brief introduction of Sobel edge detection algorithm. Section III presents a modified version of the traditional Sobel edge detection algorithm. Section IV depicts the methodology followed in this work. Experimental results and discussion are presented in Section V. Finally, Section VI concludes this work.

2 Sobel Edge Detection Algorithm

Sobel Edge detection algorithm is a gradient based edge detection method [6]-[7], which finds edges using horizontal mask (HM) and vertical mask (VM) [11]. One mask is simply transpose of the other as follows:

$$VM = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, HM = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

For the convolution process, an image is scanned from left to right and top to bottom of an image using HM and VM separately. Convolution is the process multiplying each intensity value of an image with its local neighbors, weighted by the mask. Let $[P]_{3\times3}$ be a sub-window of an image, as in figure 2, to be convolved with HM and VM separately.
The horizontal gradient G_x and the vertical gradient G_y of the center pixel (P_4) of sub-window are given by equations 1 and 2 respectively.

$$G_x = f_1 - f_2 \tag{1}$$

where $f_1 = (P_6 + 2 \times P_7 + P_8)$, $f_2 = (P_0 + 2 \times P_1 + P_2)$

$$G_y = f_3 - f_4 \tag{2}$$

P_0	P_1	P_2
P_3	P_4	P_5
P_6	P_7	P_8

Fig. 2 A 3×3 subwindow of an image

where $f_3 = (P_2 + 2 \times P_5 + P_8)$, $f_4 = (P_0 + 2 \times P_3 + P_6)$
The resultant gradient of the center pixel of the sub-window is given by equation 3.

$$G = |G_x| + |G_y| \tag{3}$$

The gradient of P_4 is further used to determine whether it is an edge pixel or not by comparing it with the predefined threshold value. If the gradient of P_4 is greater than threshold value, then P_4 is considered to be an edge pixel (denoted as 1). Otherwise, P_4 is treated as non-edge pixel (denoted as 0). This is shown in equation 4.

$$D_{op} = \begin{cases} 1, & \text{if } G > T \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

3 Simplification of the Traditional Sobel Edge Detection Algorithm

A few adjustments are required in traditional Sobel edge detection algorithm because 8-bit architecture has been adopted here for hardware implementation. In [8], S. Halder et al. simplified the traditional algorithm. Some changes have been done in the simplified version of algorithm proposed by S.Halder et al. [8]. In equation 1, the values of f_1 and f_2 will be maximum when each of the contributing pixels of sub-window will have maximum intensity values, i.e., 255. The same happens with f_3 and f_4 in equation 2. So, the maximum possible value of f_1, f_2, f_3 and f_4 is 4×255, thus it will require 10-bit architecture for implementation.To resolve this issue, values of f_1, f_2, f_3 and f_4 are limited to one fourth of the original values. Hence, each of the contributing pixels has to be divided by 4 before finding out values of f_1, f_2, f_3 and f_4, as shown in equations 5 to 8.

$$f_1 = \frac{1}{4}P_6 + \frac{1}{2}P_7 + \frac{1}{4}P_8 \tag{5}$$

$$f_2 = \frac{1}{4}P_0 + \frac{1}{2}P_1 + \frac{1}{4}P_2 \tag{6}$$

$$f_3 = \frac{1}{4}P_2 + \frac{1}{2}P_5 + \frac{1}{4}P_8 \tag{7}$$

$$f_4 = \frac{1}{4}P_0 + \frac{1}{2}P_3 + \frac{1}{4}P_6 \tag{8}$$

For the same reasons, G_x and G_y has to be divided by 2 and the new equation for G is given as:

$$G = \frac{1}{2}|G_x| - \frac{1}{2}|G_y| \tag{9}$$

Now, equations 1, 2 and 3 can be rewritten as:

$$G_x = f_1 - f_2 \tag{10}$$

where $f_1 = (\frac{1}{8}P_6 + \frac{1}{4}P_7 + \frac{1}{8}P_8)$, $f_2 = (\frac{1}{8}P_0 + \frac{1}{4}P_1 + \frac{1}{8}P_2)$

$$G_y = f_3 - f_4 \tag{11}$$

where $f_3 = (\frac{1}{8}P_2 + \frac{1}{4}P_5 + \frac{1}{8}P_8)$, $f_4 = (\frac{1}{8}P_0 + \frac{1}{4}P_3 + \frac{1}{8}P_6)$
The following four cases have to be considered because the absolute value of G_x and G_y are calculated in equation 3.
Case 1: If both G_x and G_y are positive.

$$\begin{aligned} G_1 &= (f_1 - f_2) + (f_3 - f_4) \\ &= \cancel{\frac{1}{8}P_6} + \frac{1}{4}P_7 + \frac{1}{8}P_8 - \frac{1}{8}P_0 - \frac{1}{4}P_1 - \cancel{\frac{1}{8}P_2} + \cancel{\frac{1}{8}P_2} + \frac{1}{4}P_5 + \frac{1}{8}P_8 \\ &- \frac{1}{8}P_0 - \frac{1}{4}P_3 - \cancel{\frac{1}{8}P_6} \\ &= (\frac{1}{4}P_5 + \frac{1}{4}P_7 + \frac{1}{4}P_8) - (\frac{1}{4}P_0 + \frac{1}{4}P_1 + \frac{1}{4}P_3) \\ &= f_5 - f_6 \end{aligned} \tag{12}$$

where $f_5 = (\frac{1}{4}P_5 + \frac{1}{4}P_7 + \frac{1}{4}P_8)$ and $f_6 = (\frac{1}{4}P_0 + \frac{1}{4}P_1 + \frac{1}{4}P_3)$
Case 2: If G_x is positive and G_y is negative.

$$\begin{aligned} G_2 &= (f_1 - f_2) + (f_4 - f_3) \\ &= \frac{1}{8}P_6 + \frac{1}{4}P_7 + \cancel{\frac{1}{8}P_8} - \cancel{\frac{1}{8}P_0} - \frac{1}{4}P_1 \\ &- \frac{1}{8}P_2 + \cancel{\frac{1}{8}P_0} + \frac{1}{4}P_3 + \frac{1}{8}P_6 - \frac{1}{8}P_2 - \frac{1}{4}P_5 - \cancel{\frac{1}{8}P_8} \\ &= (\frac{1}{4}P_3 + \frac{1}{4}P_6 + \frac{1}{4}P_7) - (\frac{1}{4}P_1 + \frac{1}{4}P_2 + \frac{1}{4}P_5) \\ &= f_7 - f_8 \end{aligned} \tag{13}$$

where: $f_7 = (\frac{1}{4}P_3 + \frac{1}{4}P_6 + \frac{1}{4}P_7)$ and $f_8 = (\frac{1}{4}P_1 + \frac{1}{4}P_2 + \frac{1}{4}P_5)$
Case 3: If G_x is negative and G_y is positive.

$$\begin{aligned} G_3 &= (f_2 - f_1) + (f_3 - f_4) \\ &= \cancel{\frac{1}{8}P_0} + \frac{1}{4}P_1 + \frac{1}{8}P_2 - \frac{1}{8}P_6 - \frac{1}{4}P_7 \\ &- \cancel{\frac{1}{8}P_8} + \frac{1}{8}P_2 + \frac{1}{4}P_5 + \cancel{\frac{1}{8}P_8} - \cancel{\frac{1}{8}P_0} - \frac{1}{4}P_3 - \frac{1}{8}P_8 \\ &= (\frac{1}{4}P_1 + \frac{1}{4}P_2 + \frac{1}{4}P_5) - (\frac{1}{4}P_3 + \frac{1}{4}P_6 + \frac{1}{4}P_7) \\ &= f_8 - f_7 \end{aligned} \tag{14}$$

Case 4: If G_x and G_y are both negative.

$$\begin{aligned} G_4 &= (f_2 - f_1) + (f_4 - f_3) \\ &= \frac{1}{8}P_0 + \frac{1}{4}P_1 + \cancel{\frac{1}{8}P_2} - \cancel{\frac{1}{8}P_6} - \frac{1}{4}P_1 \\ &- \frac{1}{8}P_8 + \frac{1}{8}P_0 + \frac{1}{4}P_3 + \cancel{\frac{1}{8}P_6} - \cancel{\frac{1}{8}P_2} - \frac{1}{4}P_5 - \frac{1}{8}P_8 \\ &= (\frac{1}{4}P_0 + \frac{1}{4}P_1 + \frac{1}{4}P_3) - (\frac{1}{4}P_5 + \frac{1}{4}P_7 + \frac{1}{4}P_8) \\ &= f_6 - f_5 \end{aligned} \tag{15}$$

At a particular instant of time, one out of four cases will be true and rest of the cases will be false. Suppose, if case 1 is true then f_1 and f_3 are greater than or equal to f_2 and f_4, respectively. Hence, in that case G_1 is greatest among all, which is considered as resultant gradient, G, of the center pixel (P_4) of sub-window. These four cases are exclusive of each other. In turn, it is observed that the value of G will be the maximum of four possible values,which can be represented by equation 16.

$$\begin{aligned} G &= \max(G_1, G_2, G_3, G_4) \\ &= \max(|f_5 - f_6|, |f_7 - f_8|) \\ &= \max(f_9, f_{10}) \end{aligned} \tag{16}$$

where $f_9 = |f_5 - f_6|$ and $f_{10} = |f_7 - f_8|$. Finally, equation 4 is replaced by equation 17 for deciding whether the center pixel, P_4, is an edge pixel or not.

$$D_o p = \begin{cases} 0, & \text{if } f_9 < T \text{ and } f_{10} < T \\ 1, & \text{otherwise} \end{cases} \tag{17}$$

4 Methodology

Let image I is stored as a rectangular array of elements, 'A' with m rows and n columns. As memory of the computer is simply a sequence of addressed locations, the programming language stores this two-dimensional array, 'A', in memory by a block of $m \times n$ sequential memory locations either row by row in row-major order or column by column in column-major order. The order of elements and their adresses change with the above specified techniques. This can further be illustrated with an example.
Consider an image represented as array 'A' shown in figure 3. The computer does not keep track the address of every element of a linear array. Rather, it keeps track the address of the first element A(0,0), which is known as base address, B and computes the address of a particular element, A(i,j), as:

$$Rowmajor : loc(A(i,j)) = B + n \times i + j \tag{18}$$

	0	1	2	3	4	...	n-1
0	8	5	6	7	12	...	14
1	9	11	3	21	10	...	1
2	14	23	2	33	30	...	4
⋮	...	...	...	...	...	...	...
m-1	15	45	54	52	40	...	50

Fig. 3 An image I stored as two-dimensional array, 'A', with m rows and n columns

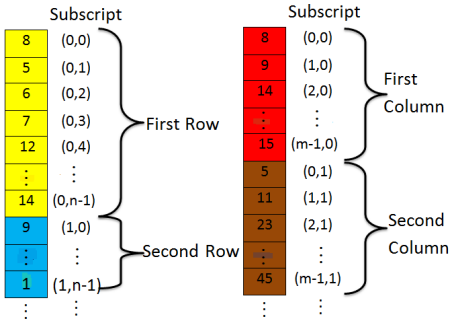

Fig. 4 Row major and Column major order storage of 'A'(from left to right)

or the equation

$$Columnmajor : loc(A(i,j)) = B + i + m \times j \tag{19}$$

Figure 4 shows the ways a two dimensional array, $[A]_{m\times n}$, are represented in memory by equation 18 and 19.

Figure 5 demonstrates the convolution process. It is clear from the figure that a predefined mask of size 3×3 has to be moved from top-left to bottom-right corner of an image. In each iteration, the specified mask has to be moved one position right and one position down when columns exhaust in a particular row. So, 9 pixels are involved in each iteration during convolution. Authors in [8], considered 9 pixels of a 3×3 sub-window as one group of an image and stored them using equation 18 for each iteration. It means that one group of 9 pixels is stored in a column for the first iteration. Again one group of 9 pixels is stored at the end of the above column in the second iteration. This process has to be continued till bottom-right corner of an image. Figure 6 shows the way how a 3×3 sub-window of an image can be stored in a linear array using equation 18.

If the above mentioned strategy is used to store the pixel information of an image, then no algorithm is required to extract the 9 pixels for each iteration during convolution. It only extracts the 9 pixels group-wise sequentially for

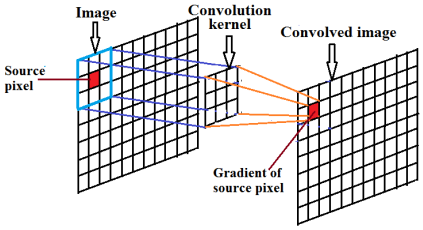

Fig. 5 Scanning of an image using mask

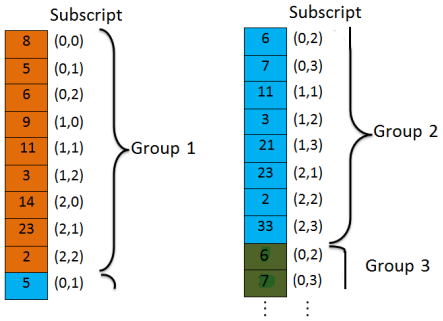

Fig. 6 Group-wise storage of pixels for convolution

each iteration. This strategy is expensive in terms of space because information of a particular pixel is repeated. As seen in figure 6 there are multiple storage of same pixels, for e.g., pixel at location (0,1) with value 5 has appeared twice. On the other hand, row-major/column-major order takes less space to store the pixel information of an image in a linear array. At most $m \times n$ entries will be there in this linear array if a two-dimensional array has m rows and n columns. But, it requires an efficient FPGA based hardware architecture to extract 9 pixels in each iteration during convolution process. After fetching, 9 pixels are convolved with mask.
The whole work is divided into two parts, of which the first part deals with the extraction of indices of the pixels of the sub-window from the image, which are convolved with mask and the second part deals with the simplification of the circuitry used for detecting the edges.

4.1 Architecture for extraction of indices of sub-window pixels

Figure 7 shows a 3×3 random sub-window with relative indices, which will be convolved with Sobel mask in each iteration during convolution. Algorithm 1 is used to extract the indices of the 9 pixels, which helps to fetch the intensity values of the extracted locations by $A(l_t)$ where, t varies from 0 to 8. Then these intensity values are convolved with predefined masks for finding out gradient.

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i, j-1)	(i, j)	(i, j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)

Fig. 7 A random 3×3 sub-window with relative indices

Algorithm 1 Extraction of Indices

```
1: procedure iExtract()
2: //Variable 'i' represents row number whereas j represents column number
3: //Variable 'B' is the base address of the array
4: //Variable 'n' is the total number of the column of an image
5:
6:     for i = 1 to n − 2 do
7:         Set k = B + i ∗ n
8:         for j = 1 to n − 2 do
9:             l0= k − n + j − 1
10:            l1= k − n + j
11:            l2= k − n + (j + 1)
12:            l3= k + (j − 1)
13:            l4= k + j
14:            l5= k + (j + 1)
15:            l6= k + n + (j − 1)
16:            l7= k + n + j
17:            l8= k + n + (j + 1)
18:        end for
19:    end for
20: end procedure
```

It is clear from the algorithm that there are three operations namely, subtraction, addition and multiplication, involved for finding out indices, to fetch the intensity values for the convolution process. A FPGA based hardware architecture is proposed based on the mathematical operations involved in algorithm 1. Figure 8 shows the hardware circuit for finding out the indices. Adder/subtractor composite block is sufficient for the above specified operations.

4.2 Modified architecture for Sobel Algorithm

The extracted intensity values $P_t = A(l_t)$ are fed into the next level of FPGA based architecture as inputs for the convolution process. It is clear from section 3 that subtraction, addition, division, absolute value and comparison need to be performed for detecting the edge pixel of an image. For subtraction and addition 8-bit adder-subtractor composite block is used. It has also been ob-

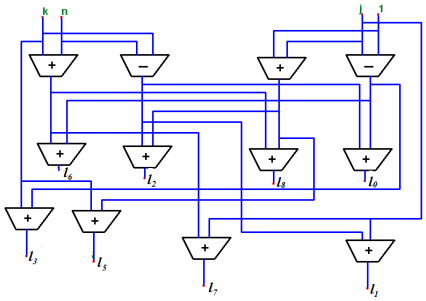

Fig. 8 A FPGA based architecture for extraction of indices

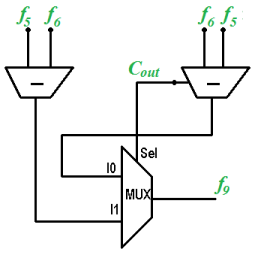

Fig. 9 Architecture for finding absolute values

served that each intensity value for the convolution process has to be divided by 4. Normally, divider is used for division operation. But, right shift operator is used here twice to divided each intensity value by 4 because divider takes more time compare to right shift operator. A 2:1 multiplexer is used for finding absolute values of $|f_5 - f_6|$ and $|f_7 - f_8|$ as shown in equation 16 . To find the absolute value the subtraction is done in two ways viz. by subtracting subtrahend from minuend and by swapping the positions of subtrahend and minuend as in figure 9. The carry bit of the subtractor block when turns out to be concludes the result as negative. This carry bit is used as the select line of the multiplexer. If carry bit is 1, then $f_9 = f_5 - f_6$, otherwise $f_9 = f_6 - f_5$. Same circuitry is used to find value of f_{10}.

After finding the absolute values, the values of f_9 and f_{10} are compared with predefined threshold (T) as shown in equation 17 for deciding whether a pixel of an image is an edge pixel or not. Generally, a comparator is used to compare two values but, complexity of a comparator is more. So, instead of a comparator, two subtractors and a NAND gate are used for the comparison which reduces the complexity. The carry bits of both the subtractors as shown in figure 10 are fed into a NAND gate to take the decision for edge pixel. The truth table of NAND gate suggest that when both the inputs are 1 then only output is 0 otherwise 1. In other words, when f_9 and f_{10} are less than the predefined threshold, there will be absence of edge pixel otherwise the corresponding pixel is considered to be an edge pixel.

The final architecture is represented in figure 11, which takes input directly

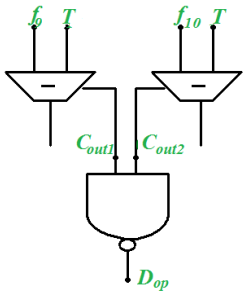

Fig. 10 Architecture for comparison with Threshold

from the addressing circuit and computes the edge pixel by comparing gradient with the threshold. Different components for the convolution process are depicted above. Figure 11 shows the complete FPGA based architecture for Sobel edge detection algorithm.

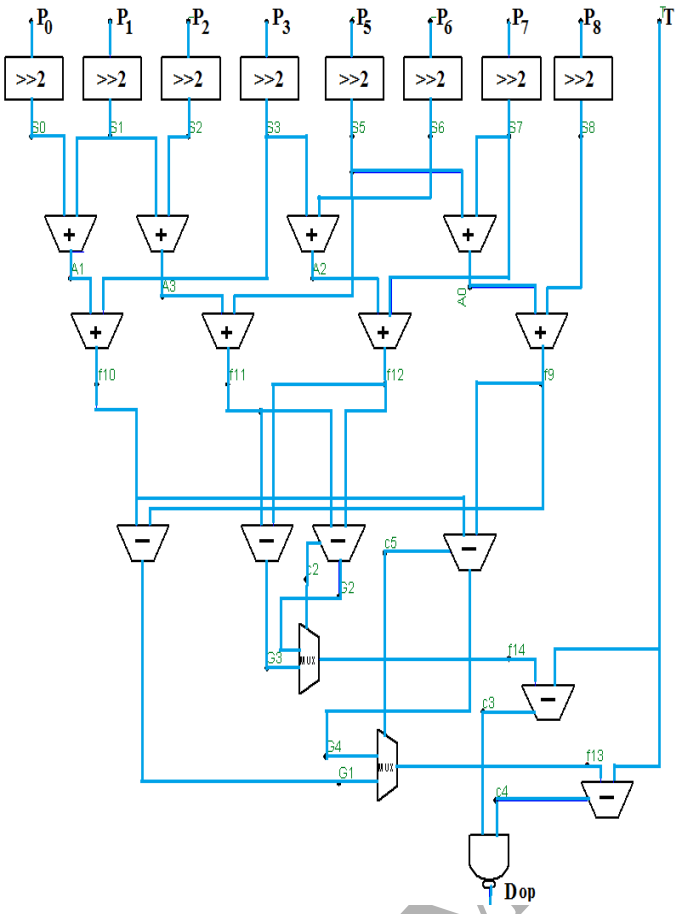

Fig. 11 Proposed circuit for hardware implementation of Sobel edge detection algorithm

5 Experimental Results and Discussion

The proposed architectures are implemented using VHDL. The program is simulated followed by synthesis on Xilinx Sparta 6 XC6SLX43TQG144 FPGA board. The simulation is done on grayscale images of various sizes using a predefined threshold. Figure 12 shows 8-bit grayscale input images and their corresponding edge images by hardware and software implementations respectively. In both the cases, same threshold of intensity value 27 has been chosen.

Table 1 shows the comparitive study of device utilisation summary in case of both the architecture when synthesized on Sparta 6 XC6SLX43TQG144 board. The design is capable of operating at a frequency of 504.007 MHz. The comparisons with existing design have been made in table 2 to show the time of execution of architectures.

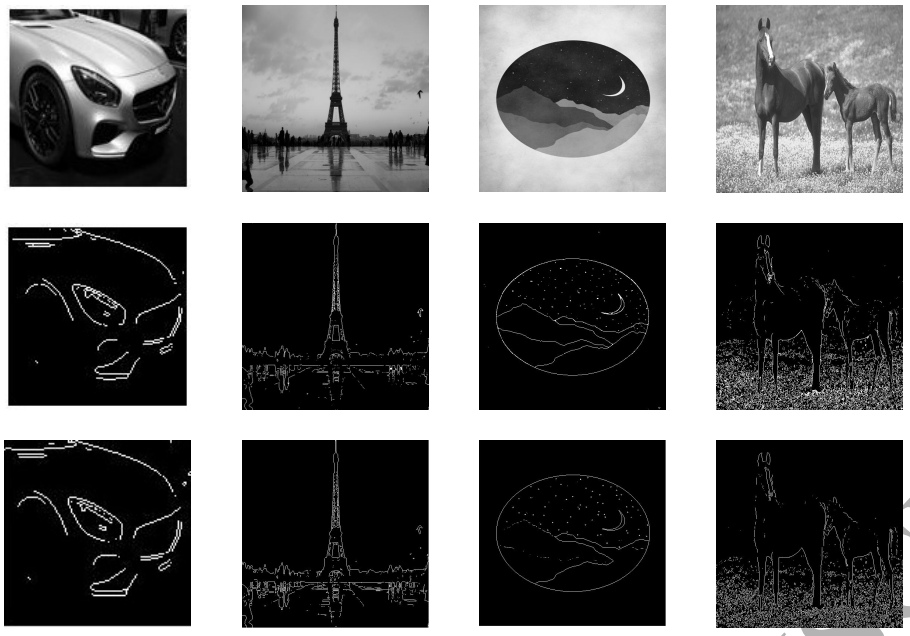

Fig. 12 Row 1: Grayscale images; Row 2: Edges by hardware implementation on FPGA; Row 3: Edge by software implementation on MATLAB.

Table 1 Device Utilisation Summary

	Total	Santanu et.al Method [8]	Proposed Method
No.of slice registers	4800	49	0
No. of slice LUTs	2400	135	114
No. of fully used LUT-FF		38	0
No. of unique control sets		1	0
No. of Bonded IOBs	102	58	57

Table 2 Timing Summary

	Maximum Frequency	Minimum Period	Time to process 128×128 image	Time to process 512×512 image
Halder et. al [8]	236.572 MHz	4.227 ns	0.07 ms	1.10 ms
Proposed Method	504.007 MHz	1.984 ns	0.032 ms	0.52 ms

6 Conclusions

The proposed architecture of Sobel edge detection uses direct indexing of desired pixels unlike the redundant storage of sub-window pixels as in the previous architectures thus reducing space complexity. The circuit is able to perform at a faster frequency than existing designs. Hence, the time to process an image is less comparatively. When this architecture is used for large databases it will show a significant difference. The time complexity of the proposed algo-

rithm is less than previously existing algorithm for hardware implementation of Sobel edge detection. Also, in main architecture, certain significant changes have been made which improve the resource utilization and make efficient use of FPGA board. It opens up the possibility of implementation of some parallel architecture on the same board.

Acknowledgments Ayan Seal is thankful to Deity, Government of India for providing him Young Faculty Research Fellowship under Visvesvaraya PhD Scheme for Electronics and IT.

References

1. Tanvir A Abbasi, Mohd Abbasi, et al. A proposed fpga based architecture for sobel edge detection operator. *Journal of Active & Passive Electronic Devices*, 2(4), 2007.
2. Fahad M Alzahrani and Tom Chen. A real-time edge detector: algorithm and vlsi architecture. *Real-Time Imaging*, 3(5):363–378, 1997.
3. M Boo, E Antelo, and JD Bruguera. Vlsi implementation of an edge detector based on sobel operator. In *EUROMICRO 94. System Architecture and Integration. Proceedings of the 20th EUROMICRO Conference.*, pages 506–512. IEEE, 1994.
4. Stephen Brown and Jonathan Rose. Architecture of fpgas and cplds: A tutorial. *IEEE Design and Test of Computers*, 13(2):42–57, 1996.
5. Stephen D Brown, Robert J Francis, Jonathan Rose, and Zvonko G Vranesic. *Field-programmable gate arrays*, volume 180. Springer Science & Business Media, 2012.
6. Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.
7. Rafael C Gonzalez, Richard E Woods, and Steven L Eddins. Digital image processing using matlab. 2004.
8. Santanu Halder, Debotosh Bhattacharjee, Mita Nasipuri, and Dipak Kumar Basu. A fast fpga based architecture for sobel edge detection. In *Progress in VLSI Design and Test*, pages 300–306. Springer, 2012.
9. Douglas L Perry. *VHDL: programming by example*, volume 4. McGraw-Hill, 2002.
10. James D Plummer. *Silicon VLSI technology: fundamentals, practice, and modeling*. Pearson Education India, 2009.
11. John C Russ. *The image processing handbook*. CRC press, 2015.
12. Alfredo Gardel Vicente, Ignacio Bravo Munoz, Pedro Jiménez Molina, and José Luis Lázaro Galilea. Embedded vision modules for tracking and counting people. *Instrumentation and Measurement, IEEE Transactions on*, 58(9):3004–3011, 2009.