Research Article

Comprehensive Review and Comparative Analysis of Hardware Architectures for Sobel Edge Detector

**Sanjay Singh,¹ Sumeet Saurav,² Ravi Saini,¹ Anil Kumar Saini,¹
Chandra Shekhar,¹ and Anil Vohra³**

¹ IC Design Group, CSIR-Central Electronics Engineering Research Institute (CSIR-CEERI), Pilani, Rajasthan 333031, India

² CSIR-CEERI, Academy of Scientific & Innovative Research (AcSIR), Pilani, Rajasthan 333031, India

³ Electronic Science Department, Kurukshetra University, Kurukshetra, Haryana 136119, India

Correspondence should be addressed to Sanjay Singh; sanjay.csirceeri@gmail.com

Received 30 January 2014; Accepted 20 February 2014; Published 6 April 2014

Academic Editors: T. Laopoulos and X. Yang

Copyright © 2014 Sanjay Singh et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a comprehensive review and a comparative study of various hardware/FPGA implementations of Sobel edge detector and explored different architectures for Sobel gradient computation unit in order to show the various trade-offs involved in choosing one over another. The different architectures using pipelining and/or parallelism (key methodologies for improving the performance/frame rates) are explored for gradient computation unit in Sobel edge detector. How the different architectures affected performance (in terms of video frame rate and image size) and area (in terms of FPGA resources usages) has been demonstrated. By exploiting the trade-offs between video frame rate, image size, and FPGA resources a designer should be able to find an optimal architecture for a given application.

1. Introduction

Edge detection, one of the fundamental and most important problems of lower level image processing, plays a very important role in the realization of a complete vision based understanding/monitoring system for automatic scene analysis/monitoring [1]. Edges provide significant and important information related to objects present in the scene. This information helps in achieving higher level objectives like segmentation, object recognition, scene analysis, and so forth.

Edges in digital images are defined as the image positions/points where the intensity/brightness of two neighboring pixels is significantly different. Many robust and complex approaches for the edge detection have been proposed in scientific literature. These give different responses and details for same input images. Sobel operator based edge detection technique is very popular and intensively used in many applications due to its ability to counteract the noise sensitivity over simple gradient operators and its easier implementations [2].

Very different approaches have been used in the literature for Sobel operator based edge detection algorithm. These range from use of general purpose processors or special purpose digital signal processors or graphics processing units (GPUs) using compute unified device architecture (CUDA) to application specific integrated circuits (ASICs) or applications specific instruction set processors (ASIPs) or even programmable logic devices like field programmable gate arrays (FPGAs). FPGAs provide real-time performance, limit the extensive design work and time required for ASICs, and provide possibility to perform algorithmic changes in later stages of system development. These features make FPGAs a suitable choice for implementing image processing algorithms (in particular Sobel operator based edge detection scheme). Therefore, design of application specific optimized hardware/VLSI architecture for Sobel operator based edge detection and its FPGA implementation is a very crucial research issue.

There are three important metrics for hardware/VLSI architecture when it comes to image processing application (in this case Sobel edge detector): video frame rate, frame

size, and area. In this paper, a comprehensive review of existing hardware implementations (ASICs/FPGAs) of Sobel edge detector has been presented. In addition to this, different architectures using pipelining and parallelism are explored for gradient computation unit of Sobel edge detector which involves various trade-offs. We have demonstrated how the different architectures affected performance (in terms of video frame rate and image size) and area (in terms of FPGA resources usages). By exploiting the trade-offs between video frame rate, image size, and FPGA resources a designer should be able to find an optimal architecture for a given application.

2. Sobel Edge Detection Algorithm

In this section the used algorithm is briefly described; for a more detailed description we refer to [2–4]. The Sobel operator is widely used for edge detection in images. It is based on computing an approximation of the gradient of the image intensity function. The Sobel filter uses two 3×3 spatial masks which are convolved with the original image to calculate the approximations of the gradient. The Sobel operator uses two filters H_x and H_y :

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad (1)$$

$$H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

These filters compute the gradient components across the neighboring lines or columns, respectively. The smoothing is performed over three lines or columns before computing the respective gradients. In this Sobel operator, the higher weights are assigned in smoothing part to current center line and column as compared to simple gradient operators.

Consider the digital image shown in Figure 1. The solid black color boundary shows the current computing window containing eight neighborhood pixels and current pixel P_{22} . The dashed line boundary shows the computing window movement in the vertical direction and dotted line boundary shows the computing window movement in the horizontal direction. This computing window is moved over the entire image for each pixel in order to compute edge map for the whole image. The absolute values of gradient H_x and H_y for computing edge map for pixel data P_{22} are given by the following expressions:

$$|H_x| = |(P_{13} - P_{11}) + (2 * (P_{23} - P_{21})) + (P_{33} - P_{31})|, \quad (2)$$

$$|H_y| = |(P_{31} - P_{11}) + (2 * (P_{32} - P_{12})) + (P_{33} - P_{13})|.$$

We can combine these two equations into one and rewrite them as

$$|H_i| = |(A_1 - B_1) + (2 * (A_2 - B_2)) + (A_3 - B_3)|, \quad (3)$$

P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	...	P_{1N}
P_{21}	P_{22}	P_{23}	P_{24}	P_{25}	...	P_{2N}
P_{31}	P_{32}	P_{33}	P_{34}	P_{35}	...	P_{3N}
P_{41}	P_{42}	P_{43}	P_{44}	P_{45}	...	P_{4N}
P_{51}	P_{52}	P_{53}	P_{54}	P_{55}	...	P_{5N}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
P_{M1}	P_{M2}	P_{M3}	P_{M4}	P_{M5}	...	P_{MN}

FIGURE 1: Sobel gradient computation flow for input image.

where

$$\{A_1, B_1, A_2, B_2, A_3, B_3\}$$

$$= \begin{cases} \{P_{13}, P_{11}, P_{23}, P_{21}, P_{33}, P_{31}\} & \text{for } i = x \\ \{P_{31}, P_{11}, P_{32}, P_{12}, P_{33}, P_{13}\} & \text{for } i = y. \end{cases} \quad (4)$$

The local edge strength is defined as the gradient magnitude given by

$$GM(x, y) = \sqrt{H_x^2 + H_y^2}. \quad (5)$$

This equation is computationally costly because of square and square root operations for every pixel. It is more suitable computationally to approximate the square and square root operations by absolute values:

$$GM(x, y) = |H_x| + |H_y|. \quad (6)$$

This expression is much easy to compute and still preserves the relative changes in intensity (edges in images).

3. Literature Review

The various VLSI architectures and FPGA based implementations are presented in the literature for Sobel edge detection. These are designed using different design methodologies. These work at different operating frequencies and occupy different number of FPGA resources. Also, these provide different frame rates for different video/image size. A comprehensive study is performed for the existing literature.

The available literatures on FPGA based edge detection differ from each other due to design methodologies/approaches, design tool chain used, and algorithmic improvements considered for achieving enhanced accuracy. Based on the design methodologies there are five different categories: general purpose processor based approach, digital signal processors (DSPs) based approach [5], application specific integrated circuit (ASIC) approach, FPGA based hardware design approach, or FPGA based hardware/software

codesign approach. Using design tool chain approach, the differences can be based on the use of VHDL/Verilog, high level hardware description language like Handle-C or SystemC, MATLAB-Simulink software, or embedded development kit and System Generator tool. Another difference is based on algorithmic improvements for enhanced accuracy.

First ASIC chip for Sobel operator based edge detection was designed by [6]. The chip architecture is highly pipelined in performing the computations of gradient magnitude and direction (angle) for the output image samples. Motivated to increase the performance of the edge detector [7] proposed a novel ASIC VLSI architecture for robust Sobel edge detection. In this work the authors demonstrated the power of the cooperating data-path model for medium level image processing applications. The result reported by the work out-performs the then existing realizations. The design was aimed for 512×512 size image but could be easily adapted to the changing specifications, such as the image size, or the maximum and minimum thresholds to the edges which affects the robustness. The unique feature of the reported edge detector is that it performs adaptive thresholding and produces a single pixel wider edge and offers the edge information (location and orientation) in real time. With the further advancement of technology systolic processor arrays based architectures were started being employed for the implementation of edge detectors. In [8], the authors used the systolic array processor for the implementation of the Sobel operator in an application specific integrated circuits for an efficient exploitation of the advantage of VLSI technology, that is, exploiting as much as possible its parallelism and pipelining. The resulting chip provided the value for the pixels of the gradient images (rows and columns), alternatively each clock cycle, with a latency of 20 clock cycles. The maximum operating frequency achieved was 50 MHz which proved to be an adequate design for real-time image processing.

To increase the clock frequency the research communities started looking for alternatives and then the age of FPGAs came. The introduction of FPGAs revolutionized the research activity related with the edge detection in the computer vision community. Current generation FPGAs provide the real-time performance, which is difficult to achieve with general purpose processors (GPPs) or application specific digital signal processors (DSPs). Furthermore, FPGAs provide the possibility to perform algorithm modification in the later stages of the system development and they offer relatively low cost and reduce time to market and many other advantages. All these features of FPGAs advocate their use as a real alternative to ASICs for low level image processing applications. In [9], the authors presented a single instruction and multiple data (SIMD) architecture implemented on FPGA devices. This architecture is based on parallel processing units with internal pipeline and uses Sobel gradient operators for edge detection. The architecture takes advantage of the FPGAs capabilities for parallel processing in order to reduce the execution time needed using a sequential machine. The proposed architecture is able to segment up to 43 images of 640×480 pixels in one second with 40 MHz clock frequency. This design provided 153 times faster image edge detection

compared to the sequential schemes. However, the architecture uses multiple processing elements and therefore requires more FPGA resources. The architecture clock frequency can be improved by using pipelining. A pipelined architecture for real-time gray scale image detection is presented in [10]. The architecture has been implemented on Verilog HDL, synthesized for a XCS3S1500-5FG320 device from Xilinx Spartan 3 family, and simulated on ModelSim SE 5.8c from Mentor Graphics Corporation. The test images used were 512×512 pixels with 256 gray levels. The architecture is capable of operating at a speed of 99.499 MHz, which is much better than processing images on a software platform using high level programming languages like C or C++. The main disadvantage of this architecture is that it utilized all the BRAMs available on the FPGA board. Since edge detection is only a part of the actual image processing system, it is desired to reduce the memory utilization of the architecture. A memory efficient architecture for Sobel edge detection has been proposed in [11]. In this paper, the authors have explored different memory systems on FPGA chips in order to show the various trade-offs involved with choosing one memory system over the other. Generally there are four important metrics for hardware designs when it comes to memory: bandwidth, latency, size, and area. For a typical pipelined architecture bandwidth would be the size of the pipe or the number of the pipes. Latency can be described as the length of the pipe(s). The longer the pipe is, the longer it takes for the data to reach its destination and the slower the overall system is. There are typically three different types of memory available on FPGAs: register, block memory (RAM), and external memory (DDR). The architecture designed with the external memory has the fastest clock speed and utilizes the least amount of resources on the FPGA but has the longest execution time. The number of execution cycles can be reduced by adding more registers to the input buffer so that the pixels can be stored and reused until they are no longer needed. This is the fastest design because it required the fewest number of execution cycles with almost the same clock speed as external memory design. However, because the input buffers are composed entirely of registers, which are distributed uniformly over the FPGA, they utilized the largest area on the FPGA. The area utilized can be reduced by utilizing block RAM instead of registers in the input buffer. However, this design has longer pipeline latency. Thus, while having a slower clock speed and a larger number of execution cycles, the design utilized a much smaller area on the FPGA. Finally, a combination of block RAM and registers is utilized which has the same performance as the register design and utilizes almost the same area as the block RAM design. This design makes the best trade-off between area and performance. The architecture designed to reduce the number of calculations required for the Sobel edge detection process has been discussed in [12]. This architecture enhances the data reuse by minimizing the frequency of memory access. The processor is multiplier-free and based only on simple additions, subtractions, shift registers, and modulus operators. The synthesis results have not been provided; however it has been mentioned that the architecture reduces the subtraction operations by 50% compared to the architecture discussed

in [13]. In [14], a more optimized architecture for Sobel edge detection has been proposed. Here the optimization has been done with the motivation to minimize memory utilization, redundant calculations, and hence overall logic resources used to implement the processor on FPGA. The optimization is achieved by exploiting the FPGAs' high parallelism, flexibility, and I/O bandwidth. Results show that the optimized processor architecture uses 22% less adaptive lookup tables (ALTs), 40% less dedicated logic registers, and 10% overall logic resources utilization reduction over basic architecture [13]. The design has been implemented on Stratix II EP2S60 and achieves 50% less subtraction operation as well as 40% less RAM space which leads to 10% reduction in the total logic utilization than the reference design [13]. In order to improve the performance of the Sobel edge detector, the architecture has been discussed in [15]. The FPGA based Sobel edge detection operator was modeled using Verilog hardware description language, compiled, synthesized, and downloaded to Cyclone II Altera development board by using Quartus II 7.2 SP3 web edition. The image used was 720×720 pixels with 256 gray levels. The design was able to operate at 27 MHz clock frequency. The processor was able to detect the edges in 2 ms. In order to enhance the clock frequency and performance of the Sobel edge detector, a parallel architecture has been implemented in [16]. This architecture is based on the binary images and has been implemented in Verilog HDL and synthesized using Virtex-4, XC4LX200 device. The time taken for the Sobel operator to calculate the gradients is 400 microseconds at 200 MHz clock frequency.

The Sobel operator has the advantage of simplicity in calculation. But the accuracy is relatively low because it uses only two convolution kernels to detect the edges of image. Therefore, the orientation of the convolution kernels is increased from 2 to 4 in order to increase the accuracy of edge detection. A parallel architecture for Sobel edge detection enhancement algorithm has been discussed in [17]. The design has been implemented on Xilinx XC3S200-5 ft256 and can process one pixel in every clock cycle. Similar architecture has been discussed in [18], but the processor has been implemented on Xilinx Spartan-3 XCS3S200 FPGA chip and has been coded using VHDL hardware description language. A modified architecture of Sobel edge detection with adjustable threshold level has been discussed in [19]. The design has been realized on Xilinx Spartan-3A FPGA board.

With the introduction of reconfigurable platforms such as FPGA and advent of new high level tools to configure them, image processing on FPGA has emerged as a practical solution for most of the computer vision and image processing problems. In [20], the authors have proposed the implementation of the Sobel edge detector on FPGA using powerful design tools, system generator (SysGen), and embedded development kit (EDK) for hardware software codesign. The design integrates the edge detection hardware as a peripheral to the Micro-blaze 32-bit soft RISC processor with an input from a CMOS camera and output to a DVI display. The results have been verified to be in real time. Similar work has been reported in [21], where the Sobel edge detector has been implemented on a Spartan-3A DSP FPGA board and processes the image at a rate of 60 fps for

an input image of resolution 720×480 . The implemented system architecture has 88.547 MHz clock frequency. Based on the same methodology of hardware software codesign in [22], the authors have proposed a high performance hardware coprocessor for cellular neural networks (CNNs) applied to edge detection and its integration with OpenCV library. The parallel nature of the CNNs makes them suitable to be implemented on a reconfigurable device, such as FPGA. An FPGA implementation of CNNs achieves high performance and flexibility due to fine-grain parallelism of the FPGA based implementation. The designed processor is modeled using Handel-C language and implemented on a Vertex-II Pro FPGA board hosted on an Alpha data ADM-XPL board. The target board is interfaced with PC using the PCI bus, and the coprocessor is fully integrated with the OpenCV libraries. Handel-C is one of the most extended approaches for implementing hardware architecture starting from C-based implementations, reducing the design effort and accelerating the development cycle [23]. With more advancement of technology, there are some works related to the SoC implementation of the Sobel edge detector. In [24], the authors have described the generation of Sobel edge detection filter in the Zynq-7000 programmable SoC ZC702 base board using the Vivado high-level synthesis (HLS) tool. Following the similar methodology, in [25], the authors have presented a novel implementation of the modified Sobel edge detector using the combination of EDK and Matlab environments. The processor has been implemented on Virtex-5 ML506 board and claims to be memory efficient with better edge detection in the noisy environment. A similar implementation can also be found in [26], where authors have implemented an efficient video edge detection system using Sobel operator. An alternative hardware-software combination of EDK with SystemC for Sobel edge detection has been discussed in [27]. Sobel edge detection implementation on graphics processing units (GPUs) using compute unified device architecture (CUDA) and OpenGL has been presented in [28].

4. Sobel Edge Detection Architecture

The basic dataflow block level diagram for detecting edges in a video stream coming from camera is shown in Figure 2. The camera interface module decodes the incoming video stream from the camera and performs the color space conversion (YCrCb to RGB). Finally, it converts 24-bit RGB data into 8-bit gray data. DVI module uses edge information (output of edge detection module) and video timing signals information (from camera interface) to display the edge detected video stream on display monitor. For Sobel edge detector, there are three main modules: Sobel buffer memory, gradient computation module, and edge map module. Sobel edge detector is a window based operator which requires pixel neighborhood information for computing the edge map of a particular pixel. Therefore, 8-bit gray pixel data coming from the camera interface module cannot be processed directly. It must be stored in FPGA memory before processing. The gradient computation module uses eight neighborhood pixels coming from buffer memory for computing the approximate gradient

FIGURE 2: Block diagram of Sobel edge detector.

FIGURE 3: Smart buffer based buffer memory architecture for Sobel edge detector.

value which is the sum of absolute values of horizontal and vertical gradients. Edge map module is a simple comparator which compares the gradient value (GRD) with user defined threshold (TH).

It is observed that for the Sobel edge detector there are two main modules: Sobel buffer memory and gradient computation unit. The comparative study and exploration of different memory systems on FPGA and different VLSI architectures for gradient computation unit are required for making a proper choice of the Sobel edge detection hardware architecture which involves various trade-offs. The different memory components available on FPGA boards are explored in [11], for Sobel edge detection. They showed the various trade-offs (I/O performance and area) involved in choosing one memory system over another. They found that a combination of registers and block memory worked best for a Sobel edge detector because it makes the best trade-off between area and performance. It used a type of smart buffer that shifts values into the sliding window in every clock cycle. Figure 3 shows the architecture of smart buffer based memory architecture for Sobel edge detector.

Research and exploration on different architectures for gradient computation unit are not yet presented in the literature and are very crucial and of high importance for making proper selection of architecture for a particular application.

5. Gradient Computation Unit Architectures

This section explores the various possible hardware architectures for gradient computation unit.

5.1. Architecture I (Using Single PE). This first implementation is a sequential architecture (Figure 4). This is based on hardware realization of expression (3). The sequential architectures for Sobel compass edge detector and Sobel edge detector were explored by Sanjay et al. in [29, 30]. In sequential Sobel edge detector architecture [30], gradient computation for both horizontal and vertical directions is realized through single gradient computation module. The module is used in appropriate sequential order in different time slots for computing both horizontal and vertical gradients. This architecture greatly economizes on the FPGA resources usages (area) for approximate gradient computation but needs storage elements to store results for future use and set of multiplexers for switching of inputs and outputs in different time slots. This sequential architecture also requires a controller which insures proper functioning of complete design. The maximum possible frame rate for this implementation is the lowest among all explored architectures but meets the real-time requirements of surveillance video applications.

5.2. Architecture II (Using Single Pipelined PE). The gradient computation unit module in above-mentioned architecture (Figure 4) is purely combinational which uses many adders/subtractors. These are actively used only for a fraction of the total time required for computing one gradient component. This is an inefficient way of using the resources. For efficient utilization and improvement in the throughput of combinational processing module, the most common technique is to insert storage elements between two successive operations. The storage elements are called pipeline registers

FIGURE 4: Sobel gradient computation using single processing element.

FIGURE 5: Sobel gradient computation using single fully pipelined processing element.

and the architecture is called pipelined architecture. This will improve the throughput but does not speed up the generation of outputs for any set of inputs. The pipelined architecture is shown in Figure 5. There are a total of four pipeline stages (processing module followed by pipeline register). Therefore, there is significant improvement in the performance at the cost of area occupied by the pipeline registers.

5.3. Architecture III (Using Two PEs). An alternative approach to pipelining for improving throughput is parallelism. The two combinational gradient computation units are working in parallel on different data items to produce horizontal and vertical gradient components simultaneously (Figure 6). This implementation is direct VLSI implementation of expression (2). This parallel architecture is possible for gradient computation because the different data items needed for horizontal and vertical gradient computations are available simultaneously. This is the most obvious and standard implementation. Most of the existing literatures used this implementation. It can process the video at a frame rate twice the first architecture but can occupy larger FPGA resources.

5.4. Architecture IV (Using Two Pipelined PEs). The final architecture (Figure 7) uses both the ideas of pipelining and parallelism for achieving very high frame rates for high resolution video streams. In this design, the gradient computation unit is pipelined and at the same time uses two pipelined gradient computation units in parallel. The design performs at very high frame rates but occupies large FPGA resources. The previous pipelined implementation of Sobel edge detector was presented by [10]. The implementation presented in Figure 7 is significantly optimized pipelined implementation as compared to existing work for two reasons. Firstly, in [10], six pipeline stages are used, while in this implementation only four pipeline stages are used and the achieved frame rates are much higher. Secondly, in [10], the authors have inserted pipeline registers for achieving divide by 4 and divide by 2 operations. In hardware, divide by 2 and divide by 4 operations are achieved by right shift operations. So we can simply leave/drop 2 LSBs for divide by 4 and 1 LSB for divide by 2 and append corresponding 0's at MSB positions. Therefore, these operations do not require any arithmetic operations. Therefore, insertions of pipeline registers for divide by 4 and divide by 2 operations only

FIGURE 6: Sobel gradient computation using two processing elements in parallel.

FIGURE 7: Sobel gradient computation using two fully pipelined processing elements in parallel.

increase the FPGA resources (area) without any significant improvement in clock frequency (throughput). Moreover, in first pipeline stage, the authors used addition of three numbers, which will use two adders in series and degrades the system performance. All these issues are taken care of in this implementation.

6. Results and Discussions

All the four above mentioned architectures are coded in VHDL and simulated using ModelSim. Synthesis is carried out using Xilinx ISE tool chain (version 10.3.1). A complete real-time working prototype system has been developed using Xilinx ML510 (Virtex-5 FXT) FPGA platform. It consists of a camera (Sony EVI D-70P) interfaced with FPGA platform using high speed I/Os of FPGA platform and display monitor connected using DVI port of FPGA platform. Table 1 demonstrates the FPGA resources utilized (after place and route results) by each implemented architecture of the gradient computation unit. Table 2 shows the maximum operating frequency and maximum possible frame rates (for various image sizes) for all four implemented architectures. The real-time situations captured by camera and processed

TABLE 1: FPGA resource utilization comparison.

Architecture	Logic Slices	LUTs	Registers	LUT FF Pairs	Route-thrus
I	24	83	23	83	2
II	31	96	77	97	0
III	42	128	1	128	8
IV	53	137	141	172	1

edge detected images produced by each architecture are shown in Figure 8. The test images are of PAL size (720 × 576). All four architectures produce the same edge detected output images for a given input image. It is verified by subtracting the output images produced by different architectures and the result was zero matrix/image insuring that outputs are identical.

7. Conclusions

A comparative study of different hardware implementations of the Sobel edge detector has been presented. Different architectures using pipelining and parallelism have been explored

FIGURE 8: Input test images taken from the camera and output edge detected images produced by architectures I, II, III, and IV, respectively.

TABLE 2: Performance comparison.

Architecture	Frequency (MHz)	Frame rate for PAL video 720×576	Frame rate for CIF video 352×288
I	137.339 MHz	165 fps	667 fps
II	284.297 MHz	342 fps	1402 fps
III	117.696 MHz	283 fps	1160 fps
IV	350.214 MHz	844 fps	3454 fps

for gradient computation unit of Sobel edge detector. We have demonstrated how the different architectures affected performance (in terms of video frame rate and image size) and area (in terms of FPGA resources usages) of image edge detection system. By exploiting the trade-offs between video frame rate, image size, and FPGA resources a designer should be able to find an optimal architecture for a given application. Sequential architecture is the best choice for area constrained real-time video surveillance application, while a combination of pipelining and parallelism is most suited for very high frame rate applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was carried out under a project supported by Department of Electronics & Information Technology (DeitY), Ministry of Communications and Information Technology (MCIT), Government of India. Sanjay Singh would like to thank Dr. AS Mandal and Mr. Raj Singh, Group Leader, IC Design Group, for their constant guidance, suggestions, support, and encouragement.

References

- [1] M. B. Ahmad and T.-S. Choi, "Local threshold and boolean function based edge detection," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 674–679, 1999.
- [2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Education, New Delhi, India, 2009.
- [3] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, Morgan Kaufmann, San Francisco, Calif, USA, 2004.
- [4] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, New York, NY, USA, 1995.
- [5] K. N. Ngan, A. A. Kassim, and H. S. Singh, "Parallel image-processing system based on the tms32010 digital signal processor," *IEE Proceedings E: Computers and Digital Techniques*, vol. 134, no. 2, pp. 119–124, 1987.
- [6] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988.
- [7] C.-Y. Lee, F. V. M. Catthoor, and H. J. de Man, "Efficient ASIC architecture for real-time edge detection," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1350–1359, 1989.
- [8] M. Boo, E. Antelo, and J. D. Bruguera, "VLSI implementation of an edge detector based on sobel operator," in *Proceedings of the 20th EUROMICRO Conference on System Architecture and Integration*, pp. 506–512, 1994.
- [9] R. L. Rosas, A. de Luca, and F. B. Santillan, "SIMD architecture for image segmentation using sobel operators implemented in FPGA technology," in *Proceedings of the 2nd International Conference on Electrical and Electronics Engineering (CIE '05)*, pp. 77–80, September 2005.
- [10] T. A. Abbasi and M. U. Abbasi, "A novel FPGA-based architecture for Sobel edge detection operator," *International Journal of Electronics*, vol. 94, no. 9, pp. 889–896, 2007.
- [11] C. Moore, H. Devos, and D. Stroobandt, "Optimizing the FPGA memory design for a Sobel edge detector," in *Proceedings of 20th Annual workshop on Circuits, Systems and Signal Processing*, 2009.
- [12] Z. E. M. Osman, F. A. Hussin, and N. B. Z. Ali, "Optimization of processor architecture for image edge detection filter," in *Proceedings of the 12th UKSim International Conference on Modelling and Simulation (UKSim '10)*, pp. 648–652, March 2010.

- [13] V. Thomas, A. Emmanuel, A. Prabhakar, M. Thomas Mathew, S. John, and V. Kumar, *Hardware Image Edge Detection Project on FPGA Using Sobel Operator*, 2009.
- [14] Z. E. M. Osman, F. A. Hussin, and N. B. Z. Ali, "Hardware implementation of an optimized processor architecture for SOBEL image edge detection operator," in *Proceedings of the International Conference on Intelligent and Advanced Systems (ICIAS '10)*, pp. 1–4, June 2010.
- [15] I. Yasri, N. H. Hamid, and V. V. Yap, "Performance analysis of FPGA based Sobel edge detection operator," in *Proceedings of the International Conference on Electronic Design (ICED '08)*, pp. 1–4, December 2008.
- [16] V. Sanduja and R. Patial, "Sobel edge detection using parallel architecture based on FPGA," *International Journal of Applied Information Systems*, vol. 3, no. 4, pp. 20–24, 2012.
- [17] Z. Guo, W. Xu, and Z. Chai, "Image edge detection based on FPGA," in *Proceedings of the 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES '10)*, pp. 169–171, August 2010.
- [18] A. Nosrat and Y. S. Kavian, "Hardware description of multi-directional fast sobel edge detection processor by VHDL for implementing on FPGA," *International Journal of Computer Applications*, vol. 47, no. 25, 2012.
- [19] A. R. Ibrahim, N. A. Wahed, N. Shinwari, and M. A. Nasser, "Hardware implementation of real time video edge detection with adjustable threshold level (edge sharpness) using Xilinx Spartan-3A FPGA," Tech. Rep., 2011.
- [20] K. C. Sudeep and J. Majumdar, "A novel architecture for real time implementation of edge detectors on FPGA," *International Journal of Computer Science Issues*, vol. 8, no. 1, pp. 193–202, 2011.
- [21] Y. Said, T. Saidani, F. Smach, M. Atri, and H. Snoussi, "Embedded real-time video processing system on FPGA," in *Proceedings of International Conference on Image and Signal Processing*, pp. 85–92, Springer, 2012.
- [22] M. A. Nuño-Maganda, M. Morales-Sandoval, and C. Torres-Huitzil, "A hardware coprocessor integrated with OpenCV for edge detection using cellular neural networks," in *Proceedings of International 6th International Conference on Image and Graphics (ICIG '11)*, pp. 957–962, August 2011.
- [23] Celoxica, "Handel-C Language Reference Manual," 2003.
- [24] F. M. Vallina, C. Kohn, and P. Joshi, "Zynq all programmable SoC Sobel filter implementation using the Vivado HLS tool," Application Note XAPP890, Xilinx, 2012.
- [25] A. M. Khidhir and N. Y. Abdullah, "FPGA based edge detection using modified sobel filter," *International Journal For Research and Development in Engineering*, vol. 2, no. 1, pp. 22–32, 2013.
- [26] J. Wu, J. Sun, and W. Liu, "Design and implementation of video image edge detection system based on FPGA," in *Proceedings of the 3rd International Congress on Image and Signal Processing (CISP '10)*, pp. 472–476, October 2010.
- [27] N. Prathyusha and A. Balaji Nehru, "A high speed ASIC design for sobel edge detection using FPGA," *International Journal of New Trends in Electronics and Communication*, vol. 1, no. 4, pp. 28–32, 2013.
- [28] K. Shah, "Performance analysis of sobel edge detection filter on GPU using CUDA & OpenGL," *International Journal For Research in Applied Sciences and Engineering Technology*, vol. 1, no. 3, pp. 22–26, 2013.
- [29] S. Singh, A. K. Saini, R. Saini, A. S. Mandal, C. Shekhar, and A. Vohra, "Area optimized FPGA based implementation of sobel compass edge detector," *ISRN Machine Vision*, vol. 2013, Article ID 820216, 6 pages, 2013.
- [30] S. Singh, A. K. Saini, R. Saini, A. S. Mandal, C. Shekhar, and A. Vohra, "A novel real-time resource efficient implementation of sobel operator based edge detection on FPGA," *International Journal of Electronics*, 2014.