FPGA implementation of sobel edge detection algorithm

K. Navinkumar[1], R. Logesh[2], P. VishnuBabu[3], A.V. Ananthalakshmi[4],

[1,2,3]Department of ECE, Puducherry Technological University, Puducherry, India.

Abstract

Sobel Edge detection algorithm is used to extract the edges (region of maximum variation) from an image. It is based on the concept that the edges of an image contains maximum information whose computation depends on multipliers and square root. As multipliers consume more logic, a modified sobel edge detection algorithm which does not employ multipliers and square root function is proposed. A mathematical model of the proposed sobel edge algorithm was first developed and MATLAB was used to verify the model. On comparing with the original model, the proposed model has a SSIM of 96.43%. To analyse the hardware complexity, Verilog model of the modified sobel edge detection algorithm was developed using Quartus II. The chosen evaluation board is Cylone III FPGA EP3C120F780. The performance metrics such has Logic Elements utilization, Power dissipation and Maximum Operating Frequency were obtained. Open-Source toolchain (Yosys, OpenVPR, and Google Skywater 130nm PDK) was used to obtain the RTL Netlist and Synthesis reports. Verilog Modules for the Camera (CMOS OV7670) interface and FIFO Buffer were synthesized. The modified algorithm was integrated with them. An HSMC (HSMB) breakout board was connected to the FPGA Development board to increase the number of I/O ports. Thus in real time, the proposed modified Sobel Edge detection system can be used as a pre-processor to reduce the amount of computations and power consumption.

Keywords: Sobel Algorithm, Edge Detection, Matlab, FPGA

Received on 3 May 2024, accepted on 2 September 2024, published on 23 October 2024

doi: 10.4108/eetiot.5148

*Corresponding author. Email: phd.kiranaswal@gmail.com

1. Introduction

Edge in an image contains the number of important parameters which is widely used in image analysis. Thus, edge detection acts as a pre-processing in the analysis of an image, image segmentation and in image feature extraction. Some of the widely employed edge detection algorithms are Robert, Prewit, LOG, Canny, Sobel and other algorithms all belong to spatial detection [1]. Of the above mentioned, Robert and Prewit has low edge positioning accuracy. LOG operators cannot identify the direction of edges and are sensitive to noise. Canny operators have superior functions but are complex to implement [2]. It is difficult to use them in realtime hardware systems. Although traditional Sobel algorithms need to manually specify detection thresholds, they have the advantages of simple detection principle and easy hardware implementation.

The Sobel edge detection algorithm is a method for detecting edges in images by applying a set of convolution filters to the image data. It is widely used in image processing and computer vision applications, particularly for detecting edges in grayscale images. It is relatively simple to implement and produces good results, but it can be sensitive to noise and may produce some false edges. It can be implemented on both a GPU (graphics processing unit) and an FPGA. Both implementations can provide significant performance improvements over a CPU-based implementation, particularly for large or complex images. Ultimately, the choice between a GPU and an FPGA for the Sobel edge detection algorithm will depend on the specific requirements of the application. A GPU may be more suitable for tasks that require high-performance graphics processing or that can be parallelized, while an FPGA may be more suitable for tasks that require custom hardware acceleration or real-time processing. Thus the proposed modified sobel edge algorithm will be implemented using FPGA.

The organization of the paper is as follows:- Section 2 elaborates on the literature survey, section 3 discusses the methods employed in the proposed work. Section 4 discusses the results and finally section 5 concludes the paper.

2. Related Works

Several works were reported on the implementation of sobel edge detection algorithm using different platforms as shown in Table 1.

Table 1. Related Works

Related Works	Year	Platform Used	Inference
[3]	2007	FPGA	Complex in hardware architecture
[4]	2008	FPGA	Complex in hardware architecture
[5]	2012	FPGA	Suffers from space and time complexity
[6]	2013	FPGA	Complex in hardware architecture
[7]	2014	FPGA	Complex in hardware architecture
[8]	2014	GPU (NVIDIA GeForce 310) & Xilinx Virtex-5 FPGA device	FPGA implementation is speed efficient than GPU.
[9]	2017	FPGA	Complex in hardware architecture
[10]	2018	FPGA	Complex in hardware architecture
[11]	2018	Xilinx Spartan 6 FPGA	Reduced the number of resources and space complexity with higher clock rate than the work proposed in [5]
[12]	2018	FPGA	No open source implementation
[13],[14]	2015, 2018	OpenCL software on Intel Terasic DE5 target FPGA device	Shows significant improvement in performance by employing pipelining but at the cost of hardware resources. Thus OpenCL is not preferred to program FPGAs.
[15]	2020	FPGA	Fastens the performing processes and has reduced the spatial complexity of the FPGA with increase in hardware resource.
[16]	2021	Software platform employed (MATLAB)	Instead of employing the hardware accelerator, software based method is employed using Standard C, AVX intrinsics and OpenMP directives. Software development time is reduced at the cost of latency.
[17]	2021	FPGA	Edge detected image cannot be obtained completely and the time and space complexity are high. No open-source implementation.
[18]	2022	MATLAB and OpenCV, Xilinx	Increased the hardware resource as well as the time delay gets affected. Also FPGA has increased the performance when compared to software method.

From the literature survey, it is inferred that FPGA implementation of sobel algorithm offers very good performance when compared to software based methods. The works proposed using FPGA either it has increased the hardware complexity or it has increased the time complexity. Further, there is no open source implementation. The motivation of the proposed work is to increase the speed and minimize the power consumption by making the hardware architecture simpler.

3. Methodology

An FPGA implementation of the Sobel algorithm typically consists of a set of convolution filters, a gradient computation unit, and a thresholding unit. To implement the convolution filters, the Sobel algorithm requires a set of shift registers and a set of multipliers. The shift registers are used to store the image data and shift it through the filters, while the multipliers are used to perform the convolution operation. The filters can be implemented as hardwired logic or as programmable functions, depending on the specific requirements of the application. The gradient computation unit combines the results of the horizontal and vertical filters to produce the gradient image. This unit may include a set of adders, subtractors, and absolute value units to compute the gradient magnitude and direction. The thresholding unit is used to highlight the edges of interest by applying a threshold to the gradient image. This unit may include a comparator and a register to store the threshold value. Figure 1 shows the Sobel Kernels.

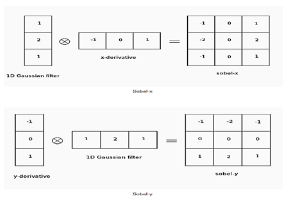


Figure 1. Sobel Kernels

In earlier works reported so far, sobel filter has been implemented with the use of multipliers and square root operation for the kernel convolution. While this might be available in higher end FPGAs, it is not always viable to include these operations in all implementations. The proposed work aims to eliminate these two operations. A high similarity index is also achieved by eliminating the multiplier and square root.

Also with the move towards the open-source hardware, there is a dire need of open source implementations of many systems. Till date, there is not much work in open-source hardware implementation of sobel filter. Closed-source hardware is proprietary to the company or individual who owns it and is not publicly available. It is typically distributed under a license that grants users the right to use the hardware but not to modify or distribute the source code or design files.

3.1 Mathematical Description of the Proposed Sobel Algorithm

Let S_x and S_y be the matrices obtained after applying the kernel convolution. Some of the entries will contain negative values. To remove these values, magnitude is taken by applying first multiplying the matrix (dot product) and then applying the square root operation.

Instead of this, a simple sign reversal is used to remove the negative signs and then they are added together. This is then repeated for the entire image. This has been explained in Figure 2.

$$If\ S_X < 0:$$
- $Then\ S_X = -S_X$

$$If\ S_Y < 0:$$
- $Then\ S_Y = -S_Y$

$$S = S_X + S_Y$$

Figure 2. Proposed sobel algorithm

3.2 Matlab Simulation Flowchart

Figure 3 shows the flowchart of the MATLAB Simulation. The necessary in-built functions are imported as needed. It is done for both the existing and proposed model for various scenarios. The results are then compared.

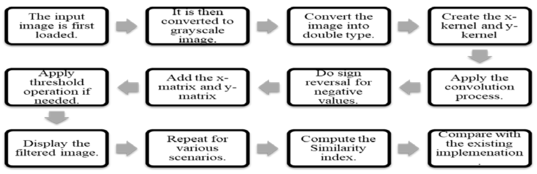

The input image is first loaded.
It is then converted to grayscale image.
Convert the image into double type.
Create the x-kernel and y-kernel
Apply the convolution process.
Do sign reversal for negative values.
Add the x-matrix and y-matrix
Apply threshold operation if needed.
Display the filtered image.
Repeat for various scenarios.
Compute the Similarity index.
Compare with the existing implemenation

Figure 3. Matlab Simulation Flowchart

3.3 Verilog Implementation Flowchart

Figure 4 shows the Verilog implementation of the proposed Sobel algorithm. It is a pipelined implementation. Pipelined design is a technique that is used to improve the performance of digital circuits by breaking a complex operation into smaller stages and executing them in parallel. In Verilog, pipelined design can be implemented by creating a pipeline of stages, each of which performs a specific operation on the data.

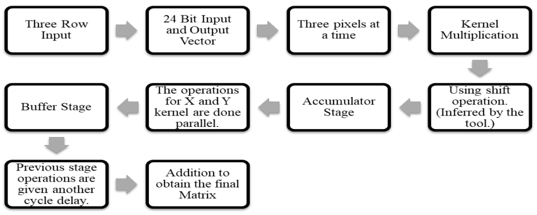

Three Row Input
24 Bit Input and Output Vector
Three pixels at a time
Kernel Multiplication
Using shift operation. (Inferred by the tool.)
Accumulator Stage
The operations for X and Y kernel are done parallel.
Buffer Stage
Previous stage operations are given another cycle delay.
Addition to obtain the final Matrix

Figure 4. Verilog Implementation Flowchart

To implement pipelined design in Verilog, a combination of registers and combinational logic can be used. Each stage performs a specific operation on the data and then passes the result to the next stage. The stages are connected by registers, which hold the data as it moves through the pipeline.

3.4. Real-Time Implementation of the Proposed Sobel algorithm using Cyclone III FPGA

The requirements for the real time implementation are:

- Live feed from an input source
- Input source – can be a camera (OV7670) or from PC through UART Module
- Need a buffer for read and write operations.
- On-board SRAM insufficient.
- SDRAM – For holding the data temporarily
- FIFO Buffer for interfacing modules with different speeds

The OV7670 is a low-power CMOS image sensor designed for use in portable devices such as mobile phones, laptops, and digital still cameras. It is a VGA (640x480) resolution image sensor that can operate at up to 60 frames per second and is capable of capturing high-quality images and video in a wide range of lighting conditions. The OV7670 includes on-chip image processing functions, such as color interpolation, gamma correction, and white balance, which can improve the quality of the captured images. The OV7670 has good low light sensitivity, allowing it to capture good-quality images in low light conditions.

SDRAM DDR2 module is the most challenging part of the real-time implementation. To interface an FPGA (field-programmable gate array) with DDR2 SDRAM, design and implement a memory controller that can handle the communication between the FPGA and the SDRAM. The memory controller is responsible for managing the transfer of data between the FPGA and the SDRAM, as well as handling the timing and control signals required for the SDRAM to operate correctly.

The PLL circuit includes a phase detector and a low-pass filter. The phase detector compares the input clock with the output clock and generates a feedback signal based on the phase difference between the two clocks. The low-pass filter filters the feedback signal and generates the output clock. The proposed sobel algorithm is implemented using Cyclone III FPGA as shown in Figure 5.

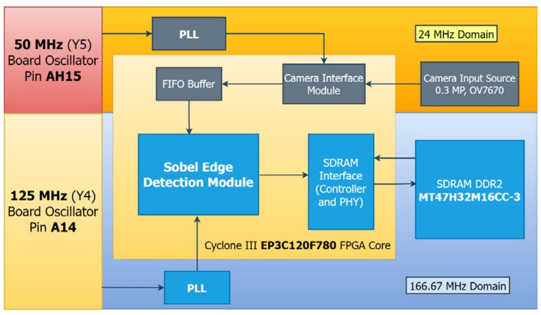
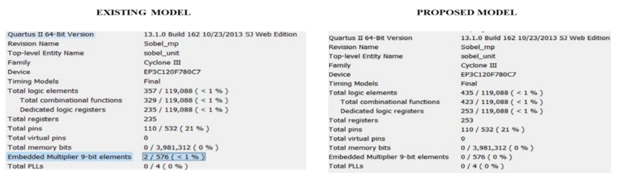

50 MHz (Y5)
Board Oscillator
Pin AH15
PLL
24 MHz Domain
FIFO Buffer
Camera Interface Module
Camera Input Source 0.3 MP, OV7670
Sobel Edge Detection Module
SDRAM Interface (Controller and PHY)
SDRAM DDR2 MT47H32M16CC-3
125 MHz (Y4)
Board Oscillator
Pin A14
Cyclone III EP3C120F780 FPGA Core
PLL
166.67 MHz Domain

Figure 5. Proposed sobel edge architecture using Cyclone III FPGA

The Cyclone III FPGA core has two parts working in different clock domains. The camera module continuously senses and sends the data to the camera interface module. It is then interfaced with the Sobel Unit using a FIFO buffer. It allows the data to be read in the same order in which it is inserted. The results are then stored in the RAM module. PLL is used to provide the necessary clocks from the on-board oscillator. The specification sheets should be referred to know the frequency of operation of various peripherals.

The orange coloured regions depict the domain, which works in 24 MHz clock domain. The blue is for 166.67 MHz clock domain. The camera works in I2C mode.

PHY is the physical layer of the SDRAM Controller. An FSM based design is used. The IP is obtained from the megacore function wizard of Quartus II. While creating the IP, specifications such as the frequency, burst mode, frames, error correction mode must be set properly in accordance with datasheet of the SDRAM DDR2 controller.



EXISTING MODEL

Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Sobel_mp
Top-level Entity Name	sobel_unit
Family	Cyclone III
Device	EP3C120F780C7
Timing Models	Final
Total logic elements	357 / 119,088 (< 1 %)
Total combinational functions	329 / 119,088 (< 1 %)
Dedicated logic registers	235 / 119,088 (< 1 %)
Total registers	235
Total pins	110 / 532 (21 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	2 / 576 (< 1 %)
Total PLLs	0 / 4 (0 %)

PROPOSED MODEL

Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Sobel_mp
Top-level Entity Name	sobel_unit
Family	Cyclone III
Device	EP3C120F780C7
Timing Models	Final
Total logic elements	435 / 119,088 (< 1 %)
Total combinational functions	423 / 119,088 (< 1 %)
Dedicated logic registers	253 / 119,088 (< 1 %)
Total registers	253
Total pins	110 / 532 (21 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 576 (0 %)
Total PLLs	0 / 4 (0 %)

4. Results and Discussion

Three samples were chosen to compare the existing and the proposed model. Figure 6, shows the MATLAB simulation results for the given sample. From the simulation results, it is inferred that the outputs of the proposed Sobel model is virtually indistinguishable from the output of the existing sobel model.

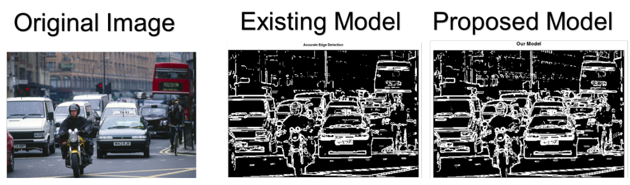


Figure 6. MATLAB Comparison of the Proposed and the Existing Sobel Model

A similarity index of **96.43%** has been achieved with the proposed model in comparison with the existing model. Figure 7 shows the Quartus II RTL netlist of the proposed sobel model.

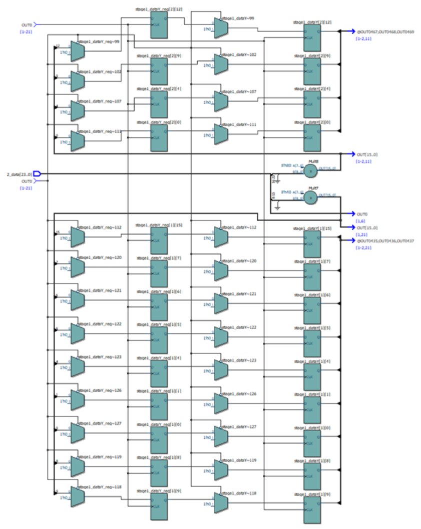


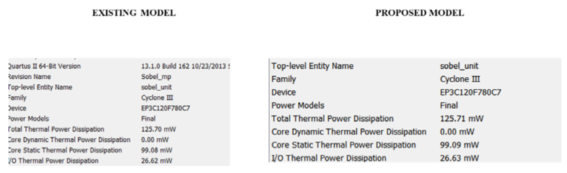
Figure 7. Quartus II RTL netlist view

4.1 Performance Metrics of the Proposed and the Existing Sobel Algorithm

The performance metrics chosen are number of logic elements, total thermal power dissipation, maximum frequency of operation and the number of multipliers used. Figure 8 shows the number of logic elements utilized in the existing and in the proposed work.

Figure 8. Comparison of the hardware complexity

Figure 9 compares the thermal power dissipation in the existing and in the proposed work.



EXISTING MODEL

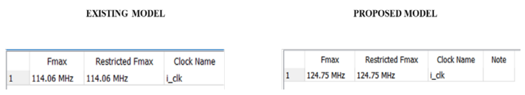
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013
Revision Name	Sobel_mp
Top-level Entity Name	sobel_unit
Family	Cyclone III
Device	EP3C120F780C7
Power Models	Final
Total Thermal Power Dissipation	125.70 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	99.08 mW
I/O Thermal Power Dissipation	26.62 mW

PROPOSED MODEL

Top-level Entity Name	sobel_unit
Family	Cyclone III
Device	EP3C120F780C7
Power Models	Final
Total Thermal Power Dissipation	125.71 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	99.09 mW
I/O Thermal Power Dissipation	26.63 mW

Figure 9. Thermal Power dissipation comparison

Figure 10 shows the maximum frequency of operation in the existing and in the proposed work.



EXISTING MODEL

	Fmax	Restricted Fmax	Clock Name
1	114.06 MHz	114.06 MHz	i_clk

PROPOSED MODEL

	Fmax	Restricted Fmax	Clock Name	Note
1	124.75 MHz	124.75 MHz	i_clk	

Figure 10. Maximum frequency of operation comparison

From Figures 8, 9 and 10, it is inferred that the proposed work has completely eliminated the use of multiplier. However this has led to the slight increase 21.62% in the number of logic elements used. The thermal power dissipation remains the same in both the models. However the proposed sobel algorithm operates at a higher speed than the existing sobel algorithm.

4.2 Real Time implementation results of the Proposed sobel Algorithm using the open source YOSYS tool

Real time implementation of the proposed sobel algorithm has been carried out by interfacing camera with Cyclone III FPGA module using the open source YOSYS tool. Figure 11 shows the YOSYS RTL netlist.

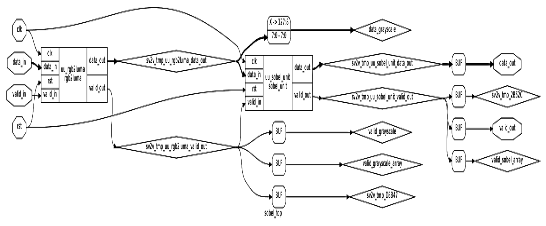


Figure 11. YOSYS RTL netlist

Figure 12 shows the DDR2 SDRAM RTL netlist.

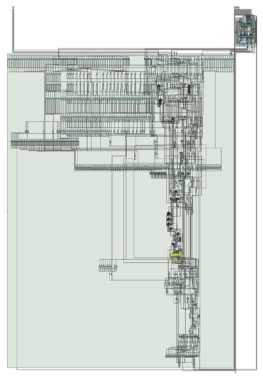


Figure 12. DDR2 SDRAM RTL netlist

Figure 13 shows the real time implementation RTL netlist of the proposed sobel algorithm by interfacing the camera module with Cyclone III FPGA.

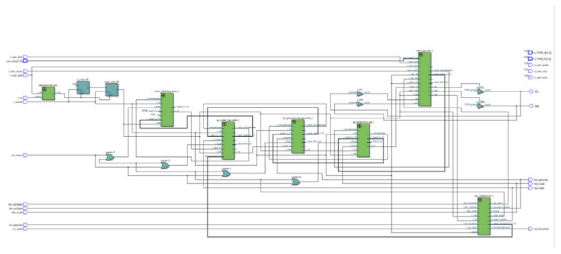
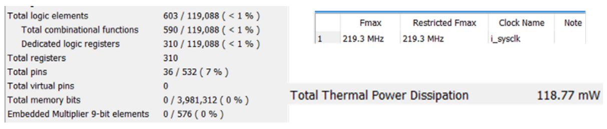


Figure 13. RTL Netlist of the proposed sobel edge algorithm

Figure 14 shows the performance metrics of the proposed sobel algorithm in real time.



Total logic elements	603 / 119,088 (< 1 %)
Total combinational functions	590 / 119,088 (< 1 %)
Dedicated logic registers	310 / 119,088 (< 1 %)
Total registers	310
Total pins	36 / 532 (7 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 576 (0 %)

	Fmax	Restricted Fmax	Clock Name	Note
1	219.3 MHz	219.3 MHz	i_sysclk	

Total Thermal Power Dissipation	118.77 mW

Figure 14. Real-time implementation Performance metrics

From the results, it is inferred that the proposed sobel algorithm in real time uses totally 603 logic elements, dissipates 118.77 mW and operates at a speed of 219 MHz as shown in Table 2.

Table 2. Performance Metrics of the proposed sobel edge in real time

Performance metrics	**Proposed Sobel algorithm in Real Time**
Logic Element Utilization	603
Total Thermal Power Dissipation	118.77 mW
Maximum Frequency of Operation	219 MHz
No. of Multipliers Used	0
Total Registers	310

Figure 15 shows the real time implementation of the proposed sobel algorithm by interfacing the camera module with Cyclone III FPGA.

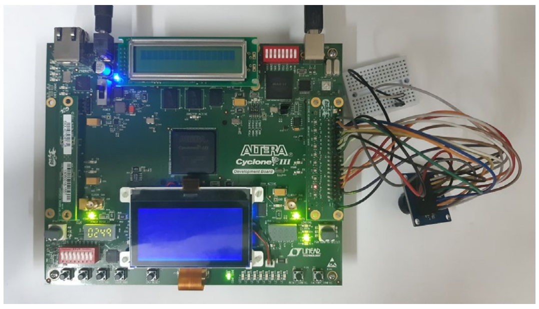


Figure 15. Real time implementation of the proposed sobel algorithm by interfacing the camera module with Cyclone III FPGA

References

[1] B. Saha Tchinda, D. Tchiotsop, M. Noubom, V. Louis-Dorr, and D. Wolf, "Retinal blood vessels segmentation using classical edge detection filters and the neural network," Informatics in Medicine Unlocked, 23, 2021, p.100521, doi: 10.1016/j.imu.2021.100521.

[2] Y. H. Kwon and J. W. Jeon, "Comparison of FPGA Implemented Sobel Edge Detector and Canny Edge Detector", In Proceedings of the IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), 2020, pp. 1-2, doi: 10.1109/ICCE-Asia49877.2020.9277425.

[3] A. Abbasi, M. Abbasi, A proposed FPGA based architecture for sobel edge detection operator, J. Act. Passive Electron. Devices, 2, 2007.

[4] I. Yasri, N. H. Hamid, and V. V. Yap, "Performance analysis of FPGA based Sobel edge detection operator," 2008 International Conference on Electronic Design, Dec. 2008, doi: 10.1109/ICED.2008.4786751.

[5] S. Halder, D. Bhattacharjee, M. Nasipuri, D.K. Basu, A Fast FPGA Based Architecture for Sobel Edge Detection, Springer, 2012.

[6] J. Monson, M. Wirthlin, B. L. Hutchings, "Optimization techniques for a high level synthesis implementation of the Sobel filter", In Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2013, pp. 1-6. https://doi.org/10.1109/ReConFig.2013.6732315.

[7] G. Chaple, R.D. Daruwala, "Design of Sobel operator based image edge detection algorithm on FPGA", In Proceedings of the International Conference on Communication and Signal Processing, 2014, pp. 788-792. https://doi.org/10.1109/ICCSP.2014.6949951

[8] M. Chouchene, F. E. Sayadi, Y. Said, M. Atri, and R. Tourki, "Efficient implementation of Sobel edge detection algorithm on CPU, GPU and FPGA", International Journal of Advanced Media and Communication, 5,(2/3), 2014, p.105.

[9] M. Amiri, F. M. Siddiqui, C. Kelly, "FPGA-Based Soft-Core Processors for Image Processing Applications" J Sign Process Syst, 87, 2017, pp. 139–156. https://doi.org/10.1007/s11265-016-1185-7

[10] K. Zhang, Y. Zhang, P. Wang, Y. Tian, and J. Yang, "An improved sobel edge algorithm and FPGA implementation," *Procedia Computer Science*, 131, 2018, pp. 243–248, doi: 10.1016/j.procs.2018.04.209.

[11] N. Nausheen, A. Seal, P. Khanna, S. Halder, "A FPGA based implementation of Sobel edge detection," Science Direct, Microprocessors and Microsystems, 56, 2018, pp. 84-91, https://doi.org/10.1016/j.micpro.2017.10.011.

[12] Z. Xiangxi, Z. Yonghui, Z. Shuaiyan, Z. Jian, "FPGA implementation of edge detection for Sobel operator in eight directions", In Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems, Chengdu. 2018, pp. 520-523.

[13] K. Hill, S. Craciun, A. George, H. Lam, "Comparative analysis of OpenCL vs. HDL with image-processing kernels on stratix-v FPGA", In Proceedings of the IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2015, https://doi.org/10.1109/asap.2015.7245733.

[14] H. Waidyasooriya, M. Hariyama, and K. Uchiyama, "Design of FPGA-based computing systems with OpenCL", Springer International Publishing, 2018, https://doi.org/10.1007/978-3-319-68161-0

[15] D. R. Menaka, D. R. Janarthanan, and D. K. Deeba, "FPGA implementation of low power and high speed image edge detection algorithm," Microprocessors and Microsystems, 75, 2020, p. 103053, doi: 10.1016/j.micpro.2020.103053.

[16] G. K. Ijemaru et al., "Image processing system using matlab-based analytics," Bulletin of Electrical Engineering and Informatics, 10, (5), 2021, pp. 2566–2577, doi: 10.11591/eei.v10i5.3160.

[17] M. A. Dávila-Guzmán, R. G. Tejero, M. Villarroya-Gaudó, D. S. Gracia, L. Kalms, and D. Göhringer, “A Cross-platform openVX library for FPGA accelerators,” In Proceedings of the 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2021, 2021, pp. 75–83, doi: 10.1109/PDP52278.2021.00020.

[18] Ahmed Khazal Younis, Basma MohammedKamal Younis, Mohammed Sabah Jarjees, “Hardware implementation of Sobel edge detection system for blood cells images-based field programmable gate array”, Indonesian Journal of Electrical Engineering and Computer Science, 26, (1), 2022, pp. 86~95 ISSN: 2502-4752, DOI: 10.11591/ijeecs.v26.i1.pp86-95.