ASEAN Engineering Journal

Full Paper

FORWARD DIFFERENCE EDGE DETECTOR BASED ON FPGA IMPLEMENTATION

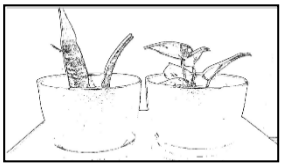
Shatha AbuShanab

Department of Computer Systems Engineering, Faculty of Engineering and Technology, Palestine Technical University – Kadoorie (PTUK), Palestine

Article history
Received
01 February 2025
Received in revised form
02 May 2025
Accepted
08 July 2025
Published online
28 February 2026

Shatha AbuShanab
shatha.abushanab@ptuk.edu.ps

Graphical abstract



Abstract

This paper aims to develop and implement the forward difference edge detector on different Field Programmable Gate Array (FPGA) technologies. It investigates how different CMOS technologies influence the power dissipated by digital design. Digital design in terms of low power has become a very challenging problem. Recently, FPGAs, which are semiconductor devices, have been targeted for image processing applications to reduce the computational times needed due to their parallelism techniques. The forward difference algorithm is developed at a high level of abstraction and is performed using physical instruments to obtain real data that is valuable and essential. ModelSim Altera and Quartus Intel are used to describe the algorithm, selecting the VHDL language. The digital design for the forward difference edge detector is completely implemented and verified on Cyclone IV, Cyclone V, and Cyclone LP 10 FPGA devices. At the hardware level, more information is obtainable to estimate the power dissipation correctly. The results indicate CMOS technologies used can be determined based on the design specifications and their power dissipation.

Keywords: *FPGA, Forward Difference Operator, Edge Detector, VHDL, Image Processing*

1.0 INTRODUCTION

Image processing can be defined as a sequence of mathematical operations applied to an image to enhance, detect, or measure specific features within it. The edge detection technique is used commonly in digital image processing applications, for which numerous algorithms have been developed [1–4]. The rapid scaling of CMOS technology has enabled to implement of a complete digital system on a single chip. Furthermore, the evolution of microelectronics technology determines significant changes in the design approach, particularly as Field Programmable Gate Array (FPGA) devices. FPGAs provide various logic resources, including flip-flops, multipliers, memory blocks, and high-speed input/output interfaces. These resources support the implementation of designs from simple digital design to complex systems on an FPGA chip [5–7]. Consequently, many researchers have proposed the use of FPGA for various image-processing applications [8–11].

Azhari, Setumin, Noorsal, and Abdullah propose an enhancement algorithm on hardware to improve the image quality. MATLAB was used to convert the input image from bitmap format into a hexadecimal file. The functionality of the suggested design was verified using the ISIM simulator from ISE Xilinx and the system stored the output RGB pixel in the output image memory. The brightness and contrast for each fetched pixel value from the memory block were modified using the proposed method. The design was coded using hardware description language to be implemented on hardware [12].

Ravivarma, Gavaskar, Malathi, Assha, Ashok, and Aarthi present a hardware Sobel edge detection algorithm using FPGA to reduce space and time complexity. This study aims to avoid broken edges and thick boundaries in real-time image processing [10].

Mahalle and Shah develop and implement various gradient-based edge detection algorithms on Spartan-3E FPGA. Xilinx System Generator is used to design Robert, Prewitt, Sobel, and Canny edge detection algorithms for hardware implementation and eliminate the need for Hardware Description Language (HDL) coding. The implementation and investigation of the different edge detectors indicated that the use of an algorithm can be determined based on its application [13].

Orthy, Islam, Rashid, and Hasan focus on explaining the enhancement techniques for detecting edges in diabetic retinopathy (DR) images. Their object is to optimize the visual

quality of images to assist additional facile disease detection. The proposed model utilizes FPGA technology for simulation to reduce the time required compared to software approaches [14].

This research investigates the implementation of the forward difference edge detection operator on different CMOS technologies using different FPGA devices, focusing on its dissipated power and the design specification. The proposed design is completely implemented on an FPGA chip. The hardware implementation is targeted because hardware provides parallelism techniques essential for image processing applications as well as for estimating the power dissipation correctly.

The paper is divided into five sections. This section provides an introduction to the research that developed and implemented the forward difference edge detection on various FPGA technologies. Section two introduces and defines the forward difference edge detection. In section three the implementation of the forward difference edge detector and the methodology of the research have been found. The penultimate section presents and discusses the results obtained. A conclusion can be found in section five.

2.0 FORWARD DIFFERENCE EDGE DETECTION

An edge can be defined as a set of connected pixels that forms a boundary between two discontinuous regions. The edge detection operator locates the pixels (points) that create these edges (lines) in an image. Edge detection can be found through a derivative approach. Thus, a gradient operator is often used for this purpose, as it can detect the discontinuity in intensity between neighbouring pixels. The derivative is often used to detect local intensity change at each pixel in an image. The gradients for horizontal (x) and vertical (y) directions are defined as vectors in Eq. (1) [1, 2]:

$$g[f(x,y)] = \begin{bmatrix} g_x(x,y) \\ \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \quad (1)$$

The horizontal gradient gx and the vertical gradient gy can be calculated using the forward difference method in terms of finite differences at every pixel location in an image, as described in Eq. (2) for the x-direction and Eq.(3) for the y-direction respectively [1, 2].

$$g_x(x,y) = \frac{\partial f(x,y)}{\partial x} = f(x+1) - f(x) \quad (2)$$

$$g_y(x,y) = \frac{\partial f(x,y)}{\partial y} = f(y+1) - f(y) \quad (3)$$

The magnitude of the vector gx and gy can be calculated by Eq. (4) [1, 2]

$$g[f(x,y)] = \sqrt[2]{g_x(x,y)^2 + g_y(x,y)^2} \quad (4)$$

The magnitude can be approximated using absolute values as described in Eq. (5) [1, 2]:

$$g[f(x,y)] = |g_x(x,y)| + |g_y(x,y)| \quad (5)$$

Edge pixels' p (x, y) can be found by comparing the gradient with the threshold value. If the computed gradient is greater than the threshold value, it is assigned a value of 0 where an edge pixel is found. Conversely, if the gradient is less than the threshold, the pixel is considered non-edge and assigned a value of 1, as described in Eq. (6) [1, 2].

$$p(x,y) = \begin{cases} 0\,, & g > Thresold \\ 1\,, & otherwise \end{cases} \quad (6)$$

Figure 1 demonstrates a zoomed section of an image, each square is a pixel that has x and y coordinates. To calculate the partial derivative in the x-direction, the forward difference operator is used, which subtracts the intensity of the left pixel from that of the right pixel. Likewise, for the y-direction, the intensity of the pixel above is subtracted from the intensity of the pixel below in the neighbourhood. Next, the absolute values of gx and gy are added, and the value of the result is compared with the threshold to find the edge pixels.

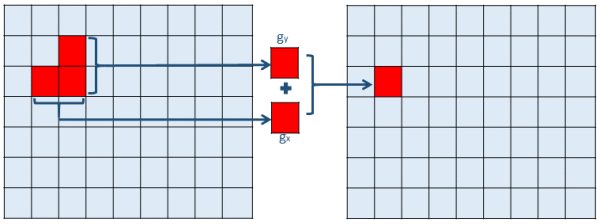


Figure 1 The Forward difference operator

3.0 FORWARD DIFFERENCE EDGE DETECTOR IMPLEMENTATION

The main idea of this approach is to develop and implement the forward difference edge detection algorithms on various FPGA devices and to investigate how using different CMOS technologies influences the power dissipated by the design. The forward difference edge detector is the focus of this study, compared to the Sobel, Prewitt, and Canny techniques, for several reasons: it requires less computational time and cost, using only subtractions, which makes it faster. Additionally, it uses low memory, making it preferable for real-time and FPGA applications in high-resolution images. The simplicity of the technique assists in investigating the impact of using different CMOS technologies and low-power design [1–6].

The methodology of this study consists of three sequential stages to assist in achieving a complete and successful design process, as well as to verify the functionality of the forward difference edge detection design, as illustrated in Figure 2. The stages are as follows:

1. Implementing the algorithm using MATLAB code
2. Verifying and simulating the algorithm using the ModelSim Altera program
3. Synthesizing and implementing the design using the Quartus Intel program on different FPGA technologies

The forward difference edge detector is implemented on FPGA devices to process the image, and the output is an image that contains the pixel edges. Thus, the input for the system is a two-dimensional (2-D) colour image. A 2-D digital image can be presented mathematically as two-dimensional matrices, where p

(x, y) indicates the intensity, with x and y representing spatial coordinates of that picture element or pixel. At the hardware level, the 2-D matrix will be read one pixel at a time for every clock cycle. Each pixel read has three colour components: Red (R), Blue (B), and Green (G).

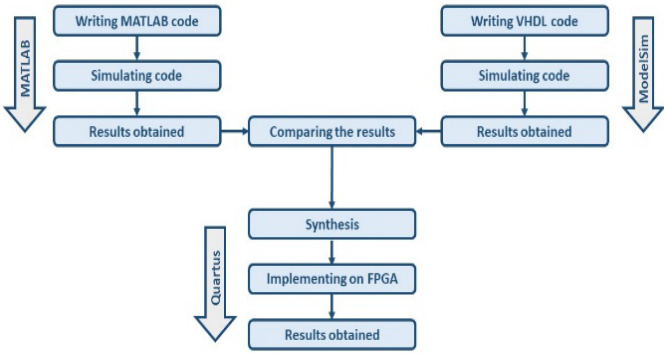


Figure 2 The sequential stages of the methodology used

As previously mentioned, this approach divided the work into three sequential stages. For each stage, the steps will be followed to apply the forward difference edge detector algorithm in horizontal direction, vertical direction, and both vertical and horizontal directions:

- Converting the input RGB image to a greyscale image, the luminosity method is used as described in Eq. (7). Grayscale conversion is essential because it uses less memory space and reduces processing time compared to an RGB image [15]

$$Grayscale = 0.299R + 0.587\,G + 0.114\,B \quad (7)$$

- Applying the horizontal and vertical equations to compute the gradient value gx and gy, the forward difference operator is applied using Eq. (2) and Eq. (3)
- Finding edge pixels in the horizontal direction, the computed value of gx is compared to the threshold. Similarly, for the vertical direction, the value of gy is compared to the same threshold. If the gradient value is greater than the threshold, this indicates finding an edge pixel, whose color will be black. If it is not greater than the threshold, the pixel colour is white, as described in Eq. (6)
- Finding edge pixels for both horizontal and vertical directions:
    - Adding the absolute magnitude of gx and gy to obtain the total gradient using Eq. (5)
    - Comparing the computed total gradient to the threshold and determining if the pixel is pixel edge as described in Eq. (6)

3.1 MATLAB Program

Many researchers frequently use the MATLAB program to verify and compare the results [16–18]. The MATLAB code is written to apply the forward difference edge detector algorithm and is run on MATLAB software. The output is a hexadecimal text file, which is compared and further converted to an image for observation. The sample image undergoes the algorithm and the output is three hexadecimal text files: one for the horizontal edge detection, one for the vertical edge detection, and a third for both horizontal and vertical edge detections.

3.2 ModelSim Altera Program

ModelSim Altera program is used to simulate and verify the functionality of the design before implementation on an FPGA device. VHDL, which is Hardware Description Language (HDL), describes the forward difference edge detector. The algorithm requires current and previous readings to find edge pixels for horizontal and vertical directions. Zero padding is used when edge computation begins with the first row or the first column. In this way, values outside the image's border; where previous reading is not available, are assumed to be zero [1, 2]. A column counter is essential to find the edge pixel in the horizontal direction. Every clock cycle, the system reads one pixel which presents the current reading. In the VHDL process, the last reading is preserved to be the previous reading and stored in registers. The row counter is necessary to find the edge pixel in the vertical direction. During the computation of the first row, each pixel that is read is also stored in RAM. Once the last pixel of a row is read, RAM contains all pixel values from that row. As the system proceeds to read the next row, it can access the previously stored values from RAM to perform forward difference edge detection computation in the vertical direction. The pixel directly above the current pixel is read from RAM and stored in a register for use in computation. RAM then overwrites a previous pixel with the current pixel, being it available for the next row's computation. This method effectively reduces the memory space needed.

The data for the system is a two-dimensional RGB image that will be processed by the implemented design. The RBG image is represented as an N x M matrix, where N denotes the number of rows and M denotes the number of columns, as shown in Figure 3 (left side). Each element or pixel, consists of three R, G and B components. At the hardware level, the image is read one pixel at a time during every clock cycle, that is R, G and B values corresponding to a single pixel. To validate this process, the MATLAB program is used to convert the image into a hexadecimal text file. This text file contains the R, G and B values for each pixel listed in a row, with the values for the next pixel being in the following row, as illustrated in Figure 3 (right side). Each pixel has 8 bits per component, resulting in 24 bits for one pixel. Therefore, each colour component ranges from 0 to 255.

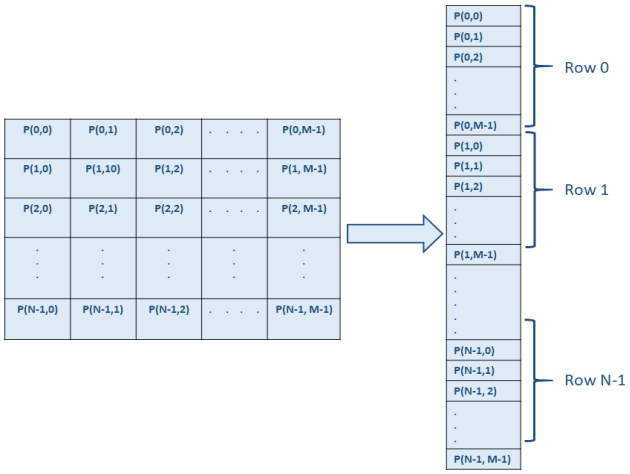


Figure 3 An image is a two-dimensional matrix, with M columns and N rows (left side) and a sequence stream array (right side)

For ModelSim Altera simulation, it is required to generate video input control signals are also required to simulate an image. The timing control signals of a two-dimensional digital image to read a frame: horizontal blanking, horizontal synchronisation pulse, vertical blanking, and vertical synchronisation pulse. The timing diagram for a two-dimensional scanned video image is illustrated in Figure 4. The visible region is where the data is read for both horizontal and vertical lines. The last horizontal line in the visible region follows as a trail with horizontal blanking and synchronous signal where no data is read. A similar process is used for controlling the vertical timing [8].

The output from ModelSim Altera simulation is a hexadecimal text file, which is then compared to the output text file from the edge detector using MATLAB. The results must match. Moreover, the output text file from ModelSim Altera can be converted into an image using MATLAB to be observed. At this stage, the design is verified the functionality and can be correctly implemented into various FPGA technologies.

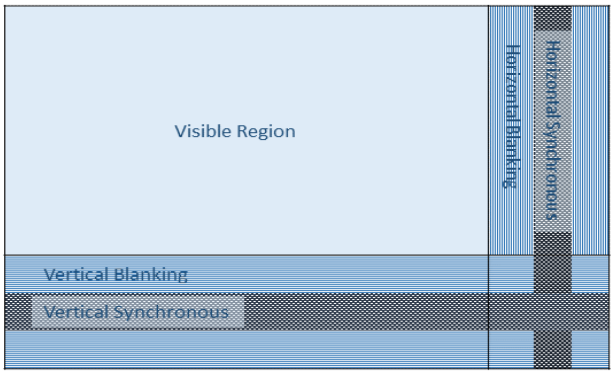


Figure 4 The timing diagram for a two-dimensional scanned video image

3.3 Synthesis and Implementation into FPGA

The objectives of this approach are to implement a forward difference edge detector design completely on different FPGA technologies and investigate the results obtained at the hardware level to obtain real and correct data. The input for the design will be an RGB video image. The remote lab has been used to implement the design into the FPGA technologies. The FPGA-vision-lab system is available accessible as an open educational resource: https://www.h-brs.de/de/fpga-vision-lab. This remote system uses image processing to apply a real application based on FPGA as a design platform, where the design is implemented and the actual results are obtained and measured [19–21]. Furthermore, the output from the remote system is an RGB image, and power dissipation is measured as well. Consequently, the input image must be correctly set for accessible remote lab, which requires resizing using MATLAB. The image resolution is 720x1280 to ensure proper processing.

The Quartus Intel program is used to synthesise the design described by VHDL. Synthesis is the process of building a physical design from an abstract described in VHDL, which is also simulated using ModelSim Altera. Once the synthesis process is successful, a binary file is generated for the algorithm to be implemented on different FPGA technologies. This design addresses the computing cost and complexity by converting R, G, and B to grayscale which is represented as integer value. Additionally, using integer data types allows for replacing division operations with multiplication, enabling the results to be scaled and compared to a scale threshold.

The FPGA-vision-lab system offers three FPGA families, Table 1 compresses the three FPGA devices concerning the requirements of the design [22–24]:

- Cyclone IV FPGA (EP4CE22E22C7): Cyclone IV FPGA devices are based on 60 nm semiconductor technology, which are often known for higher leakage currents and greater power dissipation due to their older technology
- Cyclone V FPGA (5CEBA2F17C6): Cyclone V FPGA devices are based on the 28 nm semiconductor technology used, which provides more logic resources compared to the previous families
- Cyclone 10 LP FPGA (10CL120ZF484I8G): Cyclone 10 FPGA devices are based on 60 nm semiconductor technology, which results in lower leakage current, and better power dissipation at the same or complex task.

Table 1: The comparison between Cyclone IV, Cyclone V, and Cyclone 10 FPGA devices

Item	Cyclone IV	Cyclone V	Cyclone10
Lithography	60 nm	28 nm	60 nm
Logic Elements (LE)	22,000	25,000	120,000
Maximum Embedded Memory	594 Kb	1.956 Mb	3.888 Mb
Adaptive Logic Modules (ALM)	-	9,434	-
User I/O	153	224	525

The design of the forward difference edge detector design was implemented on different FPGA technologies in the last stage: Cyclone IV, Cyclone V and Cyclone 10 LP FPGA devices. Quartus II Intel was used for the synthesis and generating of binary files necessary for implementing the design. As a result, the FPGA resources utilized for horizontal (x), vertical (y), and both horizontal and vertical (x-y) edge detectors, are presented in Tables 2, 3 and 4.

Table 2 FPGA resources utilized for Cyclone IV E (EP4CE22E22C7) technology

Resources	X	Y	X and Y
Logic Element	214 (<1%)	322 (1%)	431 (2%)
Register	173	266	342
Memory Bits	0 (0%)	30,720 (5%)	30,720 (5%)

Table 3 FPGA resources utilized for Cyclone V (5CEBA2F17C6) technology

Resources	X	Y	X and Y
Logic utilization (ALMs)	54 (<1%)	62 (<1%)	90 (<1%)
Register	180	181	255
Memory Bits	0 (0%)	30,720 (2%)	30,720 (2%)

Table 4 FPGA resources utilized for Cyclone 10 LP (10CL120ZF48418G) technology

Resources	X	Y	X and Y
Logic Element	224 (<1%)	239 (<1%)	349 (<1%)
Register	173	171	247
Memory Bits	0 (0%)	30,720 (<1%)	30,720 (<1%)

The resources utilized are determined by the specifications and requirements of the design. The resources utilized for any design are independent of the FPGA technology used during the design, which can be freely used and modified at a particular design stage.

Using a remote laboratory system enables to implement the design and meets the performance targets of reliability while also considering factors that influence on power dissipation of the design.

Figure 5 illustrates the graphical user interface (GUI) of the remote system. At the top centre, the FPGA technology is selected. To the right, the first input from a client will be loaded, which is a binary file from the synthesis process needed for completely implementing the design on the FPGA chip. In the middle of the figure, the core current and core supply voltage are measured and presented, enabling the calculation of the power dissipated by the design implemented on the FPGA chip; as a result, the power dissipation is an output of the system. At the bottom left, the second input (image) loaded by the client will be processed by the implemented design. The image in the bottom right corner is the output (processed) image, which is observed but not stored due to limited resources and the circuit complexity of FPGA. Pixel values are fed to FPGA via DVI cable as the input image, and those pixel values are then processed to obtain the output edge detected image via HDMI [19].

4.0 RESULTS AND DISCUSSION

After completing the three sequential stages as previously mentioned to correctly complete the design of the forward difference edge detector; the results from MATLAB code and ModelSim Altera are compared and must match, then synthesis and implementing the design on various CMOS technologies using different FPGA devices that are available using the remote FPGA board.

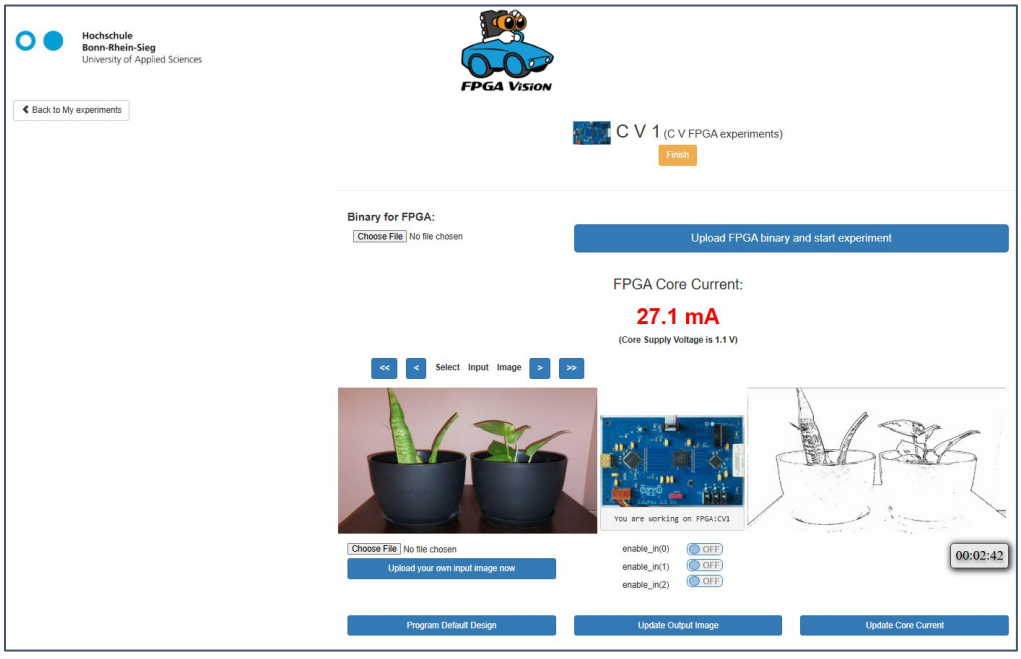


Figure 5 The GUI of the remote system

Table 5 presents the output from the forward difference edge detector, which is a processed image of the original image (input) shown in Figure 6 using MATLAB code, Modelsim Altera simulation, and the remote FPGA board. The results confirm that the algorithm successfully and correctly detects the edges in the horizontal, vertical, and both horizontal and vertical directions.

The edge detection operator in two directions provides improved accuracy in finding edge pixels compared to an operator that performs in one direction.

Moreover, Table 6 indicates that the power dissipation varies based on the CMOS technology used and the requirements of the design. This approach also aims to investigate the influence of CMOS technolgy used and its power dissipation. The algorithm is designed at a high level of abstraction, and the results are obtained and measured at a hardware level, where more information is available to correctly estimate the power dissipation. The design not only needs to complete the implementation correctly, but it also needs to know the characteristics of the target device, which can be a trade-off between the cost, power dissipation, and the requirements. The results indicate differences in power dissipation among Cyclone IV, Cyclone V, and Cyclone 10 FPGA devices.

Power dissipation in CMOS technology is dynamic and static power dissipation. Dynamic power is related to the switching activity and the short-circuit currents in the transistors of the digital design. The switching activity dissipates the power when the digital design is active during changing states. Short-circuit power dissipation occurs due to a direct current path from the power supply to the ground during the rise and fall times of transition intervals. Therefore, short-circuit power is affected by both the rise and fall times and the load capacitance. Static power dissipation includes leakage current categories in CMOS technology. As the transistor size shrinks, the leakage current increases.

Figure 6 Original Image

Table 5 The processed image using MATLAB code, ModelSim Altera simulations, and remote FPGA board

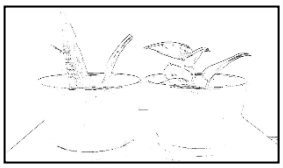
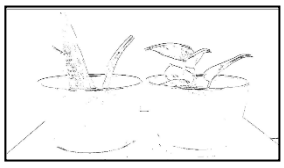
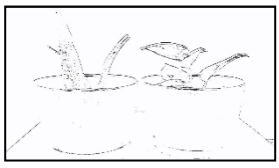
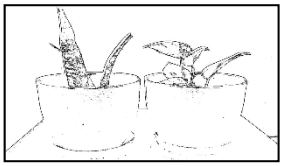
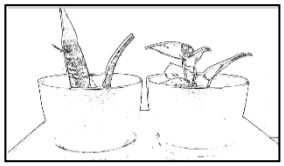
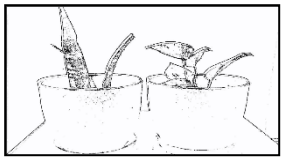
Output Image	MATLAB Code	ModelSim Simulation	FPGA board
x-direction			
y-direction			
Both x and y directions			

Table 6 Power dissipation of the forward difference edge detection operator based on the FPGA device

Direction	Cyclone IV	Cyclone V	Cyclone 10
X	8.89 mA	23.29 mA	4.65 mA
	1.2 V	1.1 V	1.0V
	10.67mW	25.62mW	4.65mW
Y	17.08 mA	26.11 mA	9.6 mA
	1.2 V	1.1 V	1.0 V
	20.50mW	28.72mW	9.6mW
X and Y	17.79 mA	27.1 mA	10.27 mA
	1.2 V	1.1 V	1.0V
	21.35mW	29.81mW	10.27mW

The results indicate differences in the power dissipated by the design on FPGAs influenced by the following [19] [22–24]:

- As a platform, FPGAs have different resources used to implement the design; the x-direction requires only registers to store the previous data, while the y-direction also requires RAM, the size of which corresponds to the number of pixels along the horizontal axis of an image. The results show that the edge detection dissipates less power in the x-direction than in the y-direction due to the differences in resources and their quantities. Thus, the two-directional edge detector utilizes more

resources and results in higher power dissipation than one detector on the same FPGA device.

- Each FPGA device includes different architectures and components utilized to implement the design, which also significantly impact the overall power dissipation depending on how the design is implemented in comparison to different FPGA devices used. Additionally, Cyclone V devices have advanced features, which adds more dynamic power dissipation if these features are not fully utilized. Conversely, Cyclone IV devices have fewer advanced features, which results in lower overall power dissipation in a low-utilization design.
- Process technology in FPGA devices influences the power dissipation of the implemented design. The Cyclone 10 FPGA shows lower power dissipation compares to Cyclone V and Cyclone IV. While Cyclone IV still dissipates less power than Cyclone V. The Cyclone IV FPGA is built on 60 nm semiconductor technology. The Cyclone V FPGA is based on the 28 nm semiconductor technology, providing greater logic resources than previous families and less power dissipation when the FPGA chip is fully utilized. In contrast, Cyclone V, which is a smaller CMOS technology, provides better performance and speed, but higher static power dissipation; the static power predominates over dynamic power due to a simple digital design implemented with less than 1% chip utilization, contributing to higher total power dissipation. The Cyclone 10 LP FPGA, on the other hand, has a lower leakage current and lower power dissipation for the same tasks.
- Power-saving techniques adopted in the FPGA device impact the power dissipation. Cyclone V FPGAs include a power-saving technique often designed to run at higher speeds and more complex designs, resulting in higher power dissipation than a simple design. The Cyclone 10 LP FPGA also provides advanced features, including improved power management, contributing to lower power dissipation compared to both Cyclone IV and Cyclone V.

The results for the power dissipation using different FPGA devices for the same digital design indicate that CMOS technologies can be selected based on the design specifications and their power dissipation.

5.0 CONCLUSION

This approach aims to develop and apply the forward difference edge detector using different CMOS technologies and investigate how the design specifications, as well as the utilized CMOS technologies influence the power dissipation. The design of the forward difference edge detector has been tested and verified for its functionality using MATLAB and ModelSim Altera. The design is described using VHDL and further synthesized using Quartus Intel for complete implementation on various CMOS technologies using different FPGA devices, from which the results are obtained. Compared to one-direction detection, the forward difference edge detector design for horizontal and vertical directions performs better in finding and accurately detecting edge pixels. On the other hand, the two-directional edge detector utilizes more resources, resulting in higher power dissipation.

The forward difference edge operator is selected due to its simplicity; the algorithm requires low memory, low computational time and cost, fewer resources, and a low power design to assist the study of its power dissipation, which is influenced by the CMOS technology used when implementing the same design for the forward difference edge detector on Cyclone IV, Cyclone V, and Cyclone 10 LP FPGA devices. Cyclone 10 LP FPGA device, which features more advanced process technology, newer architecture, and more advanced power management techniques, dissipates lower power than earlier technologies. While Cyclone IV, which is an older process technology, shows lower power dissipation than Cyclone V, this is primarily because the design is not fully utilized in Cyclone V. Consequently, Cyclone 10 is determined to be more power-efficient for the design. Ultimately, The results prove that the CMOS technology used can be determined based on the design specification and its power dissipation.

Acknowledgements

The author thanks the Palestine Technical University-Kadoorie (PTUK) for supporting this research.

Conflict of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper

References

[1] W. Burger and M. J. Burge, 2022. *Digital image processing: An algorithmic introduction*. Cham, Switzerland: Springer.

[2] R. C. Gonzalez and R. E. Woods, 2018. *Digital image processing*. New York, New York: Pearson Education, [Online]. Available: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5832133 4th Edition, Pearson Education, New York.

[3] A. Dehghani, A. Kavari, M. Kalbasi, and K. RahimiZadeh, 2022. "A new approach for design of an efficient FPGA-based reconfigurable convolver for image processing," (in En;en), *The Journal of Supercomputing* 78(2): 2597–2615, DOI: 10.1007/s11227-021-03963-6.

[4] A. Metkar, M. Maroo, A. Sawant, V. Singh, and S. Mhatre, 2020. *Hardware Implementation of Image Processing Algorithms on FPGA*, In *Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST).*

[5] Farhanaaz and V. Sanju, "Field Programmable Gate Array(FPGA): An Innovation In Hardware Technology," in *2023 2nd International Conference for Innovation in Technology (INOCON)* Bangalore, India, 1–6. DOI: 10.1109/INOCON57975.2023.10101210

[6] J. Ruiz-Rosero, G. Ramirez-Gonzalez, and R. Khanna, 2019. "Field Programmable Gate Array Applications—A Scientometric Review," *Computation*, 7(4): 63. DOI: 10.3390/computation7040063.

[7] P. Babu and E. Parthasarathy, "Reconfigurable FPGA Architectures: A Survey and Applications," *Journal of The Institution of Engineers (India): Series B 102(1)*: 143–156, 2021, DOI: 10.1007/s40031-020-00508-y.

[8] D. G. Bailey, 2024. *Design for embedded image processing on FPGAs*. Hoboken, NJ: John Wiley & Sons Inc

[9] T. Jagadesh, "Implementation of Prewitt Operator based Edge Detection Algorithm using FPGA," *International Journal for Research*

in Applied Science and Engineering Technology 8(5): 1078–1082, 2020, DOI: 10.22214/ijraset.2020.5171.

[10] G. Ravivarma, K. Gavaskar, D. Malathi, K. G. Asha, B. Ashok, and S. Aarthi, 2021, "Implementation of Sobel operator based image edge detection on FPGA," *Materials Today: Proceedings*, 45: 2401–2407. DOI: 10.1016/j.matpr.2020.10.825.

[11] L. Xu and D. Zheng, 2022, "A Novel Sobel Edge Detection Accelerator Based on Reconfigurable Architecture," *Traitement du Signal.* 39(4): 1421–1427. DOI: 10.18280/ts.390436.

[12] Z. I. Azhari, S. Setumin, E. Noorsal, and M. H. Abdullah, 2023. *Digital image enhancement by brightness and contrast manipulation using Verilog hardware description language*. *International Journal of Electrical and Computer Engineering,* 13(2), 1346-1357. DOI: 10.11591/ijece.v13i2.

[13] A. G. Mahalle and A. M. Shah, 2017. " FPGA Implementation of Gradient Based Edge Detection Algorithms," I*nternational Journal of Innovative Research in Computer and Communication Engineering* (IJIRCCE), 5(5)

[14] M. Orthy, S. M. R. Islam, F. Rashid, and M. A. Hasan, 2023, "Implementation of Image Enhancement and Edge Detection Algorithm on Diabetic Retinopathy (DR) Image Using FPGA," *IET Circuits, Devices & Systems*, 2023, 1: 1–12, DOI: 10.1049/2023/8820773.

[15] Z. N. Khudhair *et al.,* 2023. "Color to Grayscale Image Conversion Based on Singular Value Decomposition," *IEEE Access*, 11: 54629–54638, doi: 10.1109/ACCESS.2023.3279734.

[16] P. H. Lai, D. B. Nguyen, T. V. Nguyen, K. D. Ta, T. V. , Truong, and D. D. Nhu, 2020. "Mixed-signal generator module:Design and Verification in MATLAB and Verilog hardware description language," *International Journal of Emerging Trends in Engineering Research,* 8(9): 5064–5069, 2020, DOI: 10.30534/ijeter/2020/28892020.

[17] A. S. Karnea and S. S. Navalgunda, 2013*mplementation of an image thinning algorithm using Verilog and MATLAB*, *" International Journal of Current Engineering and Technology, (Ncwse),* 333-337.

[18] P. G. Patel, A. Ahmadi, and M. Khalid, 2021. "Implementing An Improved Image Enhancement Algorithm On FPGA," In 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 1-6. DOI: 10.1109/CCECE53047.2021.9569049.

[19] S. AbuShanab, 2018. "Remote and on-site laboratory system for low-power digital circuit design," Ph.D. thesis, Siegen. https://nbn-resolving.org/urn:nbn:de:hbz:467-13459. (accessed: Jan. 5, 2025).

[20] A. Schwandt and M. Winzker, 2023. "Virtual Mobility for All with the FPGA Vision Open Online Course," In International Conference on Remote Engineering and Virtual Instrumentation, 703-714. Cham: Springer Nature Switzerland. DOI: 10.1007/978-3-031-42467-0_66.

[21] S. AbuShanab, M. Winzker, and R. Bruck, 2015. "Remote low-power digital design system," in *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. 1–6.

[22] Altera: Cyclone® IV Featured Documentation - Quick Links Guide. https://www.intel.com/content/www/us/en/docs/programmable/767845/current/cyclone-iv-featured-documentation-quick.htm (accessed: Jan. 5, 2025).

[23] Altera: Cyclone® V Device Datasheet, 5CEBA2F17C6. https://www.datasheets360.com/part/detail/5ceba2f17c6/4718592257345144063/. (accessed: Jan. 5, 2025).

[24] Altera: Cyclone® 10 LP Device Overview. https://www.intel.com/content/www/us/en/docs/programmable/683879/current/device-overview.html. (accessed: Jan. 5, 2025).