TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 4, August 2021, pp. 1307~1315
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i4.18988

Detection of brain stroke in the MRI image using FPGA

Dheyaa Alhelal, Ahmed Khazal Younis, Ruaa H. Ali Al-Mallah
Department of Technical Computer Engineering, Technical Engineering College, Northern Technical University, Iraq

Article Info

Article history:

Received Dec 11, 2020
Revised Jan 5, 2021
Accepted Jan 20, 2021

Keywords:

FPGA
Image processing
Median filter
Morphological filters

ABSTRACT

One of the most important difficulties which doctors face in diagnosing is the analysis and diagnosis of brain stroke in magnetic resonance imaging (MRI) images. Brain stroke is the interruption of blood flow to parts of the brain that causes cell death. To make the diagnosis easier for doctors, many researchers have treated MRI images with some filters by using Matlab program to improve the images and make them more obvious to facilitate diagnosis by doctors. This paper introduces a digital system using hardware concepts to clarify the brain stroke in MRI image. Field programmable gate arrays (FPGA) is used to implement the system which is divided into four phases: preprocessing, adjust image, median filter, and morphological filters alternately. The entire system has been implemented based on Zynq FPGA evaluation board. The design has been tested on two MRI images and the results are compared with the Matlab to determine the efficiency of the proposed system. The proposed hardware system has achieved an overall good accuracy compared to Matlab where it ranged between 90.00% and 99.48%.

Corresponding Author:

Ahmed Khazal Younis
Department of Computer Technology Engineering, Engineering Technical College
Northern Technical University
Iraq
Email: ahmedkhazal@ntu.edu.iq

1. INTRODUCTION

Biomedical image is not easy to deal with because of complex nature for these images. Therefore, even good doctors have difficulty to analysis and diagnose some conditions in these images, especially in the brain images. There are many ways to reduce these difficulties for doctors to allow them to properly diagnose the disease. One of these ways is image segmentation. Image segmentation, clear from the name, is partition the image into multiple objects in order to simplify it to be ready for analysis and diagnosis. There is more than one technique for image segmentation such as morphological and thresholding which is usually done by using software applications such as Matlab [1]. On the other hand, there is another option to do that which is hardware tool such as field programmable gate arrays (FPGA).

FPGA Increasingly used to implement image processing applications especially for real-time embedded applications when power consumption and response time are important. The FPGA is an integral part of a smart camera that capable of performing a lot of image processing directly such as the image is followed from sensors. The parallelism of hardware is one of the embedded system advantages where is able to invest the locative (data level) and time (task level) parallelism that implied within a lot of image processing mission [2]. On the other hand, there are some limitations when FPGA devices are used for image processing and here are some examples. First, when FPGA is used to implement image filter, the most common processing mode for that is pipelined stream which is process one pixel per clock cycle and that will cause memory bandwidth problem when reading all the input pixels. In order to avoid that issue, row buffers are used to hold pixels from previous image [3]. Second, memory hierarchy issue (on-chip and off-chip memory) where there are advantage and disadvantage for each of them. For example, FPGA on-chip memory has low access time and at the same time it has small capacity compared to the off-chip memory (Dynamic memory). In some cases, there is no option but to choose on-chip memory especially when power consumption is important [4]. Despite the limitations and difficulties in FPGA implementation still worth because it will be stand-alone device by itself [5].

2. RELATED WORK

The working principle of linear filters depends on the addition and multiplication [6]. There are many designs for filters that have been proposed by researchers which can be classified according to the mechanism of its work into two categories [7]: first one is multiplier based filters which takes a set of multipliers and implement multiplications through them such as in [8]-[10]. While the second one is multiplerless filters which depend on avoiding using costly multipliers through representations and various arithmetic transformations such as works in [11]-[14]. Filters can also be classified according to the nature of their use such as median filter and morphological filters which are used in this article. Like all other fields, there are a lot of papers proposed median filter designs in different methods such as in [15]-[18]. Here other example with some details, the authors in [19] design and implement median filter by using very high speed integrated circuit hardware description language (VHDL) code in Xilinx ISE 12.2. The filter has ability to process 8-bit gray scale image and mask (3x3) is used for filtering. The filter can take care of salt and pepper noise for gray scale image. At the same time, it is fast enough to use it for real time image processing. Morphological operations are usually including two processes dilation and erosion. There are many researchers represented these filters using FPGA in different mechanism such as in [20]-[23]. Here are two examples with some details, first one in [24]. The authors depending on dynamic and partial reconfiguration (DPR) techniques, they proposed a hardware implementation of morphological operations by using xilinx tools and Virtex-5 FPGA board. Their design gives the ability to choose appropriate morphological operations (dilation or erosion) depending on the limitation of the image. As a result, they proved that using DPR can save at least 11% of area on FPGA and improving the performance at the same time according to their results. In the second example, the authors in [25] using the FPGA-based parallel implementation of morphological filters to present the hardware implementation for grayscale morphological operations where rectangular flat top structuring elements is used. Their design gives more than one advantage such as, throughput and processing frame rate are high, internal memory which is needed is low and low latency. The Xilinx design suite 14.2 ISE is used to synthesize the proposed architecture and Virtex-5 FPGA board is used to prototype it.

There are many papers deal with the magnetic resonance imaging (MRI) image by using FPGA and these are some examples but not limited to. The authors in [26], [27] use a graphical user interface where the Xilinx system generator and Matlab Simulink are linked together in order to enhance the quality of MRI image. In they fed that Improved image to artificial neural network ANN to be classified if it is normal or abnormal while in [27], they use some parameters such as mean, variance, and standard deviation to classify the tumor. The authors in [28] use stationary wavelet transform and principal component analysis (SWT-PCA) technique for the processing of fusion. Matlab Simulink and blocks generator system, Xilinx synthesized with synthesis tool are used to design and simulate of the proposed system. Here in [29], they enhance the MRI image by using enhanced canny edge detection then, they use a modified watershed segmentation algorithm to separate the tumor from the original image. The proposed system is implemented by using Xilinx Virtex-5 FPGA.

3. RESEARCH METHOD

In any image processing process, most of the time the preprocessing is the first step which is usually used to remove some unwanted features or enhance some other or it is used to do both of them in order to make the next step easier. In this paper, the preprocessing has two steps, first one aims to enhance the contrast of image and adjust the image intensity where it can be done by using histogram equaliuzation and the result multiply by factor (in our case 1.3) to detect noise and determine which filter should be applied in order to remove that noise. While the second step aims to improve the quality of the image by removing detected noise. In our work, the common filter which is used to remove the pepper and salt noise is median filter. This filter is used to remove unwanted noise and at the same time saving the sharpness of the image from change because this kind of filter is less sensitive than other linear filters.

After finishing the preprocessing operation that is implemented based on the median filter, the image is ready to the segmentation process. As is known, there are many methods for segmentation one of them is thresholding which can be summarized thus each pixel in the image has an intensity value between 0 and 255 (assuming the image is grayscale image). Picking a threshold from that range will change the image to binary image with two colors black and white (shape and its background). Sometimes thresholding will be not enough to get the target from the segmentation (like our case). So, there are still some frames that are not isolated. For this reson, the morphological operations (which are usually have two processes: dilation and erosion) are used. In dilation process, we start searching for the maximum value in a small group of pixels to set it to the center. While in erosion, the searching will be for the minimum value among a specified group of pixels. In fact, there are two uses for dilation and erosion. First one is called erosion to dilation (opening used) which is used to remove the background. While second one is called dilation to erosion (closing used) which is used to smooths the image by filling all the small holes. The decision on the choice rests with the researchers and the nature of the work. In our case, it was helpful to used closing used. This methodology is implemented by the FPGA board and the implementation flowchart will be described in the next section.

4. HARDWARE IMPLEMENTATION

Dealing with the hardware system is not that easy. Many problems may be appeared during the implementation process. So, the proposed system is divided into many parts to facilitate the best handling and implementation of it. The system includes four phases, as shown in the Figure 1. The four phases are preprocessing, ajust image, implementation of median filter, and implementation of morphological filters which are described with more details below.

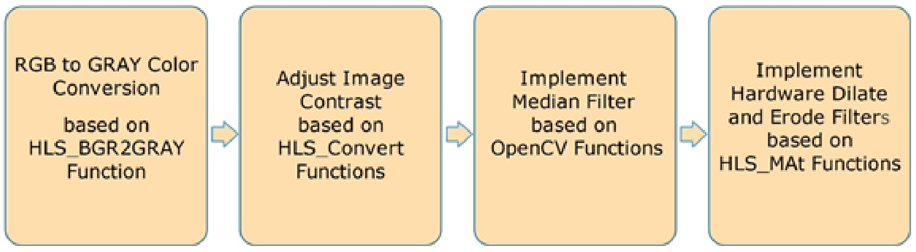

RGB to GRAY Color Conversion
based on HLS_BGR2GRAY Function
Adjust Image Contrast based on HLS_Convert Functions
Implement Median Filter based on OpenCV Functions
Implement Hardware Dilate and Erode Filters based on HLS_MAt Functions

Figure 1. Proposed system

4.1. Preprocessing stage

In this work and according to the Figure1, the first step in the preprocessing is changing the RGB image to grayscale image. In order to start that process, original image is imported as AXI-Streams, but this formula does not work with the high-level synthesis (HLS) environment. Therefore, it is necessary to convert advanced eXtensible interface (AXI) streams to the HLS: Mat format. This conversion will be repeated whenever is needed by using the AXIvideo2Mat and Mat2AXIvideo functions. The next step should be done to define the size and type of the HLS: Mat which are defined with the cvt_colour header file and calculate the maximum width and height, the number of channels and depth of each one.

4.2. Adjust image

The hardware implementation of adjust image contrast is done by a separate haredware intellectual property (IP) core side of programming logic (PL) part of Zynq system on a chip (SoC) in order to read frames from other system hardware cores. Pipelining and reshape optimization directives are used when implementing the adjust image in C-language under Vivado HLS tool to generate optimal RTL code improve the bandwidth by accessing the data in parallel all at once. During this step, the adjust image core receives the gray image from the preprocessing operations via AXI stream. As mentioned before, the conversion from AXI Streams to the HLS: Mat format or vice versa will be done whenever is needed to fit the next step.

4.3. Implementation of median filter

Usually, median filter is used to reduce image noise and it is preferred over others linear smoothing filters of same size because it provide a good noise reduction with less blurring. The principle of the filter's work is simply to take each pixel and compare it to its neighbors, and then decide whether to replace it with the median of those values or not. The Median filter is done based on the Cortex-A9 processor core side of programming system (PS) part of Zynq SoC within the FPGA architecture and open-source computer vision (OpenCV) video library. By using video libraries (which are already existing in Vivado HLS), we can migrate the OpenCV code to synthesizable C++ code. As a result, frame rate computer vision algorithms are implemented, and a high resolution is enabled by the synthesized blocks whenever integrated into a Zynq SoC design.

4.4. Implementation of Morphological filters

From the previous stage, which is the median filter, the data will send as stream to the Dilate IP-Core and then to the erode IP-Core. In both cases, the kernel that is used is a matrix 3x3 (ones). After that, the multiplication process takes place between the filter and the image. Then, the maximum value is extracted in case dilate and the minmum value is extracted in case erode. These two filters are applied in alternation and their results are stored in a specific BRAM (Block RAM) for the purpose of collecting and sending them to the computer by serial port. Finding the maximum value and the minmum value are done according to these two equations:

$$\mathrm{F}(x, y) = \max_{\substack{x-1 \le x' \le x+1 \\ y-1 \le y\prime \le y+1}} Image(x', y')$$

$$\mathrm{F}(x, y) = \min_{\substack{x-1 \le x\prime \le x+1 \\ y-1 \le y\prime \le y+1}} Image(x', y')$$

The Zynq AXI_lite interface connection is used to implement the proposed system as illustrate in Figure 2.

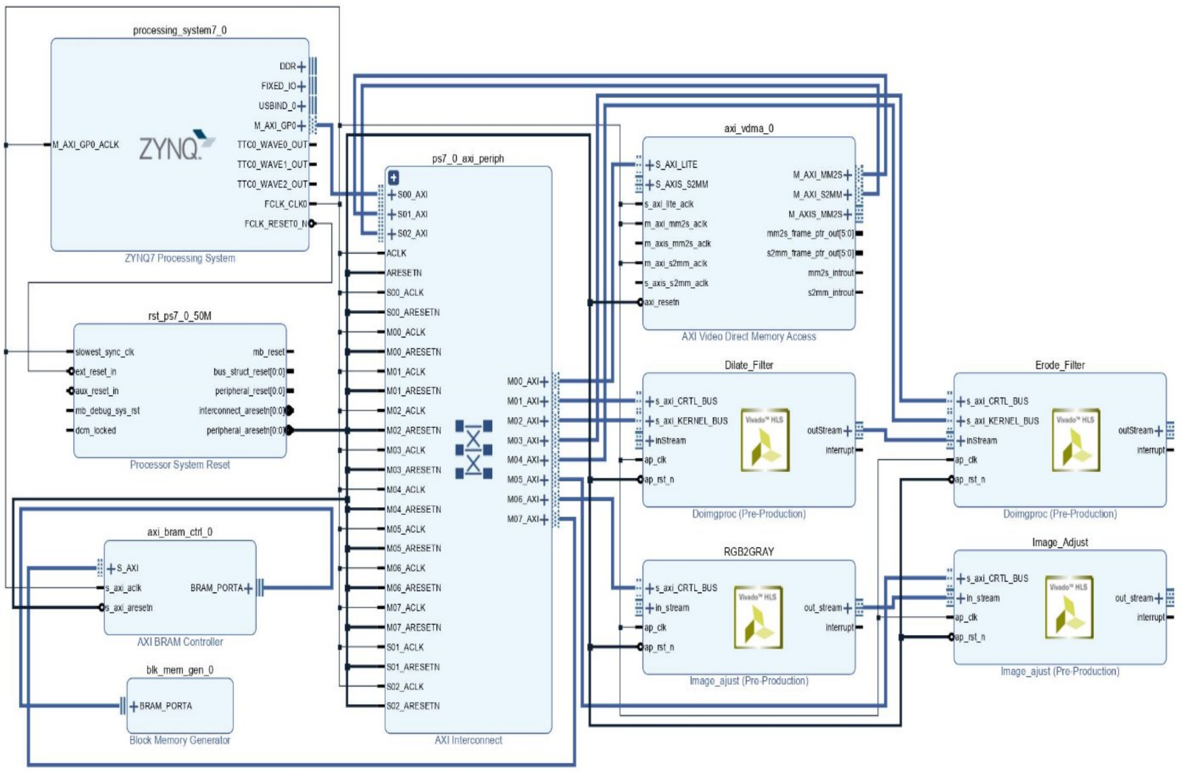


Figure 2. Overall proposed system on zynq chip

5. RESULTS AND ANALYSIS

The hardware resources for entire system are summarized in Figure 3. The Zynq xc7z020 evaluation kit comprise [30] 53200-LUT (lookup table), 106400-FF (flip-flop), 140-BRAM (block RAM), 32-BUFG, and 220-DSP (digital signal processor) block. While the implementation needs: 15% (7773) of LUT, 3% (491) of LUTRAM (RAM lookup table), 8% (8793) of FF, 6% (9) of BRAM, and 10% (23) of DSP block. In order to test the effectiveness of the proposed system, two different MRI images of brain strock are applied to our FPGA design. Then, the results are compared with Matlab program to measure the efficiency of our system as illustrated in the following flowchart:

According to the Figure 4, the flowchart includes the following steps:

- Original Image: Two MRI images are read which have different brain strock as illustrated in Figures 5 (a), (b), (c), and (d).

- Grayscale: MRI images are converted to grayscale image to simplify our work as illustrated in Figures 6 (a), (b), (c), and (d).
- Adjustment image: This step is used to adjust image intensity pixel value as illustrated in Figures 7 (a), (b), (c), and (d). Then, we multiply the results by factor 1.3 to detect the noise and which filter should be applied to remove the detected noise.
- Median filter: After using threshold segmentation method that was applied to segment brain stroke, removing salt and pepper noise are needed. Median filter is a common image enhancement technique for this step. It is used to extract an object from its background by using threshold value T=200 for each pixel such that each pixel is either classified as an object point or a background point as illustrated in Figures 8 (a), (b), (c), and (d).
- Dilate operator (close): This operation is used to fill some holes in MRI images to get more smoothens. The result of this step is illustrated in Figures 9 (a), (b), (c), and (d).
- Erode operation (open): It is used to remove the small objects which are not related to the stroke as illustrated in Figures 10 (a), (b), (c), and (d).

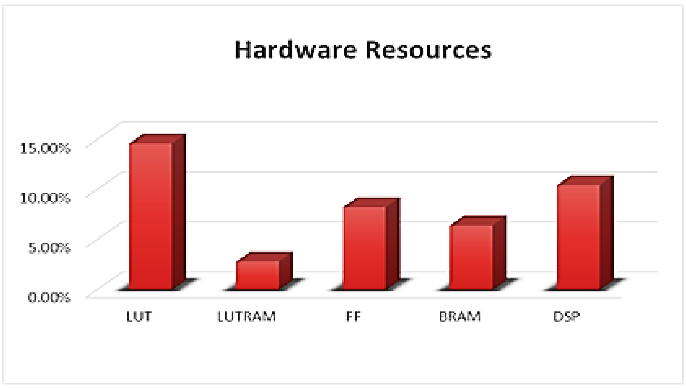


Figure 3. Resource utilization of the hardware

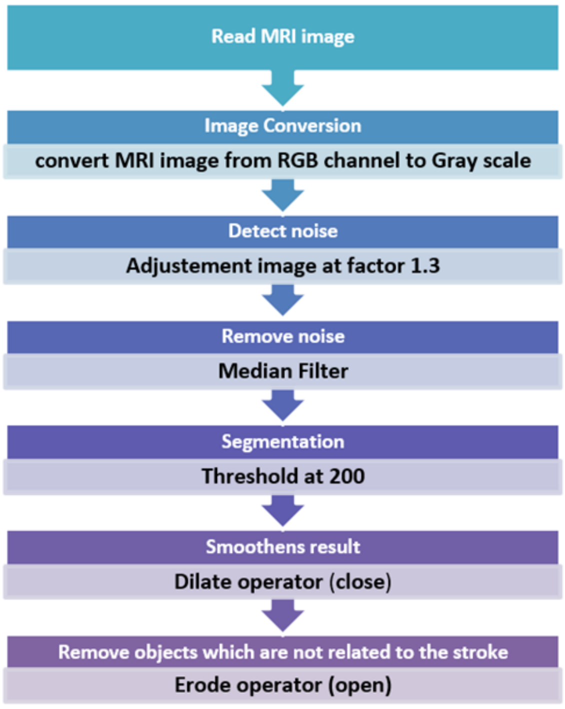
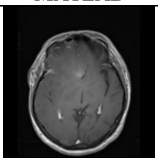
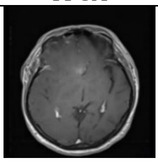
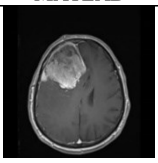
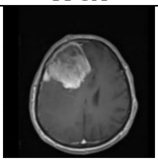
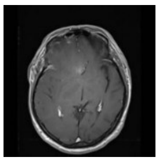
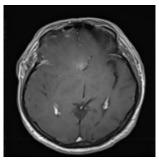
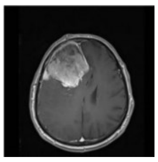
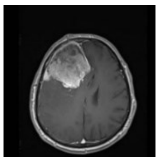
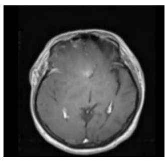
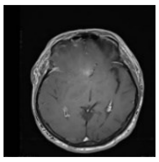
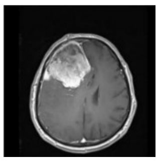
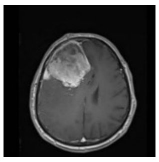
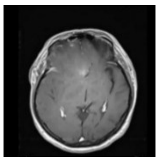
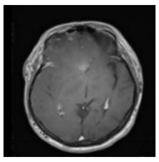
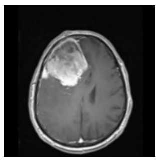
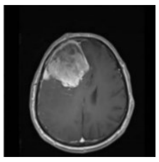
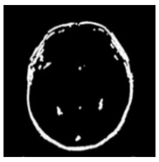
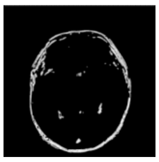
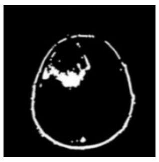
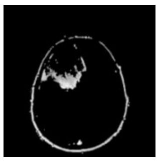
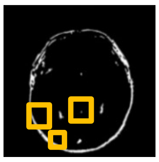
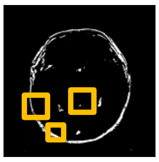
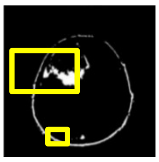
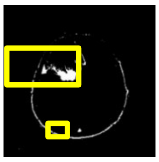
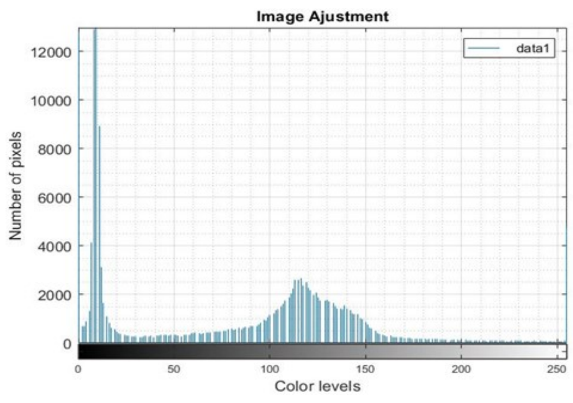


Figure 4. Functions of our proposed work

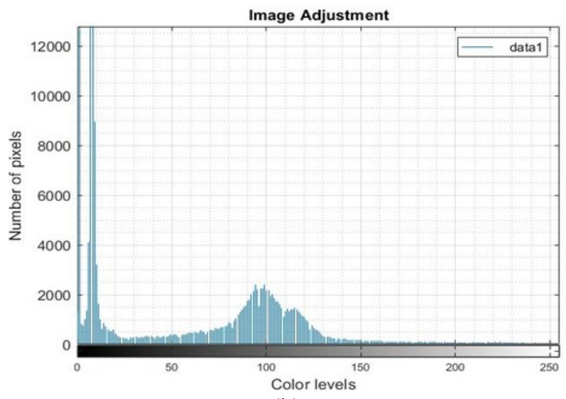
Table 1 shows that the FPGA results are more effective and accurate than Matlab as shown in step 6 figures (a), (b), (c), and d respectively). Finally, the histogram sketches in Figures 11 (a) and (b) that represent the differences between FPGA and Matlab shows more pixels detected in stroke regions. As it is clear from Figure 11, the contrast increases when the FPGA is used. Thus, the accuracy of selecting clotting location is increased compared to Matlab. The increase in contrast was the result of increase the number of pixels that have a converging value. Finally, based on Zynq ZC702 evaluation kit which is used in the proposed work, the execution time for each implemented hardware filter is calculated based on elapsed function within Vivado software development kit (SDK) tool and then compared them with the time which is takes to implement each filter by using Matlab program. Using the optimized Xilinx OpenCV function within our design help us to reduce the implemented time from 3 to 5 times (which based on the type of filter) compared with that results from Matlab as demonstrated in Figure 12.

Table 1 Results of our work between Matlab and FPGA

Steps	First MRI image		Second MRI image	
	MATLAB	FPGA	MATLAB	FPGA
1	 Figure 5 (a)	 Figure 5 (b)	 Figure 5 (c)	 Figure 5 (d)
	Accuracy 99.48 %		Accuracy 99.21 %	
2	 Figure 6 (a)	 Figure 6 (b)	 Figure 6 (c)	 Figure 6 (d)
	Accuracy 90.00 %		Accuracy 94.53 %	
3	 Figure 7 (a)	 Figure 7 (b)	 Figure 7 (c)	 Figure 7 (d)
	Accuracy 90.09 %		Accuracy 92.01%	
4	 Figure 8 (a)	 Figure 8 (b)	 Figure 8 (c)	 Figure 8 (d)
	Accuracy 91.07 %		Accuracy 90.43 %	
5	 Figure 9 (a)	 Figure 9 (b)	 Figure 9 (c)	 Figure 9 (d)
	Accuracy 92.80%		Accuracy 95.91 %	
6	 Figure 10 (a)	 Figure 10 (b)	 Figure 10 (c)	 Figure 10 (d)
	Accuracy 94.63 %		Accuracy 93.16%	



(a)



(b)

Figure 11. These figures are: (a) result of histogram by Matlab, (b) result of histogram by FPGA

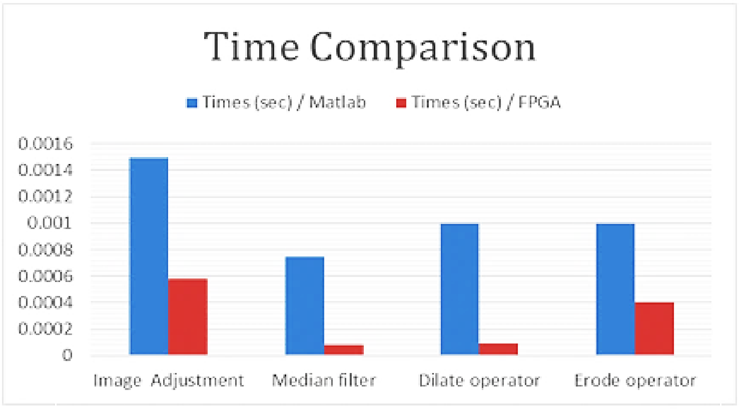


Figure 12. Time comparison between Matlab and FPGA

6. CONCLUSION

The main goal of this paper is to design and implement an efficient digital system (hardware concepts) to clarify the brain stroke in MRI image in order to improve the images and make them visible to facilitate diagnosis by doctors. The system is divided into four steps: Preprocessing, adjust image, median filter, and morphological filters alternately. On the other hand, using the OpenCV library which is combined with the FPGA-HLS tool simplify the implementation of image processing filters that are utilized in our proposed hardware system. Hardware utilization that is utilized in proposed system are computed and targeted on FPGA-Zynq evaluation kit which is equal maximum only 15%. Finally, the results are obtained by FPGA provide more accurate and smooth detection of brain stroke than Matlab's results where the accuracy ranged between 90.00% and 99.48 % when compared to matlab. In addition to reducing the execution time from 3 to 5 times compared to Matlab.

REFERENCES

[1] A. Alhawaimil, “Segmentation of Brain Stroke Image,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 9, pp. 375–378, Sep. 2015, doi: 10.17148/IJARCCE.2015.4981.

[2] D. G. Bailey, “Image processing using FPGAs,” vol. 5, no. 5, May 2019, doi: 10.3390/jimaging5050053.

[3] D. G. Bailey and A. S. Ambikumar, “Border handling for 2D transpose filter structures on an FPGA,” *J. Imaging*, vol. 4, no. 12, pp. 1-21, Nov. 2018, doi: 10.3390/jimaging4120138.

[4] R. Shi, J. S. J. Wong, and H. K. H. So, “High-throughput line buffer microarchitecture for arbitrary sized streaming image processing,” *J. Imaging*, vol. 5, no. 3, Mar. 2019, doi: 10.3390/jimaging5030034.

[5] D. Alhelal and M. Faezipour, “Denoising and beat detection of ECG signal by Using FPGA,” *Int. J. High Speed Electron. Syst.*, vol. 26, no. 3, 2017, doi: 10.1142/S012915641740016X.

[6] A. Al-Dujaili and S. A. Fahmy, “High Throughput 2D Spatial Image Filters on FPGAs,” *arXiv*, pp. 1-8, Oct. 2017.

[7] S. Badave, “Multiplierless FIR Filter Implementation on FPGA,” *Int. J. Inf. Electron. Eng.*, vol. 2, no. 2, pp. 185-188, Jan. 2012, doi: 10.7763/ijiee.2012.v2.78.

[8] L. S. DeBrunner, “Reducing complexity of FIR filter implementations for low power applications,” *Conf. Rec.-Asilomar Conf. Signals, Syst. Comput*, 2007, pp. 1407-1411, doi: 10.1109/ACSSC.2007.4487460.

[9] J. Park, K. Muhammad, and K. Roy, “High-performance FIR filter design based on sharing multiplication,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 2, pp. 244-253, Jul. 2003, doi: 10.1109/TVLSI.2002.800529.

[10] P. Bougas, P. Kalivas, A. Tsirikos, and K. Z. Pekmestzi, “Pipelined Array-Based FIR Filter Folding,” vol. 52, no. 1, pp. 108-118, Jan. 2005, doi: 10.1109/TCSI.2004.838542.

[11] S. Mirzaei, A. Hosangadi, and R. Kastner, “FPGA implementation of high speed FIR filters using add and shift method,” *IEEE Int. Conf. Comput. Des. ICCD*, 2006, pp. 308-313, doi: 10.1109/ICCD.2006.4380833.

[12] Z. Tang, J. Zhang, and H. Min, “A high-speed, programmable, CSD coefficient FIR filter,” *IEEE Trans. Consum. Electron.*, vol. 48, no. 4, pp. 834-837, Dec. 2002, doi: 10.1109/TCE.2003.1196409.

[13] J. B. Evans, “Efficient FIR Filter Architectures Suitable for FPGA Implementation,” *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 41, no. 7, pp. 490-493, Jul. 1994, doi: 10.1109/82.298385.

[14] S. S. Jeng, H. C. Lin, and S. M. Chang, “FPGA implementation of FIR filter using M-bit parallel distributed arithmetic,” *Proc.-IEEE Int. Symp. Circuits Syst.*, no. 5, pp. 875-878, Jun. 2006, doi: 10.1109/iscas.2006.1692725.

[15] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, “Novel FPGA-based implementation of median and weighted median filters for image processing,” *Proc.-2005 Int. Conf. F. Program. Log. Appl. FPL*, Sep. 2005, doi: 10.1109/FPL.2005.1515713.

[16] A. Eric, “FPGA Implementation of Median Filter using an Improved Algorithm for Image Processing,” *International Journal for Innovative Research in Science and Technology,* vol. 1, no. 12, pp. 25-30, 2015.

[17] G. L. Bates and S. Nooshabadi, “FPGA implementation of a median filter,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2, pp. 437-440, Jan. 1997, doi: 10.1109/tencon.1997.648210.

[18] S. S. Tavse, P. M. Jadhav, and M. R. Ingle, “Optimized Median Filter Implementation on FPGA Including Soft Processor,” *Int. J. Emerg.*, vol. 2, no. 8, pp. 236-239, Aug. 2012.

[19] K. S. Raju, P. Phukan, and G. Baurah, “An FPGA Implementation of a Fast 2-Dimensional Median Filter,” *National Conference on Recent Advances in Communication, Control and Computing Technology-RACCCT 2012*, pp. 144-148, Mar. 2012.

[20] G. Poshamallu, “Binary Image Processing Implementation on FPGA Using Morphological Dilation and Erosion Techniques,” *International Journal of Engineering Science and Computing*, vol. 6, no. 4, pp. 4280-4283, Apr. 2016, doi: 10.4010/2016.982.

[21] D. Mukherjee, S. Mukhopadhyay, and G. P. Biswas, “FPGA-Based Parallel Implementation of Morphological Operators or 2D Gray-Level Images,” *Arab. J. Sci. Eng.*, vol. 42, no. 8, pp. 3191-3206, 2017, doi: 10.1007/s13369-017-2429-y.

[22] R. Arunmozhi and G. Mohan, “Implementation of Digital Image Morphological Algorithm on FPGA using Hardware Description Languages,” *Int. J. Comput. Appl.*, vol. 57, no. 5, pp. 1-4, 2012.

[23] R. R. K. R. C. Gampa, “Implementation of Four Morphological Operators for Image Filtering on Fpga,” *Int. J. Sci. Res.*, vol. 2, no. 5, pp. 112-114, 2013.

[24] R. Hentati, M. Hentati, Y. Aoudni, and M. Abid, “The implementation of basic morphological operations on FPGA using partial reconfiguration,” *Int. Image Process. Appl. Syst. Conf. IPAS*, 2014, doi: 10.1109/IPAS.2014.7043260.

[25] D. Mukherjee, S. Mukhopadhyay, and G. P. Biswas, “FPGA based parallel implementation of morphological filters,” *Int. Conf. Microelectron. Comput. Commun. MicroCom*, 2016, doi: 10.1109/MicroCom.2016.7522488.

[26] G. B. Deshpande and D. K. Ramesha, “MRI Brain Image Enhancement Using XILINX System Generator and DWT,” *Bonfring Int. J. Adv. Image Process.*, vol. 5, no. 2, pp. 16-22, May 2015, doi: 10.9756/bijaip.8039.

[27] M. J. Fadhil, R. A. Fayadh, and M. K. Wali, “Design and implementation a prototype system for fusion image by using SWT-PCA algorithm with FPGA technique,” *Int. J. Electr. Comput. Eng.*, vol. 10, no. 1, pp. 757-766, 2020, doi: 10.11591/ijece.v10i1.pp757-766.

[28] S. Allin Christe, M.Vignesh, and A.Kandaswamy, “An Efficient Fpga Implementation Of Mri Image Filtering And Tumour Characterization Using Xilinx System Generator”, vol. 2, no. 4, pp. 95-109, 2011, doi: 10.5121/vlsic.2011.2409.

[29] V. Sivakumar and N. Janakiraman, "A novel method for segmenting brain tumor using modified watershed algorithm in MRI image with FPGA," *BioSystems*, vol. 198, no. April, pp. 104226, Dec. 2020, doi: 10.1016/j.biosystems.2020.104226.
[30] Xilinx Inc., "Zynq-7000 All Programmable SoC Data Sheet Overview," DS190 (v1.11) June 7, 2017.

BIOGRAPHIES OF AUTHORS

Dheyaa Alhelal completed his B.Sc. Technical Computer Engineering from Technical Engineering College Northern Technical University, Mosul, Iraq in 2006. Completed his M.Sc. from University Of Bridgeport, Bridgeport, Connecticut, USA with thesis name "Denoising and Beat Detection of ECG Signal By Using FPGA" in 2014. In Currently, he is a Assistance lecturer of Technical Computer Engineering from Technical Engineering College Northern Technical University,Mosul,Iraq. he has three international papers. His research interests include FPGA, Signal processing, Wireless Sensor Network and Image processing.

Ahmed Khazal Younis works at Northern Technical University/Engineering Technical College-Mosul-Iraq/Department of Computer Technology Engineering.He Completed his undergraduate education (1998-2002), master degree (2005-2008) at Computer Technical Engineering Department in Technical College / North Technical University and Ph.D. degree in computer engineering/ embedded system design from Yasar University in Izmir/Turkey in 2019. His research interests include: Neural networks, Neuro-fuzzy system, Microcontrollers, and FPGA devices.

Ruaa H. Al-Mallah completed her B.Sc. Technical Computer Engineering from Technical Engineering College Northern Technical University, Mosul, Iraq in 2005. Completed her M.Sc. from the same college with thesis name " Two image processing for Robot Vision" in 2008. In Currently, she is a Assistance lecturer of Technical Computer Engineering from Technical Engineering College Northern Technical University,Mosul,Iraq. she has One international paper and Two national papers. Her research interests include Robot Vision, Image processing and Computer Graphic .