




Optimized Algorithms and Hardware Implementation of Median Filter for Image Processing

H. H. Draz^{1,2}  · N. E. Elashker^{1,3} · Mervat M. A. Mahmoud^{1,3}

Received: 31 May 2022 / Revised: 23 March 2023 / Accepted: 24 March 2023

© The Author(s) 2023

Abstract

Image processing algorithms are essential for clarifying the image and improving the ability to recognize distinct characteristics of the image. The field of digital image processing is widespread in several research and technology applications. In many of these applications, the existence of impulsive noise in the obtained images is one of the most frequent problems. The median filter is a strong method to remove the impulsive noise; it effectively eliminates salt and pepper noise from the image. The main target of this paper is to investigate efficient median filter units to be connected to a general-purpose processor (GPP) for FPGA-based embedded systems. The paper exposes three novel techniques, two of them specially for median filtering techniques and the third one is used to get the maximum number of any 9 elements array. The proposed algorithms are inspired by the Median Of Median (MOM) algorithm. The first two techniques are tested for filtering 3×3 image windows and optimized for producing the expected result in high accuracy, short time, and reduced number of comparisons. The last technique is tested for a 9 elements array for extracting the maximum number in same high efficiency manner. Furthermore, the three proposed techniques are implemented leveraging the advantage of the parallel processing and the

H. H. Draz: The authors contributed equally to this work

✉ H. H. Draz
hdraz@eri.sci.eg

N. E. Elashker
nahlaelazab@eri.sci.eg

Mervat M. A. Mahmoud
mervat-m@eri.sci.eg

¹ Microelectronics Department, Electronics Research Institute, Cairo, Egypt

² Adjunct Assistant Professor at the Communications and Electronics Department, Faculty of Engineering, Cairo University, Cairo, Egypt

³ Adjunct Assistant Professor at the Electrical Engineering Department, The British University in Egypt, Cairo, Egypt

FPGA flexible resources to satisfy the real-time processing constraints. A comparison between the first two proposed filtering units and their counterparts in the literature is included. The comparison reveals the superiority of the first technique in terms of accuracy with fewer comparators than previously published techniques. Besides, the paper illustrates how the concept beyond the proposed techniques can be used to perform the maximum pooling for convolution neural networks.

Keywords Embedded systems · FPGA · Image de-noising · Median filter · Median of medians

1 Introduction

Digital images have a significant role in daily life applications and areas of research and technology such as magnetic resonance imaging, satellite television, geographical information systems, and astronomy [18]. They also act a vital role in some medical applications such as diabetic retinopathy diseases [17]. Datasets gathered by image sensors, the circuitry of a scanner or digital camera are generally infected by noise. It can originate from the image sensor, the film grain, or the ideal photon detector's obligatory shot noise. In addition to the problems with the data acquisition process and imperfect instruments, transmission errors, interfering natural phenomena, and compression can also be sources for image noise [15]. This noise can be recognized as a fundamental signal distortion that degrades the process of information extraction and image observation. Eliminating image noise (denoising) forms a necessary basis for image processing and analysis. Image denoising has been exhaustively studied in the area of computer science. Therefore, many filtration algorithms exist. Denoising algorithms include transform-domain thresholding, statistical models, random fields, dictionary learning methods, anisotropic diffusion methods, spatial domain filtering, and hybrid methods [26].

Regarding the spatial domain techniques, the similarities between pixels or patches of an image are exploited. Spatial domain methods comprise local and non-local filters. A non-local filter makes use of the correlation between the complete range of image pixels. Local filters are limited by spatial distance. Moreover, the noise reduction schemes in local filtering are classified into linear filtering and nonlinear filtering techniques. These linear filters depend on convolving the image matrix with a filter mask to produce a linear extension of neighborhood values. This kind of spatial filtering is the most accessible and most primitive form of noise removal, but it frequently causes an undesirable quantity of edges smoothing, loss of details, and poor feature localization [9, 14, 23]. Hereafter, a variety of nonlinear filters have been introduced in order to focus on the aforementioned restrictions in a manner of their innovative and improvizations concepts [3, 10].

The median filter is one of the nonlinear filters based on the order-statistics theory [7] and is used for impulse noise removal. Impulse noise for image frames is also expressed as salt and pepper noise. This filter substitutes each pixel by the median of the surrounding pixels. The median value must essentially be one of the neighborhood pixels values. As a result, separated noise points will be removed without generating

new impracticable pixel values. Therefore, this technology is preferred for eliminating the random additive and the salt-and-pepper noise. It can also keep the high-frequency portion of the image to a great extent, thus ensures the image visual effect. Consequently, median filters can present good noise reduction facilities, significantly less blurring than smoothing linear filters of equal size. For these reasons, the median filtering approach has become a widespread algorithm of the image repairing field in recent years [18].

This technique's methodology comprises too much manipulation involving the scan, compare and move operation, making it relatively slow [1]. Standard median filtering cannot comply with the real-time requirement. Additionally, software implementations of sorting procedures that are needed for standard median filter are time-consuming. Thus, optimized sorting techniques should be used for fast median filtering. Also, it is critical to implement the filter in a high-speed environment.

Therefore, two filtering techniques are proposed, which do not depend on sorting all the elements but on making some comparisons to exclude the nonmedian elements. This concept prevents unnecessary sorting and comparisons for satisfying the real-time processing constraints. On the other hand, the implementation of 3×3 (9 elements window) median filters has been motivated, given that larger filters usually reduce small edges [20]. Furthermore, hardware acceleration is utilized in this work using Field Programmable Gate Arrays (FPGA) to improve the performance. Since FPGAs usually offer a substantial speed improvement compared to software-based solutions via adaption to the application and parallelism features. Hence, the two proposed filtering units benefit from the merits of the parallel processing and FPGA flexible resources. The rest of the paper is organized as follows: Sect. 2 describes the background of the median filter techniques, including the sorting-based techniques and the selection-based techniques. The first proposed algorithm, the second proposed one, and the exploitation of the proposed concept for finding out the array maximum are demonstrated in detail in Sects. 3, 4, and 5, respectively. While the simulation results are illustrated in Sect. 6. Finally, the paper is concluded in Sect. 7.

2 Background of Median Filter Algorithms

The median value is the value in the mid-position of an arranged sequence. Assume that the window pixel values are put into an array named x , such that its elements are $x_1, x_2, x_3, x_4, \dots, x_n$ and they turn into $x_{i1} \leq x_{i2} \leq x_{i3} \leq x_{i4} \dots \leq x_{in}$ after sorting them in descending order, afterward their median value is $x_{i(n-1)/2}$. There are many techniques to get the median number of an array; some of them depend on sorting the array elements first then selecting the element of the mid position, and the others are done without sorting. The sorting-based techniques are more popular than the techniques that belong to the other category. Therefore, the literature includes many implementations of median filters using different sorting algorithms such as the bubble, insertion, selection, quick, heap, shell, merge, and timsort [2]. In these implementations, the effectiveness of these median filters depends mostly on the adopted sorting techniques' speed and efficiency. However, even with the high-efficiency sorting techniques, there is a waste in the used hardware and the computational time wherein

sorting all the elements is not actually required. Additionally, some of the sorting techniques, i.e., merge sorting and odd-even merge-sort, restrict the array length to be a power of 2 which means appending the window array by some zeros (or some other values), and after applying the sorting algorithm, those appended zeros should be eliminated [22]. Hence, in the case of 3×3 windows, there should be seven appended elements to satisfy this condition which incorporates undesired overhead in the used memory, the required hardware, and processing time. Therefore, these techniques are not suitable to be a base for a median filter. Thus, there is a need to get the median without sorting the entire array. Hence, other techniques with different conceptions have been arisen, which do not wholly sort the array. These techniques are divided into the quick selection algorithm and the median of median algorithm according to their main idea of thinking.

2.1 Quick Selection Algorithm

Quickselect is a selection technique to locate the k th smallest element in an unsorted group. It is interrelated to the quicksort sorting technique. Similar to quicksort, it was established by Tony Hoare, and as a result, is also recognized as Hoare's selection algorithm [13]. Like quicksort, it is effective in execution and has satisfactory average-case performance, although it has weak worst-case performance. Quickselect and its versions are the most frequently used selection algorithm in productive real-world implementations. It utilizes the same global methodology as quicksort, which considers one element as a pivot. Then separating the data into two groups, depending on the pivot, one for the elements that less than the pivot and the other for the elements which are greater than it. As a result of comparing the pivot to all other elements, the rank ' r ' of the pivot is found out. The pivot is then inserted in the r th index of the array. All other elements, equal to or smaller than the pivot, are located into positions before the pivot. Moreover, every larger element is placed after the pivot [18]. However, instead of repeating into both sides, like in quicksort, quickselect only reiterates into one side, which contains the element of interest. Like quicksort algorithm, quickselect is in general realized as an in-place technique. At the same time as selecting the k th element, it partially arranges the data. This degrades the order of the average complexity from $O(n \log n)$ to $O(n)$. Nevertheless, the worst-case situation occurs when choosing the largest or the smallest element as a pivot. Consequently, each step would eliminate just one element from the list, and the complexity will be $O(n^2)$ rather than $O(n)$. Therefore, this algorithm is a practical randomized linear algorithm for median finding. However, it is insufficient to have only a randomized algorithm where sometimes it is more critical to be accurate than to be fast. So the median of the median algorithm has been developed, which is a linear worst-case time algorithm.

2.2 Median of Median (MOM) Algorithm

The median-of-medians is a predictable linear-time selection technique. It was published, for the first time, in [5], and accordingly is named as "BFPRT" after the last names of the authors. The MOM algorithm relies on partitioning the list into some

sublists and then getting the estimated median place in each sublist. Next, all medians are collected and arranged into a new list, then finding out the list's median location. It considers the value of the median location as a pivot then utilizes it for comparisons with the other elements of the list. If an element is smaller than the pivot value, the element is placed on the pivot's left side, but if the element has a value greater than the pivot, it is put to the right. The algorithm is repeated on the list focusing on the value it is searching for. Despite the worthy average performance, the algorithm in the worst-case pivot condition is insufficient for assurance exact selection in linear time [1]. Therefore, some modifications are developed on this algorithm to overcome this problem, like median 3, median 4, and median 5. Afterward, an improvement was accomplished on MOM that is recognized as REPEATEDSTEP. However, the number of comparisons for a nine elements list is still large, which can be calculated using Eq. 1 [1].

$$C(n) \leq C(n/5) + C(7n/10) + 2n. \quad (1)$$

where n is the length of the list, another modification is applied to the modified version. The new version is known by REPEATEDSTEPIMPROVED, then again it gives a large number of comparisons that can be expressed as in Eq. 2 [1].

$$C(n) \leq C(n/9) + C(7n/9) + n + n/3 + 8n/9 \quad (2)$$

For the all aforementioned modifications, the number of comparisons for calculating the median of an array with nine numbers is larger than $5n$ where $n = 9$, which is considered an enormous number that needs relatively massive hardware and long processing time. In this context, the median of median algorithm is selected as a base to the suggested techniques. Thereafter, some modifications have been applied to it to ensure getting the median value with the highest accuracy and least number of comparators.

Consequently, this paper proposes two novel designs to select the median of nine elements list, referred to as Median 9, inspired by the MOM algorithm. The proposed techniques are based on a refined explanation of MOM, and it is partly inspired by the odd-even merge sort [21]. However, they have a lower order of complexity than the odd-even merge sort technique. The first proposed algorithm (ALG1) guarantees an exact selection in linear time. Moreover, it dramatically reduces the number of comparators. This technique provides an exact median of 9 elements array with only 17 comparators which is less than $2n$. Subsequently, it reduces the needed hardware resources for its implementation. Aside from the precise detection, another version of the first algorithm is proposed and introduced as ALG2 technique. In this algorithm, the hardware requirements are reduced with improvements in the processing time at the expense of a slight reduction of the accuracy. Thus, it represents a different option, and the preference depends on the application obligations. Also, a third algorithm (ALG3) is suggested for generating the maximum of nine elements, as it is needed in some applications. Besides, all the proposed techniques do not restrict the array length to be of a power of 2.

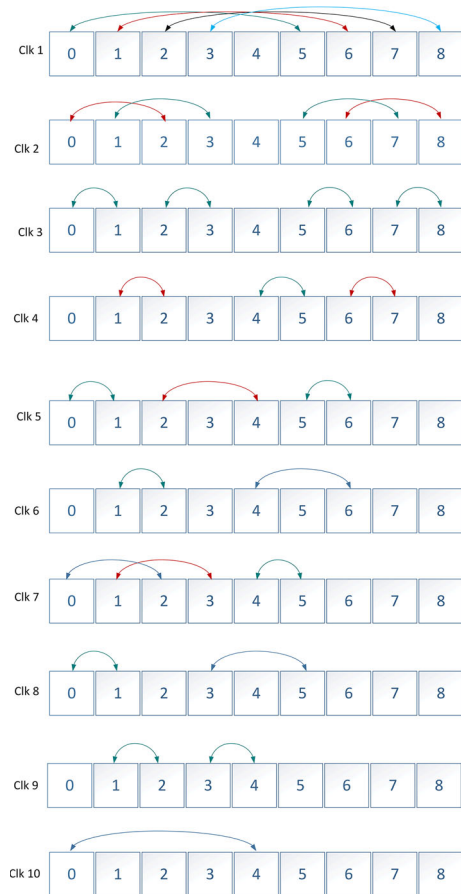
Furthermore, the FPGA implementation environment is used as a hardware accelerator to implement the proposed techniques for satisfying higher processing speed.

Since FPGAs provide inherent parallelism. The proposed algorithms will be illustrated in detail in the following sections.

3 The First (Accurate) Algorithm (ALG1)

Assuming that the pixels array, p , contains the values of all the pixels in the selected 3×3 window. The pixels are represented by 8-bit numbers and located in the indices 0, 1, 2, 3, 4, 5, 6, 7, 8. The proposed algorithm which aims to extract the median pixel relies on rearranging the array to ensure that the numbers in the last four locations are greater than the numbers in the first five locations. So in the first step, the numbers at locations 0, 1, 2, 3 are compared with their counterparts at locations 5, 6, 7, 8 respectively. Swapping processes are performed, if needed, to guarantee that $(P[0] <$

Fig. 1 The theory behind ALG1 showing the executed procedure in each clock. The comparisons in each clock cycle with swapping steps for ALG1 algorithm as software idea



$P[5]$), $(P[1] < P[6])$, $(P[2] < P[7])$, and $(P[3] < P[8])$. Secondly, the same concept is used with the array tailing half by performing the following comparisons $(P[5] < P[7])$, $(P[6] < P[8])$ with swapping if these conditions are not initially satisfied. Furthermore, if the swapping between any couple of these locations is done, another swapping should be done to keep the validity of the reached relations. Such that, if swapping between $P[6]$ and $P[8]$ is occurred then swapping between $P[0]$ and $P[2]$ should be performed regardless their arrangement. Similarly, for the case of $P[1]$ and $P[3]$ if $P[5]$ and $P[7]$ are swapped. Thirdly, a comparison between $P[7]$ and $P[8]$ is executed, hence $P[8]$ is undoubtedly greater than numbers at locations 0, 1, 2, 3, 5, 6, 7. So $P[8]$ will be excluded from the subsequent comparisons. However, in this step, if the condition $(P[8] < P[7])$ is verified, three swapping should be made (0 with 1, 2 with 3, and 5 with 6). These swapping processes are necessary to keep the reached relations between the elements valid. In the fourth step, two comparisons would be done. One of them is between the elements of indices 6 and 7 to throw the element of index 7 from the comparisons after ensuring that element number 7 is more significant than 0, 1, 2, 3, 5, 6. Another comparison is needed to include the element of index 4 in the sequence of comparisons, and it will be between $P[3]$ and $P[4]$. The following steps are processed to let $P[4]$ larger than its left side elements and less than its right side elements. After ten cycles, the median will be stored in $P[4]$. The detailed sequence of comparisons for this technique is illustrated in Fig. 1.

This algorithm includes seventeen comparisons which are executed in ten clock cycles, and it detects the median number of a nine elements array with accuracy of 100%. Some modifications can be applied to this technique in order to reduce the required number of clock cycles. However, in all of these versions, the accuracy will be decreased too. The hardware architecture of this technique is shown in Fig. 2.

The time simulation and the results of applying the ALG1 algorithm on three noisy images are portrayed in Figs. 3 and 4, respectively. One can deduce that the salt and pepper noise is significantly eliminated.

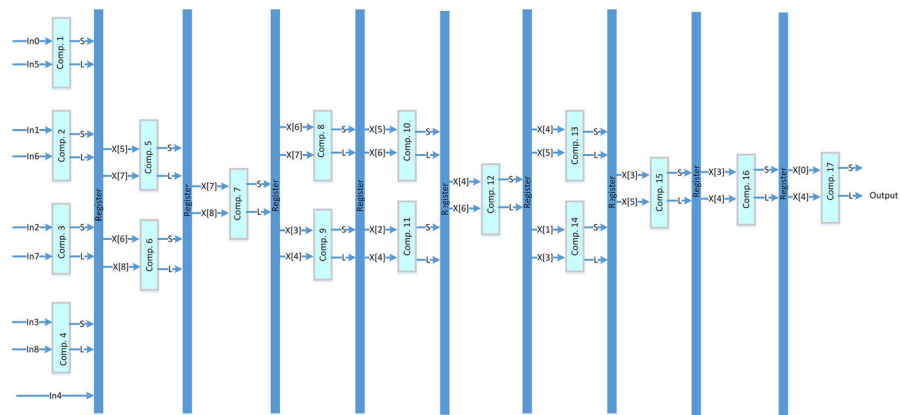


Fig. 2 The hardware implementation of the ALG1 algorithm

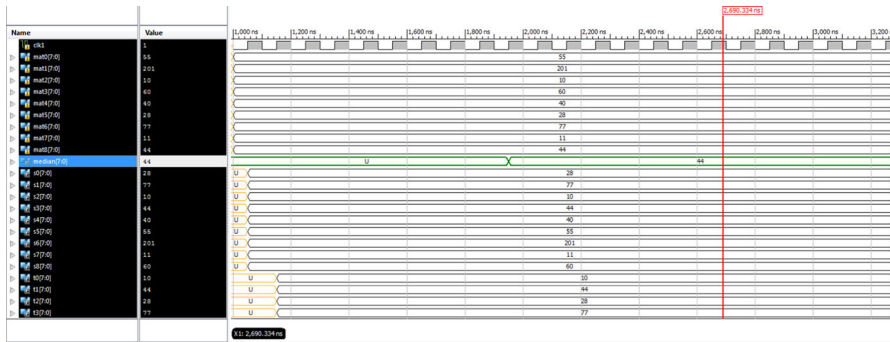
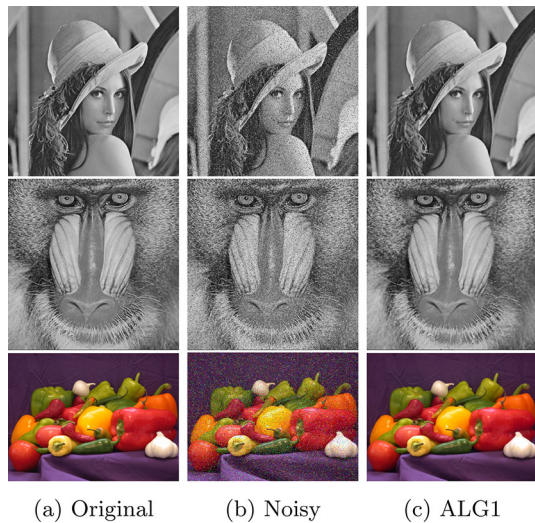


Fig. 3 The time simulation of ALG1

Fig. 4 a input images, b 10% salt-and pepper noisy image, c The noisy images after applying ALG1 algorithm



(a) Original (b) Noisy (c) ALG1

4 The Second Proposed Algorithm (ALG2)

Another technique is suggested in the march toward more reduction of the total required comparisons and the execution clock cycles. This technique is firmly grounded in fulfilling one hypothesis, which is that $P[7]$ and $P[8]$, after the third clock cycle, are greater than the array median number. By running extensive numerical simulations under various examples, the probability of this assumption’s validity is greater than 97%. The Hardware implementation, that clarifies the progression of the comparison and the parallel approach steps, is depicted in Fig. 5.

In this technique, the first and second steps are the same as those of the first technique. Depending on the above assumption, the elements at $P[7]$ and $P[8]$ are kept out from the following comparisons. In the third step, a comparison between $P[5]$ and $P[6]$ is done to determine which element will need only one comparison and then be thrown out of the following comparisons. Also, in the third clock cycle, another comparison between $P[2]$ and $P[4]$ is executed, where one of the advantages of the

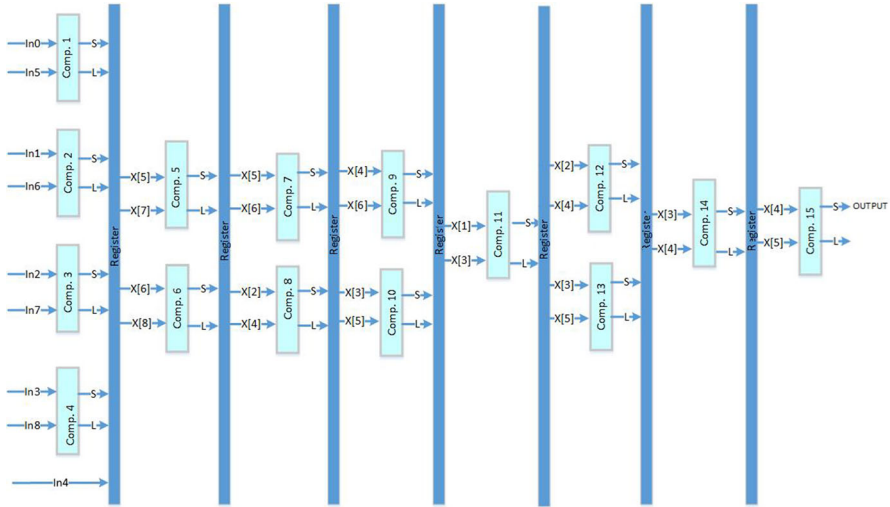


Fig. 5 The hardware implementation of the ALG2 algorithm

hardware implementation is that multi-comparisons between different stores and the index modifications are performed simultaneously without any conflict in memory. In the light of the previous postulation, step four includes a comparison between $P[4]$ and $P[6]$, then rearranging the elements in the following indices 4, 5, and 6. Where $P[5]$ in place of $P[4]$, $P[6]$ in place of $P[5]$, and finally $P[4]$ in place of $P[6]$. Whereupon, there will be six elements only for the remaining comparisons, and the comparisons will continue until grantee that $P[4]$ is larger than all elements except $P[5]$. In step five, a comparison between elements of indices 3 and 5 is executed. Next in step six, a comparison is done between indices 1 and 3. Accordingly, if the condition is true, swapping will be between indices 1 and 3. Otherwise, swapping will be between indices 2 and 3. In the seventh step, two comparisons are needed: the first is between indices 2 and 4, and the other is between indices 3 and 5. The last step includes comparing indices 3 and 4, so if the condition ($P[3] < P[4]$) is satisfied, the median is at index four; otherwise, another comparison will be executed between indices 4 and 5. As a result, the median is at index 4. The error may occur when the first assumption does not fulfill.

The time simulation and the result of applying this technique on three different graphs is presented in Figs. 6 and 7, respectively. The figure includes the original, noisy, and denoisy plots. This technique can deduce the median number after nine clock cycles using only fifteen comparisons with accuracy equals to 87%. On the other hand, in exploiting this technique for noisy image frames, the accuracy is improved to 96%.

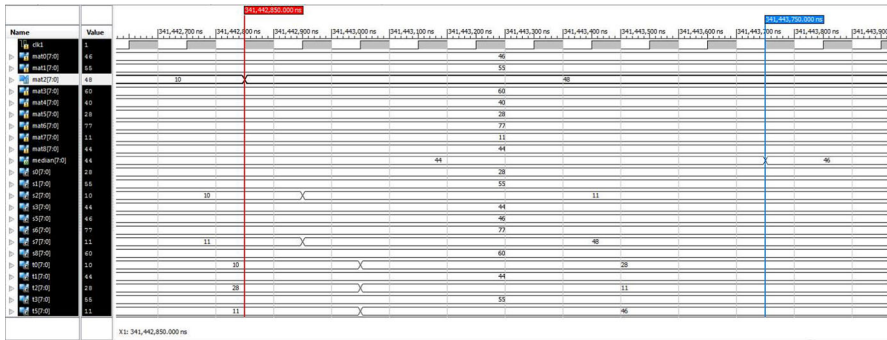


Fig. 6 The time simulation of ALG2

Fig. 7 **a** Input images, **b** 10% salt and pepper noisy image, **(c)** The noisy images after applying ALG2 algorithm



(a) Original

(b) Noisy

(c) ALG2

5 The Third Proposed Algorithm (ALG3)

In many modern applications, such as max-pooling in convolution neural networks, there is a need to obtain the maximum of nine elements array [12]. Therefore, ALG3 is proposed for generating the maximum of an array of nine elements. The same concept of the first proposed technique can be used but with limiting the swapping processes to the compared elements only. The earliest three steps will be used, and after those, there is a grantee that index 8 is greater than indices 0, 1, 2, 3, 5, 6, 7 so that the last step will be a comparison between $P[4]$ and $P[8]$. As a result, the max number of a nine elements array will be determined after four clocks with eight comparisons only. The proposed algorithm is optimized for any array size, it does not assume that the size of the array is an even number or a power-of-2 integer. It can discover the maximum of n elements using just $O(n)$ comparisons. In return, the selection sort locates the most elements and positions them where they should be. To sort n elements, $n(n-1)/2$ comparisons must be made [11]. Also, $3(n/2)$ comparisons are required to discover the minimum

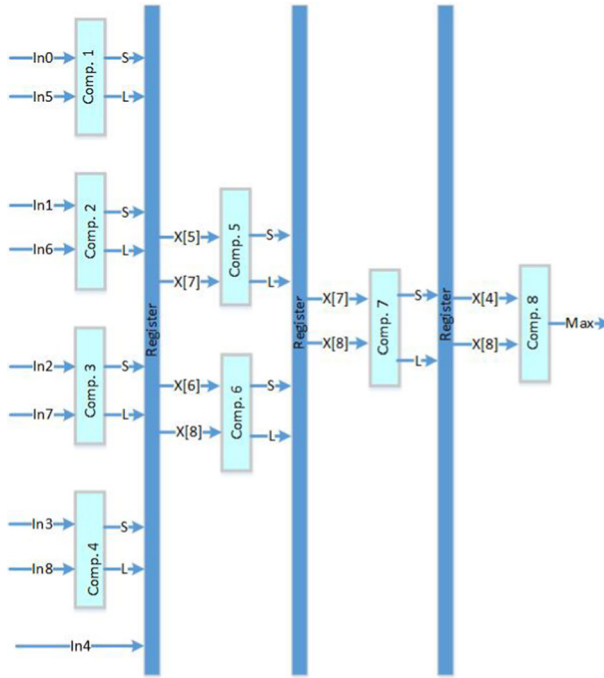


Fig. 8 The hardware implementation of the ALG3 algorithm



Fig. 9 The time simulation of ALG3 algorithm

and maximum in [6], which proves the superiority of the proposed algorithm. The hardware implementation of the Maximum algorithm is exposed in Fig. 8. The time simulation of this technique is shown in Fig. 9. This provides an improvement in the processing time to equal 10ns that is calculated based on the clock frequency and the total number of required clocks which are 400MHz and 4 clocks, respectively. On other hand to get the maximum number using the common sorting algorithms, it requires at least $O(n \log n)$ comparisons in the best case and $O(n^2)$ comparisons in the worst case for most of the outputs, which means 36 comparators at least.

6 Simulations and Results

The proposed algorithms were implemented using the Very High-Speed Integrated Circuits Hardware Description Language (VHDL) and simulated using Xilinx ISE 12.2, Xilinx EDK 12.2, and Modelsim 6.6c tools with hardware experiment's environment Xilinx Virtex-5 LX110T board. The FPGA implementation environment provides an attractive combination of apparent flexibility, high performance, and low-cost properties. Whereas, the MATLAB environment is utilized to verify the algorithms and evaluate the quality metrics between the noisy and the extracted images.

The structures of the proposed techniques mainly comprise comparator blocks and register after each stage as shown in Figs. 2, 5, and 8 which support the pipelining feature. The suggested architecture for the median filter depends on a sequence of pipelined stages to reduce the computational time. Hence, any algorithm's hardware architecture is used as a building block, identified as "median 9 block," in the median filter architecture. A detailed comparison between the ALG1, ALG2, and the designs published in [18] and in [19] is illustrated in Table 1. This comparison considers the following criteria; the number of comparators, maximum frequency, accuracy, execution time which represents the algorithm processing time, throughput, number of slice registers, number of slice lookup tables (LUT), signal to Noise Ratio (SNR), peak SNR, and Structural Similarity Index Measure (SSIM). From the simulation results, it can be deduced that our proposed filter, using the first algorithm, requires ten clock pulses with 398.2 MHz maximum operating frequency to find out the median value with 100% accuracy. This provides an improvement in the processing time than the filter proposed in [18], with approximately maintaining same image quality which is measured by the following metrics SNR, peak SNR, and SSIM. Although the technique

Table 1 A comparison among the different algorithms considering the computation complexity and the quality of denoised images

	Proposed ALG1	Proposed ALG2	[18]	[19]
Accuracy	100%	87%	93%	93%
Comparators	17	15	19	19
Frequency (MHz)	389.2	400.1	100	N/A
Clocks	10	9	3	N/A
Execution time (ns)	25.11	22.47	30	76.21
Throughput (<i>bp/μs</i>)	2866.32	3200	2400	N/A
Number of slice registers	368	240	N/A	73
Number of slice LUTs	465	314	N/A	728
Peak SNR	35.73	35.28	35.4	35.4
SNR	27.51	27.11	27.13	27.13
SSIM	0.9891	0.9881	0.9880	0.9880
Complexity in terms of Processing time (<i>μs</i>)	168.38	163.79	N/A	N/A

N/A = not available

proposed in [18] needs only three clock pulses, it operates at 100 MHz maximum operating frequency, which means slower performance. Moreover, the accuracy of median numbers calculating is enhanced by 7% and the execution time is reduced by more than 16%. Besides, by comparing the first proposed technique with the designs investigated in [8, 16, 17, 19, 25], there is more than 10% improvement in the hardware of each median 9 block because it needs only seventeen compactors instead of nineteen. However, with respect to ALG2, it has the minimum execution time and highest throughput with maintaining almost equal SNR, peak SNR, and SSIM with small reduction in accuracy compared with [18] and [19]. It is worthy to mention that the reduced computation complexity of the proposed algorithms and their small area makes them good candidate to be used as basic units in multi-stages algorithms. Since, most of these algorithms utilizes the median of 3×3 window as a pre-processing filtering stage [4] and [24].

7 Conclusion

The median of Medians is the mathematical concept that our algorithm is based on. In the state-of-the-art, this concept is utilized for designing a median-3 by extracting the median of three elements and a median-5 for five elements as a software solutions. The median-3 was used as a basic unit to implement the median of 9 elements array and provide hardware designs in the literature. Besides, our technique is a novel that implement the median-9 as one unit. Two algorithms for detecting the median element of 9 numbers, ALG 1 and ALG2, are suggested and implemented using the FPGA environment. The algorithms are tested using gray-scale and colored images. The comparison considered the Hardware complexity in terms of the number of comparators. Then, comparisons between the noisy and filtered images are carried out using the SSIM, SNR, and peak SNR as quality metrics. The hardware results show that the proposed algorithms are superiors while comprehensive output results compared to standard median techniques. ALG1 satisfies more accuracy than ALG2 but ALG2 has smaller area and is faster than ALG2. The same concept is exploited in ALG3 to extract the array maximum with minimum hardware, high speed and 100% accuracy. Subsequently, they have several prospective applications in real-time image processing and meet the different applications' requirements.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. A. Alexandrescu, Fast deterministic selection, in *Leibniz International Proceedings in Informatics (LIPIcs)* (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik), pp. 24:1–24:19
2. Y. Ben Jmaa, R. Ben Atitallah, D. Duvivier, M. Ben Jemaa, A comparative study of sorting algorithms with FPGA acceleration by high level synthesis. *Computación y Sistemas* **23**(1), 213 (2019)
3. J.L. Bentley, M.D. McIlroy, Engineering a sort function. *Softw Pract Exp* **23**(11), 1249–1265 (1993)
4. N. Bindal, B. Garg, Novel three stage range sensitive filter for denoising high density salt and pepper noise. *Multimedia Tools Appl.* 1–16 (2022)
5. M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection. *J. Comput. Syst. Sci.* **7**(4), 448–461 (1973)
6. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 4th edn. (MIT Press, Cambridge, 2022)
7. H.A. David, H.N. Nagaraja, Order statistics, in *Encyclopedia of Statistical Sciences* (2004)
8. A. Eric, FPGA implementation of median filter using an improved algorithm for image processing. *Int. J. Innov. Res. Sci. Technol.* **1**(12), 25–30 (2015)
9. R. Gonzalez, R. Woods, Image processing. *Digit. Image Process* **2**, 1 (2007)
10. B. Goyal, A. Dogra, S. Agrawal, B. Sohi, A. Sharma, Image denoising review: From classical to state-of-the-art approaches. *Inf. Fusion* **55**, 220–244 (2020)
11. M. Goyani, M. Chharchhodawala, B. Mendapara, Min-max selection sort algorithm–improved version of selection sort. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **6** (2013)
12. B. Graham, Fractional max-pooling. *Comput. Vis. Pattern Recognit.* (2014)
13. H.-K. Hwang, T.-H. Tsai, Quickselect and the dickman function. *Comb. Probab. Comput.* **11**(4), 353–371 (2002)
14. M. Lebrun, M. Colom, A. Buades, J.-M. Morel, Secrets of image denoising cuisine. *Acta Numer* **21**, 475 (2012)
15. L. Liang, S. Deng, L. Gueguen, M. Wei, X. Wu, J. Qin, Convolutional neural network with median layers for denoising salt-and-pepper contaminations. *Neurocomputing* **442**, 26–35 (2021)
16. S. Maurya, I. Gupta, FPGA based hardware implementation of median filtering and morphological image processing algorithm. *Int. J. Eng. Res. Technol.* **3** (2014)
17. C. Priyanka, Median filter algorithm implementation on FPGA for restoration of retina images. *Int. J. Innov. Sci. Eng. Technol.* **3** (2016)
18. K.S. Raju, P. Phukan, G. Baurah, An FPGA implementation of a fast 2-dimensional median filter, in *National Conference on Recent Advances in Communication, Control and Computing Technology, RACCCT*, pp. 144–147 (2012)
19. A.H. Rasheed, FPGA-based optimized systolic design for median filtering algorithms. *Int. J. Appl. Eng. Res.* **12**(24), 16100–16113 (2017)
20. A. Rauh, G.R. Arce, A fast weighted median algorithm based on quickselect, in *2010 IEEE International Conference on Image Processing*, pp. 105–108 (2010)
21. S. Sadangi, P. Priyanka, FPGA implementation of parallel sorting mechanism for turbo decoding in lte system, in *2018 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 359–362 (2018)
22. R. Sedgewick, Algorithms in java, parts 1–4. 768
23. L. Shao, R. Yan, X. Li, Y. Liu, From heuristic optimization to dictionary learning: a review and comprehensive comparison of image denoising algorithms. *IEEE Trans. Cybern.* **44**(7), 1001–1013 (2013)
24. N. Sharma, P.J.S. Sohi, B. Garg, K. Arya, A novel multilayer decision based iterative filter for removal of salt and pepper noise. *Multimedia Tools Appl.* **80**(17), 26531–26545 (2021)
25. M.A. Vega-Rodríguez, J.M. Sánchez-Pérez, J.A. Gómez-Pulido, An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems, in *Proc. 10th Mediterranean Conf. Control and Automation* (2002)
26. H.-Y. Yang, X.-Y. Wang, P.-P. Niu, Y.-C. Liu, Image denoising using nonsubsampling shearlet transform and twin support vector machines. *Neural Netw.* **57**, 152–165 (2014)