Low Complexity Median Filter Hardware for Image Impulsive Noise Reduction

Hossein Zamani HosseinAbadi*
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
h.zamanihosseinabadi@ec.iut.ac.ir
Shadrokh Samavi
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
samavi96@cc.iut.ac.ir
Nader Karimi
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
nader.karimi@cc.iut.ac.ir

Received: 11/Aug/2013 Accepted: 12/Apr/2014

Abstract
Median filters are commonly used for removal of the impulse noise from images. De-noising is a preliminary step in online processing of images, thus hardware implementation of median filters is of great interest. Hence, many methods, mostly based on sorting the pixels, have been developed to implement median filters. Utilizing vast amount of hardware resources and not being fast are the two main disadvantages of these methods. In this paper a method for filtering images is proposed to reduce the needed hardware elements. A modular pipelined median filter unit is first modeled and then the designed module is used in a parallel structure. Since the image is applied in rows and in a parallel manner, the amount of necessary hardware elements is reduced in comparison with other hardware implementation methods. Also, image filtering speed has increased. Implementation results show that the proposed method has advantageous speed and efficiency.

Keywords: Image Processing, Noise Reduction, Median Filter, Hardware Implementation, FPGA.

1. Introduction

Impulse noise or salt and pepper noise usually corrupts images during image capture or transmission. This noise may be found in situations where quick transients, such as faulty switching, take place during imaging. Impulse noise appears as black and white dots in some random pixels of an image. As a result, the effectiveness and accuracy of image's subsequent processes (such as edge detection, segmentation, feature extraction and pattern recognition) will be negatively affected [1]. Median filter is a nonlinear filter, which is common and effective tool for removing impulse noise from images. By sweeping a noisy image and outputting median value of the median window, median filter can significantly reduce the amount of impulse noise. De-noising is an important operation in image pre-processing. This process is usually performed online and inside camera's hardware. The de-noised image can then be further processed to obtain necessary information. Therefore, hardware implementation of median filter is of great importance for the purpose of online image de-noising and preparing the image for principal processing.

A variety of studies have been done on hardware implementation of median filters [2]-[6]. For implementing the median filter a method, called standard method, is the sorting of pixels and extracting the middle value of the sorted pixels as the filter's output [2]. Karaman et al. [3] propose a change to the standard method by dealing with samples in a bitwise manner, needing only single bit sorters. The strength of regular array architectures is that they can be pipelined down to a single compare-swap stage. This means high throughput and frequency. Instead of standard sorting of all pixels, multilayer sorting, as explained in [5], could also be used. Implementation of the median filter by means of trace transform is yet another important study. In that method, instead of sorting of the pixels, numbers of all pixel values are recorded. The pixel that sum of the recorded numbers of its previous pixels is half of the sum of all pixel value numbers, will be the output. This method is effective for large windows, but in small windows (such as 3×3) vast hardware resources are required and this method is not suitable [5]. Use of a threshold value for computing the output of the median filter is presented in [6]. Some other studies are about implementing other filters with equivalent performance with median filters for image de-noising applications [7]-[12]. There are also studies about implementation of median and equivalent filters at VLSI level for costume IC design in order to boost clock pulse frequency and decrease usage area of the chip [13,14].

In this paper, a pipelined structure is proposed for implementing median 3×3 filter. The structure reduces necessary registers for filter implementation up to 50%.

* Corresponding Auhor

Afterwards, based on this structure, a pipelined method is introduced for reading images in a row by row manner and de-noising them. By using parallel modules in the proposed method, speed of the filter is significantly increased in comparison with sweeping the image by a 3×3 window.

In Section 2, various median filter implementation methods including standard and multilayer implementations are described. Afterwards, in Section 3, the proposed method for implementing median filter, reading images and applying the filter on them, is introduced. In Section4, simulation results are presented. Eventually, concluding remarks are presented in Section 5.

2. Standard and Multilayer Implementations of Median Filter

Median filter is a spatial filtering operation, so it uses a 2-dimensional mask that is applied to each pixel in the input image. To apply the mask means to center it in a pixel, evaluating the covered pixel values and determining which brightness value is the median value. The median value is determined by placing the pixel values in ascending order and selecting the center value [1]. The obtained median value will be the value for that pixel in the output image. Fig. 1 shows how a median window is applied to a part of an input image.

Since bigger windows may eliminate small edges of the input image, usually a 3×3 median window is used for image filtering. Therefore, we will focus on hardware implementation of the 3×3 median filter.

Sorting windowed pixels for extracting the median value is done usually with two methods: standard and multilayer. In the standard method, all 9 pixels of a 3×3 median window are sorted together. For sorting the pixels, 8 bit comparator elements (or pixel comparators) are used [2]. The standard sorting algorithm structure for 9 pixels of the window is depicted in Fig. 2. In this figure, each of the displayed elements, are 8 bit comparators. As shown in the figure, for sorting 9 pixels, 9×(9-1)/2=36 comparators are needed. However, dark colored comparators are used for sorting all pixels and are not required for extracting the median value. Thus, 30 comparator elements will be adequate to implement the standard method [5]. Interior structure of comparator elements is shown in Fig. 3. These elements compare two pixels, and send higher value pixel into the H output and lower pixel into the L output.

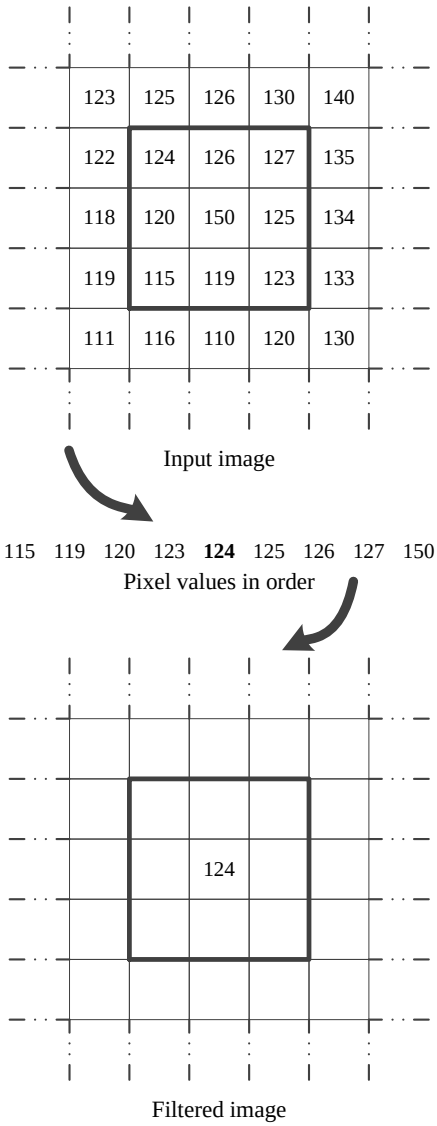

Fig. 1. Filtering of an image by median filter.

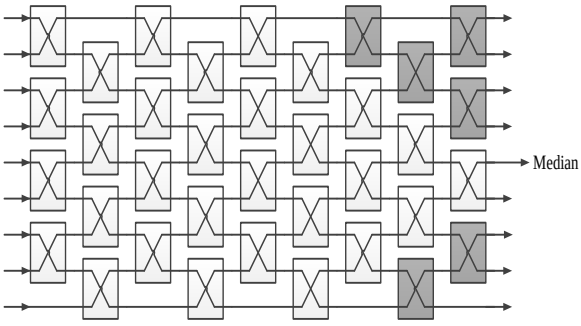

Fig. 2. Standard sorting algorithm [5].

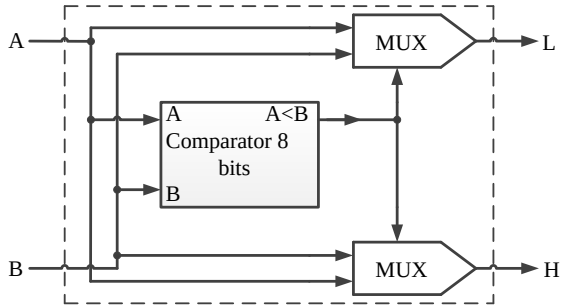

Fig. 3. Internal structure of comparators [2].

Some changes could be made in the standard sorting algorithm to reduce required hardware elements for implementing the median filter. For this purpose, instead of sorting all 9 pixels together, each column of the 3×3 median window is sorted independently in 3 sorting blocks. These sorting blocks sort the 3 input pixels in ascending order. Afterwards, outputs of the sorting blocks are grouped in 3 new sorting blocks and are sorted again in layer 2. By continuing this procedure, the median value could be extracted in the output of layer 3. This method is a multilayer implementation of the median filter. The multilayer sorting algorithm structure for 9 pixels of a 3×3 window is depicted in Fig. 4 [4]. Each of the sorting blocks in Fig. 4 is constructed from 3 comparators, as shown in Fig. 5.

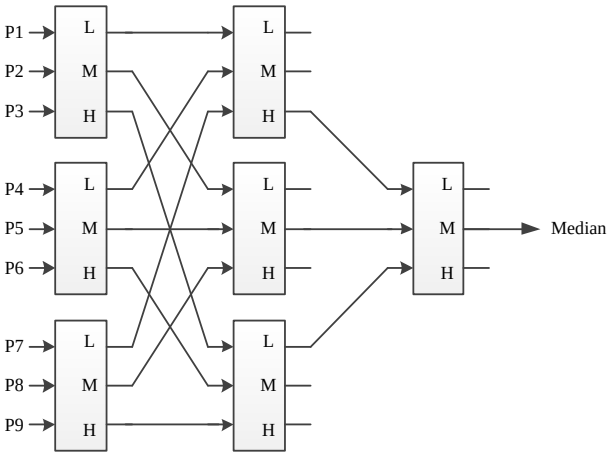

Fig. 4. Multilayer sorting of pixels.

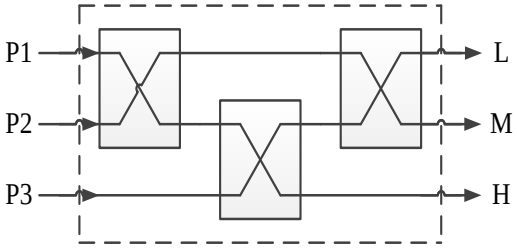

Fig. 5. Structure of a 3 pixels sorting block.

Multilayer implementation method has been proposed based on deleting non-median pixels. In fact, any pixel that is higher or lower than 5 of the other pixels in median window could not be the median value. By utilizing this logic, it could be proved that the output of the filter shown in Fig. 4 is the median pixel. This method uses only 7×3=21 comparator elements [15]. Pipeline structure for this method is introduced in [15]; this structure is displayed in Fig. 6. In the displayed structure, number of comparator elements is further reduced to only 15. In this structure, instead of sorting columns of the median window in different 3 pixels sorting blocks, by pipelining the hardware, one block is used to sort all columns in the layer one of filter.

3. Proposed Method

Median filter implementations that are introduced in previous section, must sweep the image to filter it; thus filtering of the whole image is slow and time consuming. To improve the filtering speed, at first, we propose a pipelined median multilayer structure, called 3-level pipelined filter. This structure is depicted in Fig. 7. In the proposed structure, there are 3 levels of filter and all levels are pipelined. This structure accepts 9 pixels as inputs and returns the median pixel as its output.

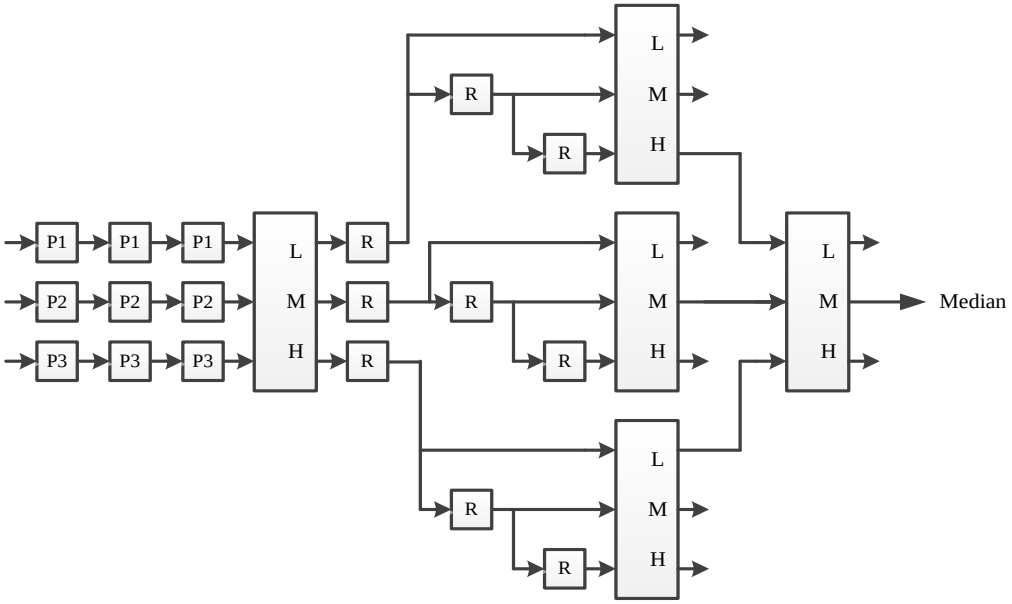

Fig. 6. Pipelined multilayer sorting structure [15]

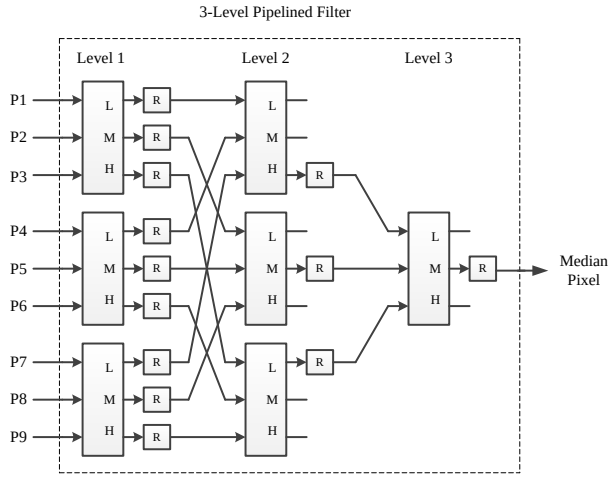

Fig. 7. Proposed pipeline architecture for multilayer median filter (3-level pipelined filter).

Although, in the 3-level pipelined filter, required hardware elements are more than the structure shown in Fig. 6; but, by introducing a useful method for reading images, these hardware elements could be decreased. By using some of the proposed 3-level pipelined filters in parallel form, hardware elements will be reduced. For the purpose of using parallel filters, image is transmitted to a pipelined hardware in a row by row structure. Thus, number of required filters is as same as number of pixels in a row of the image. Fig. 8 illustrates the manner of transmitting image pixels into the hardware. To increase clarity of the figure, only a few of the wires are illustrated.

Each median block in Fig. 8 has the proposed 3-level pipelined filter structure demonstrated in Fig. 7. According to Fig. 7, there is no need to feed all 9 pixels into a 3-level pipelined filter block of Fig. 8. In fact, by using parallel blocks, only 3 pixels could be enough to be fed into one block. To modify the 3-level pipelined filter structure, in level 1 of the structure, the upper and lower 3 pixel elements could be removed and instead of them, the 3 pixel elements from the adjacent 3-level pipelined filters could be used. Thus only 3 pixels are fed into each 3-level pipelined filter block of Fig. 8.

For reading image pixels and transmitting them into the hardware, primary pixels of each 3 rows in Fig. 8 will be transmitted into block 1; secondary pixels of each 3 rows in Fig. 8 will be transmitted into block 2, and so on, until the end of every 3 rows.

In level 2 of the 3-level pipelined filter, outputs of level 1 are used. In each 3-level pipelined filter block in Fig. 8, outputs of levels 1 of next and previous blocks can be used as well as outputs of level 1 of the block itself as the inputs of level 2.

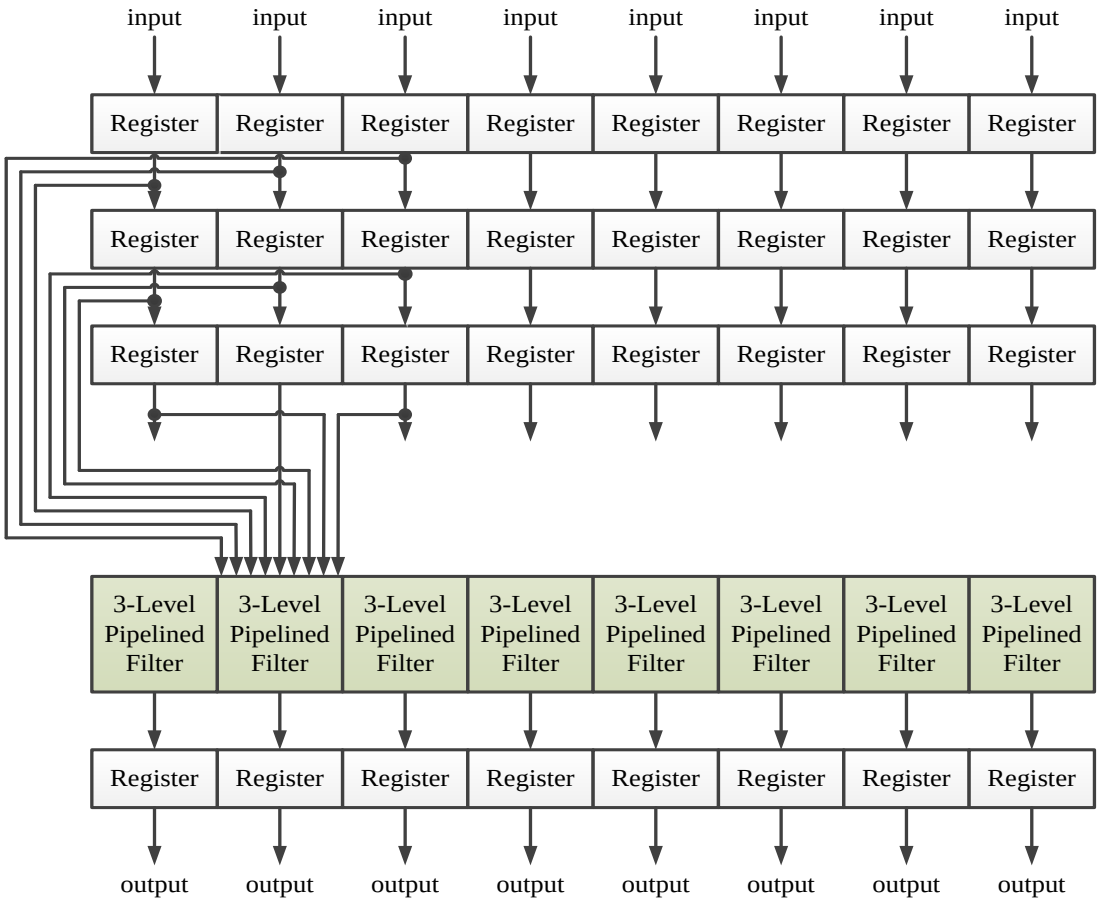

Fig. 8. Feeding of image into pipelined architecture.

According to the proposed image feeding technique, the architecture per each output pixel to de-noise a noisy image is shown in Fig. 9. As indicated in the figure, this structure accepts three pixels of a column of median window as inputs, and computes the output of median window. Since median window should be repeated in a whole row of a noisy image to de-noise that row, the structure demonstrated in Fig. 9 should be repeated for each pixel of the row. In this architecture, for each pixel, 5 comparators of the 3-inputs type are needed. Suppose that there are N pixels in each row of an image; for de-noising this image with our proposed method N demonstrated structures are required; therefore, $5\times3\times N=15N$ comparators are needed.

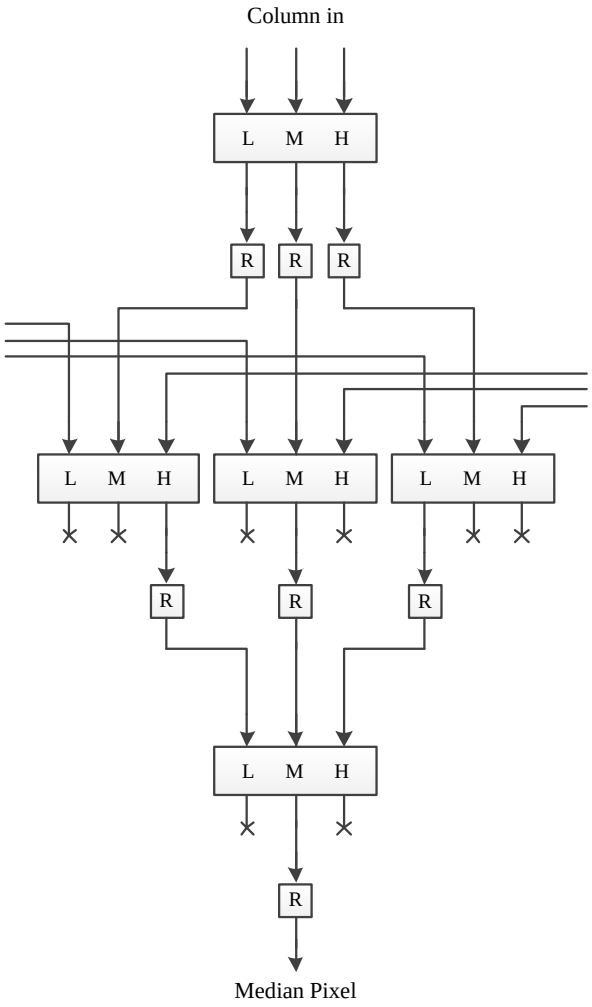

Fig. 9. Proposed pipelined multilayer median filter architecture for each pixel of image.

For de-noising the whole row of an image, and subsequently de-noising the image, the proposed pipeline multilayer median filter architecture, indicated in Fig. 9, should be repeated for all pixels of the row. Furthermore, primary registers those are shown in Fig. 8, should be included. Accordingly, by replacing each of the 3-level pipelined filter blocks and below registers in Fig. 8, by the proposed pipelined multilayer median filter shown in Fig. 9, our proposed structure for low complexity hardware image de-noising, can be constructed. Architecture of the proposed method for de-noising images is depicted in Fig. 10. This architecture will be repeated for the whole row of an image.

4. Simulation Results

The performance accuracy of the proposed architecture for image de-noising was proved by simulating it using Active-HDL 8.1 software. Several images with different row lengths used as the input noisy images. Impulsive noises by different noise factors were added to the images to corrupt them. Afterwards, the noisy images were de-noised by the proposed structure. Fig. 11 displays simulation signals including: clock pulse, input row, and output row for a de-noising process. The output signal activates after 6 clock pulses; hence, 6 clock pulses are required for the first row to be de-noised. Afterwards, subsequent rows need only 1 clock pulse to be de-noised. Therefore, the latency of the proposed structure is 6. This was expected since there are 6 registers in the path of each output pixel in the proposed structure as shown in Fig. 10. If the number of rows in an image is supposed to be M, the number of clock pulses those are needed to make the image de-noised, is equal to M+6. Moreover, as stated before in Section 3, if there are N pixels in each row of image; 5×3×N=15N pixel comparator elements are needed for de-noising this image with our proposed method.

Fig. 12 shows the results of image de-noising by means of the proposed median filter. In this figure, the results of image de-noising by the proposed structure are compared by the outputs of image de-noising by the 3×3 median filter in MATLAB software for three pictures. The pictures are corrupted by impulsive noise with different noise factors and the lengths of their rows are different. The length of image rows is 500 pixels for the left image and is 512 for the other ones. Accordingly, the proposed structure is established twice, both for 500 pixels and for 512 pixels.

In Fig. 12, original images, noisy images (10% of pixels are corrupted by salt and pepper noise for the two left images and 20% of pixels are corrupted for the two right ones), and de-noised images, both with the proposed median filter and with median filter in MATLAB, are displayed for both images. Also, the images which are de-noised by means of the 3×3 mean filter in MATLAB are displayed in the figure. As displayed in the figure, the proposed median filter could de-noise the noisy images with desired efficiency. Moreover, the priority of the proposed filter over the mean filter in image de-noising can be concluded.

In order to further illustrate the performance of the proposed filter, in Fig. 13 we have compared the produced histograms of different filters. Figure 13 (a) illustrates the original image and its histogram. The image has a relatively distributed histogram with all levels of gray-scale values. Figure 13(b) shows the noisy image due to 10 percent salt and pepper noise. The histogram of noisy image shows high number of induced zero points (peppers) and large number of white pixels (salts). The

effect of mean filter is shown in Fig. 13 (c) where the quality of the produced image may be better than the noisy image but still is far from desirable. While the histogram of the original image has three distinct peaks, the histogram of the mean filter is very much flat and barely contains the three mentioned peaks. On the other hand, Fig. 13 (d) shows the output of our median filter where the image and its histogram are very close to the original image and its histogram.

The proposed architecture was further simulated in Xilinx ISE 11.1 software to determine the required hardware elements and maximum clock frequency. A Virtex5 family FPGA, XC5VTX240T, was used as the target device. Different implementation methods of the median filter were synthesized separately; and necessary hardware elements for implementing the filter using each method were computed based on the device's resources. The proposed median window architecture was compared with the standard median filter [2], multilayer median filter [4] and the pipelined multilayer median filter introduced in [15]. Required hardware resources and minimum filter's delay for all methods were computed and are shown in Table 1.

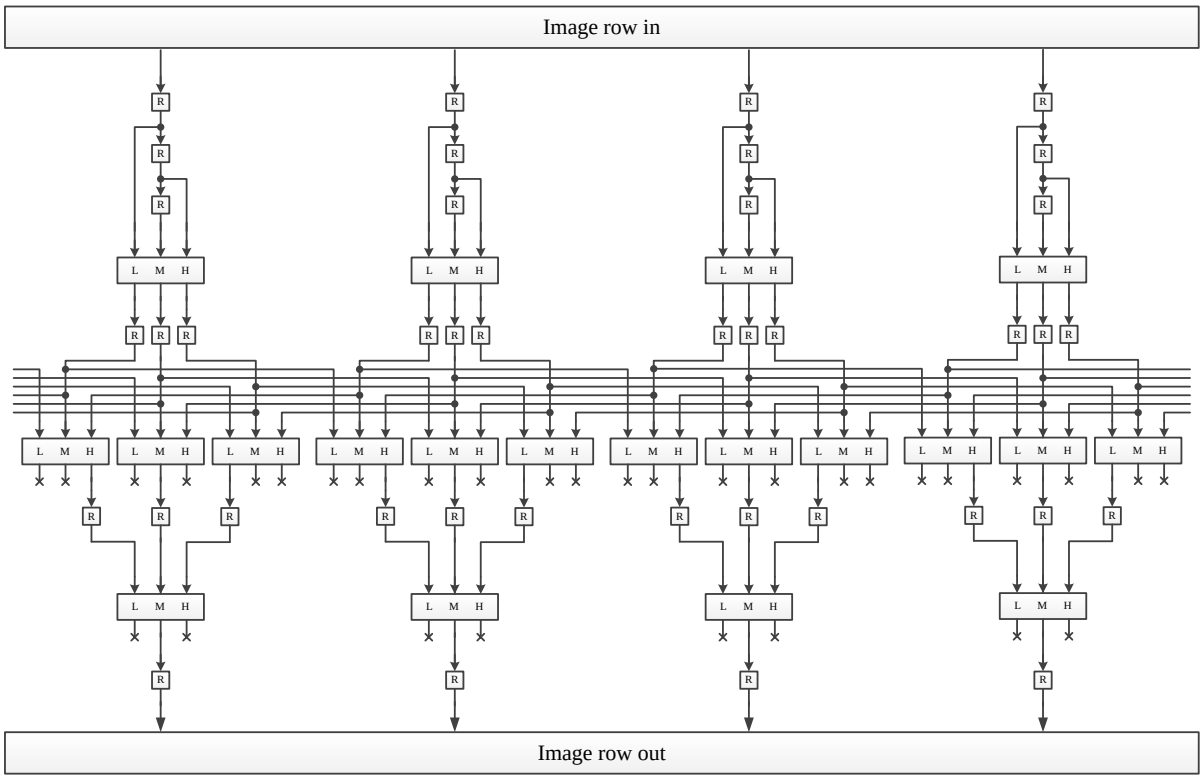

Fig. 10. Architecture of proposed method for image de-noising (structure for only 4 pixels is shown).

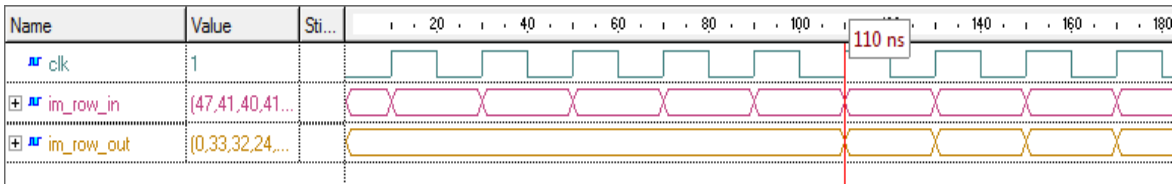

Fig. 11. Three principle simulation signal

Fig. 12. (a) Original images; (b) noisy images by 10% and 20% salt and pepper noise for the left and right images respectively; (c) images denoised by the proposed method; (d) images denoised by median filter in MATLAB software; (e) image de-noising by mean filter in MATLAB software.

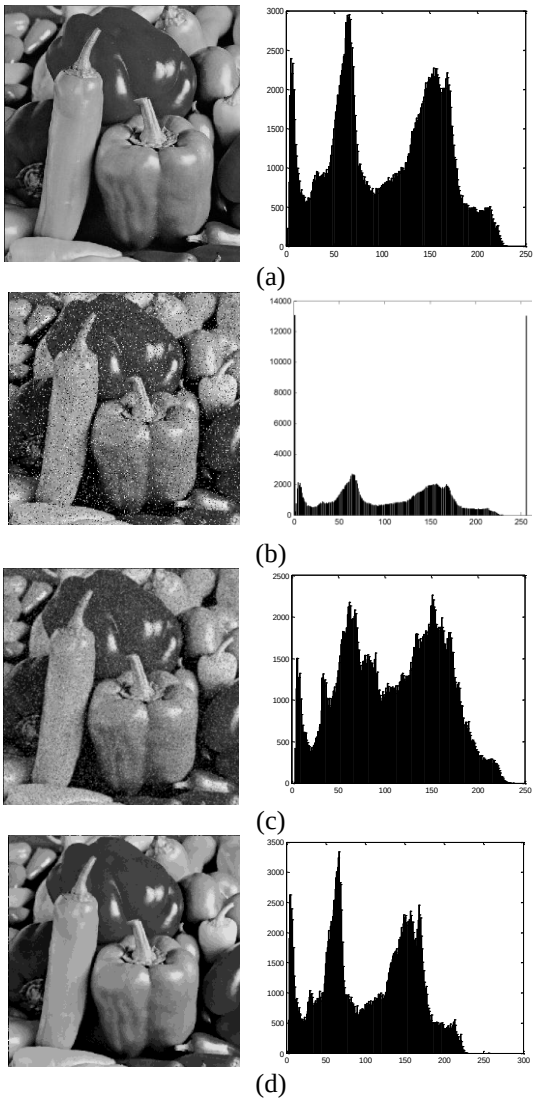

Fig. 13. Comparison between output of our filter and that of a 3x3 mean filter using histograms, (a) original image, (b) noisy image due to 10% salt and pepper noise, (c) output of mean filter, (d) output of our median filter

Table 1. Synthesis results for different median filter implementation methods.

Method	*# of LUTs*	*# of DFFs*	*Delay (ns)*
Standard [2]	552	0	34.614
Multilayer [4]	316	0	33.787
Pipelined multilayer [15]	224	768	2.006
Proposed method	224	384	1.947

According to the results demonstrated in Table 1, in comparison with pipelined multilayer median filter [15], the proposed architecture has up to 50% reduction in the required registers; also it has increased speed of the hardware up to 3%. In each of these methods (the proposed and the pipelined multilayer[15]), needed LUTs for implementing the filter is 29% lower than the multilayer median and 57% lower than the standard median filter. Also, using the proposed structure can lead to filter speed increase between 1.025 to 17.78 times in comparison with the other pipelined and not pipelined median filter implementation methods.

5. Conclusions

In this paper, a method for implementing the median filter was proposed to reduce required hardware elements and to increase the processing speed. In the proposed method, images are applied to the filter in rows and with a pipelined method. Filter elements are pipelined, too. Accuracy of the proposed architecture in removing salt and pepper noise was demonstrated by de-noising of sample noisy images. The synthesis results revealed that the proposed method could increase filter speed up to 3% and decrease the required hardware elements up to 50% in comparison with the existing pipelined multilayer median filter structure.

References

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd ed. New Jersey: Prentice Hall, 2008, pp. 225-227.

[2] D. Richards, "VLSI median filters", IEEE Transaction on Acoustics, Speech and Signal Processing, Vol. 38, No. 1, 1990, pp. 145-153.

[3] M. Karaman, L. Onural and A. Atalar, "Design and implementation of a general-purpose median filter unit in CMOS VLSI," IEEE Journal of Solid-State Circuits, Vol. 25, No. 2, 1990, pp. 505–13.

[4] J. L. Smith, "Implementing Median Filters in XC4000E FPGAs", XCell, Vol. 23, No. 4, 1996, p. 16. [Online]. Available: http://users.utcluj.ro/~baruch/resources/Image/xl23_16.pdf

[5] S. A. Fahmy, P. Y. K. Cheung and W. Luk, "Novel FPGA-based implementation of median and weighted median filters for image processing", in Proc. 2005 International Conference on Field Programmable Logic and Applications FPL, pp. 142-147.

[6] A. Burian and J. Takala, "VLSI-efficient implementation of full adder-based median filter ", in Proc. 2004 IEEE International Symposium on Circuit and Systems, Vol. 2, pp. 817-820.

[7] H. S. Yu, J. Y. Lee and J. D. Cho, "A fast VLSI implementation of sorting algorithm for standard median filters", in Proc. 1999 IEEE International ASIC\SOC Conf., pp. 387-390.

[8] C. T. Chen, L. G. Chen and J. H. Hsiao, "VLSI implementation of a selective median filter", IEEE Transaction of Consumer Electronics, Vol. 42, No. 1, 1996, pp. 33-42.

[9] L. Breveglieri and V. Piuri, "Digital median filters", Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, Vol. 31, No. 3, 2002, pp. 191-206.

[10] K. S. Srinivasan and D. Ebenezer, "A New Fast and Efficient Decision- Based Algorithm for Removal of High-Density Impulse Noises", IEEE Signal Processing Letter, Vol.14, No.3, 2007, pp. 189-192.

[11] T. Matsubara, V.G. Moshnyaga and K. Hashimoto, "A FPGA implementation of low-complexity noise removal.", 17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), IEEE, 2010.

[12] D. Prokin and M. Prokin, "Low hardware complexity pipelined rank filter", IEEE Transactions on Circuits and Systems II: Express Briefs, Vol.57, No.6, 2010, pp. 446-450.

[13] C.Y. Lien, C.C. Huang, P.Y. Chen and Y.F. Lin, "An efficient denoising architecture for removal of impulse noise in images", IEEE Transactions on Computers, Vol.62, No. 4, 2013, pp. 631-643.

[14] P. Chen, C. Lien and H. Chuang, "A low-cost VLSI implementation for efficient removal of impulse noise", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.18, No.3, 2010, pp. 473-481.

[15] K. Vasanth, S. Nirmal raj, S. Karthik and P. Preetha mol, "FPGA implementation of optimized sorting network algorithm for median filters", in Proc. 2010 International Conference on Emerging Trends in Robotics and Communication Technologies (INTERACT), pp. 224-229.

Hossein Zamani Hosseinabdi was born in 1988 in Isfahan, Iran. He received his B.Sc. and M.Sc. degrees in Electrical Engineering from the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran in 2010 and 2013 respectively. His research interests include analog/mixed signal integrated circuits, implementable and wearable electronics, hardware implementation of signal processing algorithms and digital signal processing.

Shadrokh Samavi is a Professor of Computer Engineering at Isfahan University of Technology, Iran. He is also an Adjunct Professor at the ECE department of McMaster University where he is a member of the Multimedia Signal Processing Lab. Professor Samavi completed a B.S. degree in Industrial Technology and received a B.S. degree in Electrical Engineering at California State University, a M.S. degree in Computer Engineering at the University of Memphis and a Ph.D. degree in Electrical Engineering at Mississippi State University, U.S.A. Dr. Samavi is a Registered Professional Engineer (PE), USA. He is also a member of IEEE and a member of Eta Kappa Nu and Tau Beta Pi honor societies. Shadrokh Samavi's research interests are in the areas of image processing and hardware implementation and optimization of image processing algorithms. He is also interested in compression and processing of biomedical images, as well as, VLSI design and computer arithmetic.

Nader Karimi received the B.S. degree (summa cum laude) in computer engineering from Azad University, Arak Branch, Iran, in 2002 and the M.Sc. and Ph.D. degrees (honor) in computer engineering and electrical engineering from Isfahan University of Technology (IUT), Iran, in 2004 and 2012, respectively. He is currently an Assistant Professor at the Department of Electrical and Computer Engineering, Isfahan University of Technology. His research interests are image compression, hardware implementation and optimization of image processing algorithms, and watermarking.