

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/348869848>

FPGA –based Optimized Systolic Design for Median Filtering Algorithms

Article · January 2017

CITATIONS

3

READS

223

1 author:



[Asmaa Hameed](#)

Al-Nahrain University

6 PUBLICATIONS 14 CITATIONS

[SEE PROFILE](#)

FPGA - based Optimized Systolic Design for Median Filtering Algorithms

Asmaa Hameed Rasheed

Lecturer

*College of Engineering, Baghdad University
Baghdad, Iraq.*

Abstract

FPGAs are efficiently used for high speed digital systems design and implementation due to their ability of parallel processing. FPGA based implementation of median filter is expensive, since the comparison operation needs a very complex hardware that make it a severe drain process of the available digital components of the FPGA kit. This paper presents a deep study and analysis for optimized systolic architecture of median filter design to gain maximum possible reduction in the required comparator elements and then minimize highly the needed hardware components. Many techniques can be used to implement the sorting process, which is the core of median filter architecture, to pipeline filter activities that leads to reduce overall execution time. A pipelined windowing technique is used to mask specified number of pixels to scan the processed image, and the window size is determined by filter length. Improved systolic architectures of median filter algorithms, with different filter lengths, have been analyzed to check hardware complexity according to filter efficiency and output quality. Optimized FPGA based hardware design and implementation have been discussed deeply for 3*3 filter length. A hardware complexity evaluation was made compromised with different performance criteria. Several performance criteria are tested for three different filter lengths 3*3, 5*5 and 7*7. The median filter with length of 9*9 samples is also tested but with time constraints only. The architectures presented in this work utilize MATLAB Simulink and Xilinx System Generator (XSG) both together to get appropriate hardware implementation. Generation of optimal VHDL code of the processed XSG design is achieved using ISE 14.7 hardware programming package. Implemented system has its performance measured in terms of resource utilization by using Spartan6 - (XC6SLX45T). The developed design was applied to a gray image of size 250*400, but this approach can be used for any image size without any reprogramming or reconfiguring for the used FPGA kit.

Keywords: Median filter, systolic architecture, parallel processing, FPGA, and system generator (SG).

1. INTRODUCTION

Digital images can be affected by different noise types such as Gaussian, Impulse, Rayleigh, Gamma, Exponential, and Uniform noise, [1], [2]. Impulse noise occurred due to different image system processing such as scanning, digitizing, and transmission through channels. Salt & pepper noise, is a special type of impulse noise, when corrupt an image, the affected pixel will take the minimum or maximum

gray level which are 0 and 255 respectively. Median filter is very efficient to eliminate and cancel salt & pepper noise. Median filter behaves well at lower percentages of noise densities, but an advanced version of this filter, such as adaptive median and weighted median will be needed with large noise densities [3], [4], [5]. The main obstacle for these advanced versions of median filters is that their designs cannot normally meet real time processing requirements. Sliding window always used which moves systematically over the entire noisy image depending on the stationary feature of this image [6].

FPGA's are used in a very wide applications of digital image processing including image filtering [7]. This is due to FPGA's features that make it the most candidate choice for many real time processing. Some of these features are: flexibility, reconfigurability, and parallelism. The implementation of image filtering on PC takes more time because it lacks to the parallelism activity in all programming methods and execute all its processes in a sequential manner. FPGA supports parallelism [8], that accelerates real time processing and then leads to high speed response compared to the personal computer.

The internal structure of FPGA consists of a grid of programmable logic cells supported by a programmable interconnection lines and switches that provide communication media for these logic cells. The I/O cells which are arranged around them, provide an interface between external pins and interconnection lines of the used chip [9], [10]. In fact, programming an FPGA is specifying logic function to each cell and to each interconnection lines [11], [12].

2. IMAGE FILTERING TECHNIQUES

Image denoising is one of the most applicable areas in digital image processing. Impulse noise is one of the most noise types that is produced with image processing systems such as image storage, or transmission. A suitable denoising method, depends mainly on filtering process that used to reduce noise effects and must not affect the other original details of the noisy image. Here, image filtering system is used for noise cancellation, while retaining edges and image characterizing feature.

2.1. Median Filter

The median filter is a nonlinear digital filter, effectively used for impulse noise cancellation that has a uniform probability density function as shown in Figure 1.

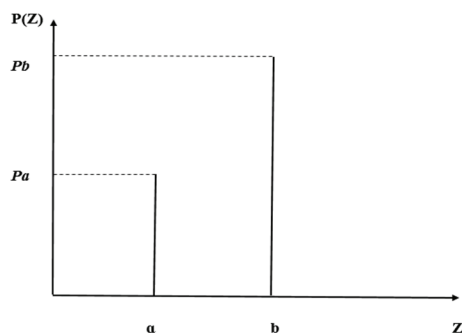


Figure 1: Probability density function for impulse noise.

The affectivity of median filter referred to its ability to preserve edges of the images and many useful details, while removing or at least eliminating noise effect under certain conditions. This filter is a local filter that produces the middle element of a sorted values of number of pixels taken from a specified window as shown in Figure 2. The filtered pixel depends on its nearby neighbors to decide if it is a good representative of its surroundings [13], so the filter replaces the value of the pixel with the median of a specified window.

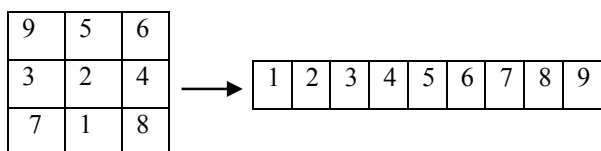


Figure 2: Sorting Procedure for traditional median filter design.

The median filter processing can be summarized by following steps; firstly: select a specified window depending on the filter length, then the used algorithm sorts the pixels in ascending or descending order. The second processing step of this filter: gives the median value of the sorted elements. The number of pixels in the masking window is preferred to be an odd number to select the middle value as median. But when this number is even, the filter then takes the average of two middle pixels as median. The last step of filter processing: is to place the median at the centre of the selected window. These processes are done for all image pixels by moving the pixel pointer horizontally and vertically.

3. H/W IMPLEMENTATION OF MEDIAN FILTER

Median filter implementation includes several processing stages. One of these stages is the image *preprocessing* in which the image must be segmented and convert each segment into 1- dimensional form. A $k \times k$ processing window is required to scan the images, so a $k \times k$ window generator is used which imitates a segmentation process, generating $(k \times k)$ pixels at a time as it scans the entire image, this $(k \times k)$ pixels are fed in a parallel manner. k is determined according to the used median filter length.

The second and the most important stage focuses on the optimal choosing of the suitable *hardware implementation technique* of median filter based on suitable H/W platform. In the present work, the design and hardware implementation of

this filter depends mainly on XILINX System Generator (XSG) block set. The final step in H/W implementation of median filter is the *post image processing*. Since the output of the system will be floating point, it needs to be converted to unsigned integer of 8-bit, because data type of image pixels are unsigned integer taking up 8 bits of data, the output pixel will be an individual pixel, which must be converted back to 2D data using the reshape block. The image is now ready for display.

These processing can be summarized, in detail as follows:

1. Read the input gray scale image.
2. Compute the size of an input image, and then duplicate all sides of processed image.
3. Initialize the pixel pointer to points to the first pixel.
4. Do the filtering activities.
5. One additional iteration is applied to eliminate any unnecessary noise at edges.
6. Display the filtered image.

When the simulation complete, the System Generator (SG) blocks are used with a selected type of FPGA kit to generate a new block that represents Xilinx block design in the last simulation step to form the real kit. Figure 3 shows the H/W design flow using system generator.

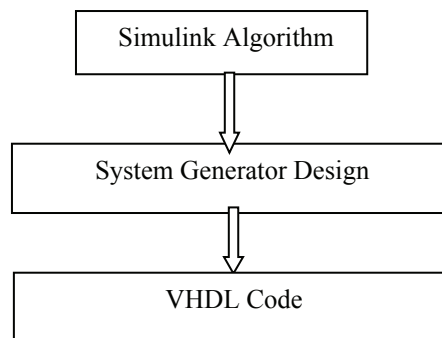


Figure 3: Design flow using XSG

The median filter is implemented here with different window size: 3×3 , 5×5 , and 7×7 to compare median filter performance with these different sizes, and to ensure the effect of increasing or decreasing the window size on the complexity of filter hardware design and on filtered image quality.

4. MEDIAN FILTER DESIGN TECHNIQUES

The core of median filter design, as mentioned earlier, is the sorting process that needs many comparison operations or comparative elements. This process can be done using S/W or H/W methods. For S/W implementation; the number of needed comparison operations is related directly to the number of sorted samples as illustrated in equation (1).

So, the sorting process of N samples needs Z comparison operations as follows:

$$Z = (N - 1) + (N - 2) + (N - 3) + \dots + 1 \quad \dots \dots (1)$$

The first sample needs $(N-1)$ comparison operations to be the smallest, or largest, element among the sorted samples, since it must be compared with all its followers which they are equal to $(N-1)$. After that, the number of rest samples is now $(N-1)$. So, the second samples needs $(N-2)$ comparison operations to be in the order of sorting elements, since it must be compared with all its followers that count $(N-2)$. This process continue until reaching last samples that do not need any comparison operation.

H/W implementation of median filter can be accomplished in different methods including optimized pipelined techniques.

4.1. Pipelined Hardware Techniques

The conversion of sequential S/W comparison to a pipelined H/W comparators for 3×3 median filter implementation results of **36** pipelined comparators as shown in Figure 4.

Some repeated comparison operations can be excluded, if it can be determined exactly, to minimize the required number of essential comparator elements. Thus, a pipelined design of sorting process of 9 samples can be done using **27** comparators only, as shown in Figure 5.

This pipelined designs can sort all the input samples, not only find the middle element which represents the median value. But for median filter design the middle element is only needed. Getting only the middle elements will lead to cancelling many outputs rather than the middle element, so several comparator elements can be removed and the overall design will be highly minimized.

Further reduction in the required number of comparators can be achieved by finding only the median element without sorting the hole samples.

This reduction can be easily developed to **23** comparators only by neglecting the sorting process of O/P samples in Figure 5, and find the middle value only as shown in Figure 6.

A special design of 9 samples can be done using $(3-I/P)$ comparators. This design classify the samples to three different groups for maximum values, median values, and minimum values. Then collect the expected median values of these three groups to be the inputs of the final comparator as shown in Figure 7-A. The core of $(3-I/P)$ comparator) is a configuration of three $(2-I/P)$ comparator) as shown in Figure 7-b. Now refer to Figure 7-a, if each one of the seven $(3-I/P)$ comparator) is replaced by its equivalent, that is shown in Figure 7-b, a new optimized design of median filter can be obtained that reduce the number of required comparator elements to **21** comparators only as shown in Figure 8.

Since we need only the middle element and the other sorted samples are not used here, the upper colored comparator can be removed to minimize the number of required nodes to **20** two-input comparators only. Also the lower section of $(3-I/P)$ comparator) enclosed by dashed lines can be flipped over as shown in Figure 9, so the lower colored comparator can be removed also, then the final number of over all required comparators can be minimized to **19** comparators only as shown in Figure 10.

In fact there are several pipelined techniques to find the middle value and the developed methods concentrate on minimizing the used number of needed comparators that is essential to compare between different samples values. Some of these techniques depend on directional searching and comparing in horizontal and vertical lines to scan the samples in the selected window. The other methods divide the processing area to distinct sequential stages to reduce the hardware complexity of the accomplished design.

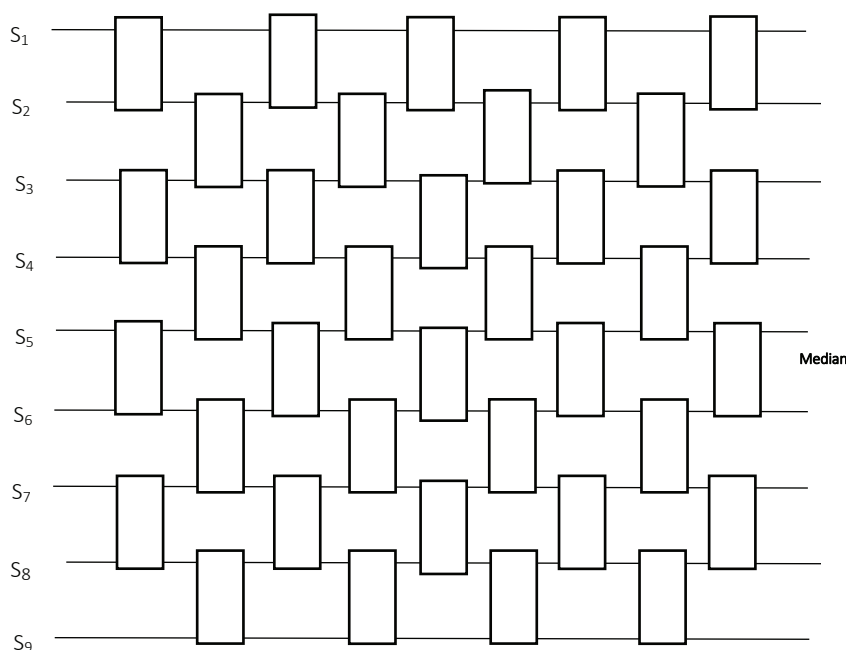


Figure 4: Pipelined design of 3×3 median filter using systolic array with 36 comparators.

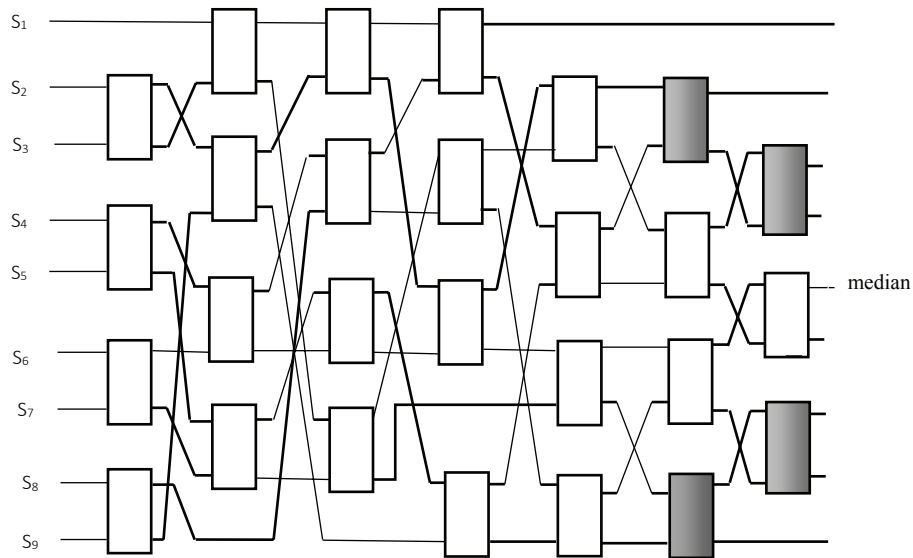


Figure 5: Pipelined design of 3*3 median filter using systolic array with **27** comparators.

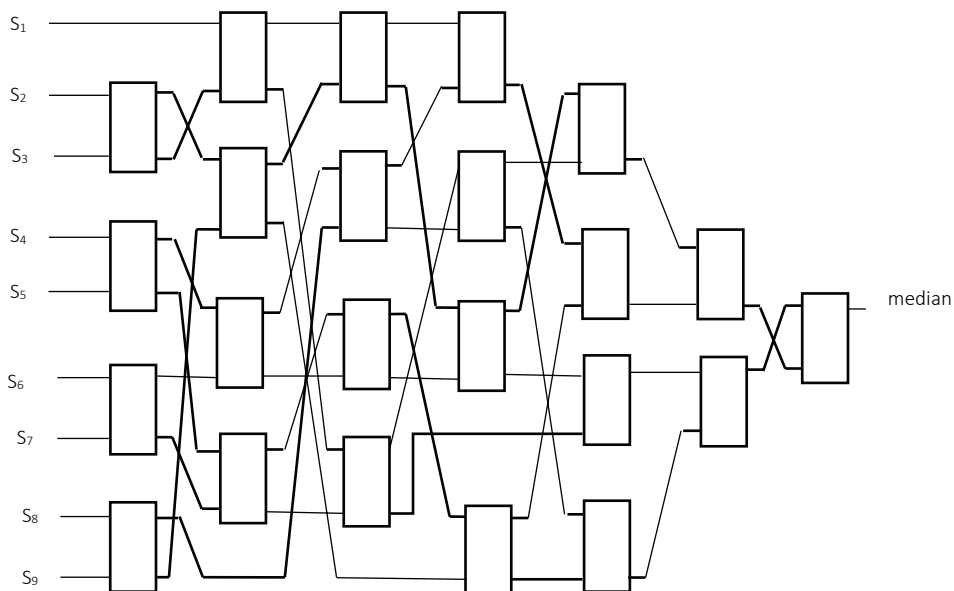


Figure 6: Pipelined design of 3*3 median filter using systolic array with **23** comparators.

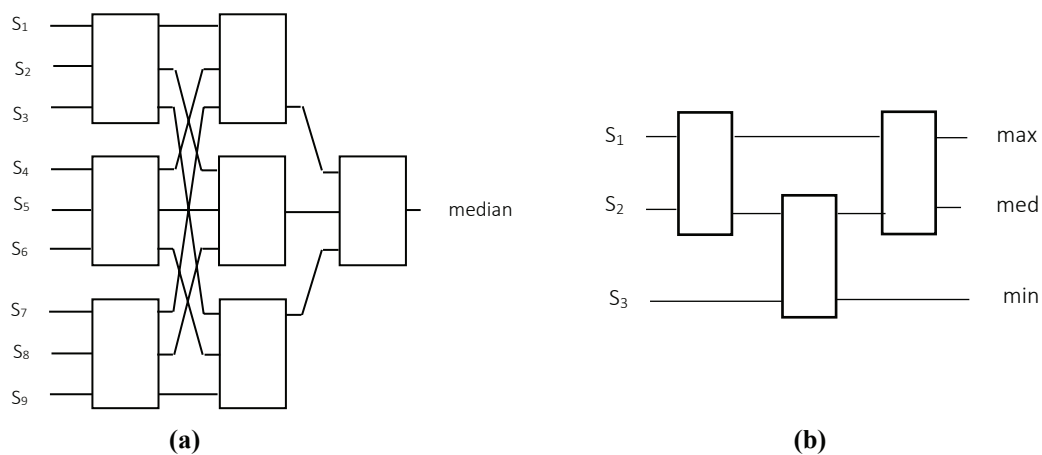


Figure 7: Pipelined design of 3*3 median filter using systolic array architecture of 3-I/P Comparator.

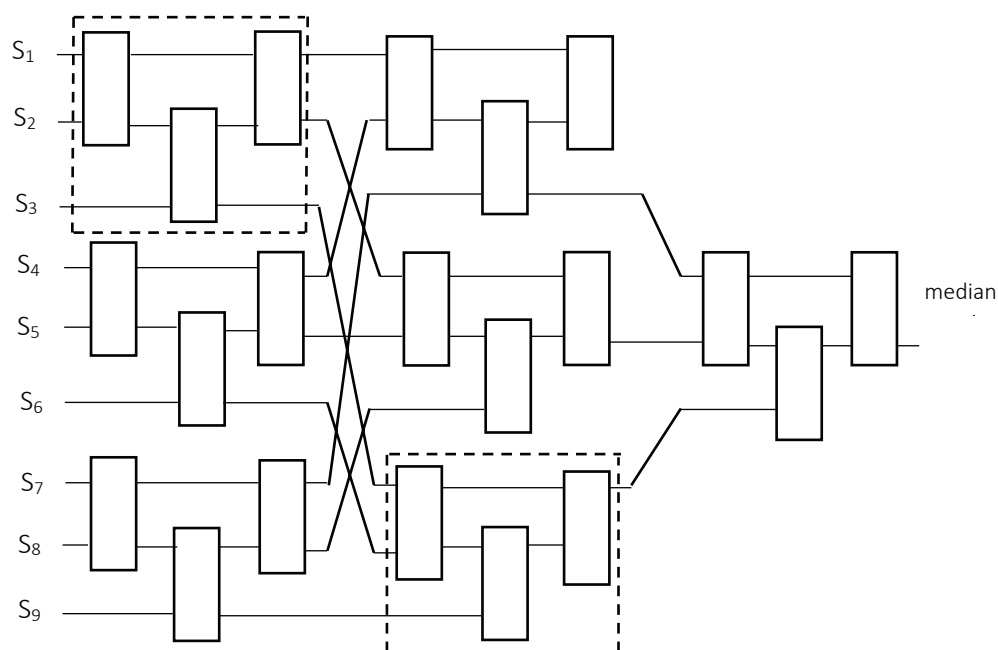


Figure 8: Pipelined design of 3*3 median filter using systolic array with **21** comparators.

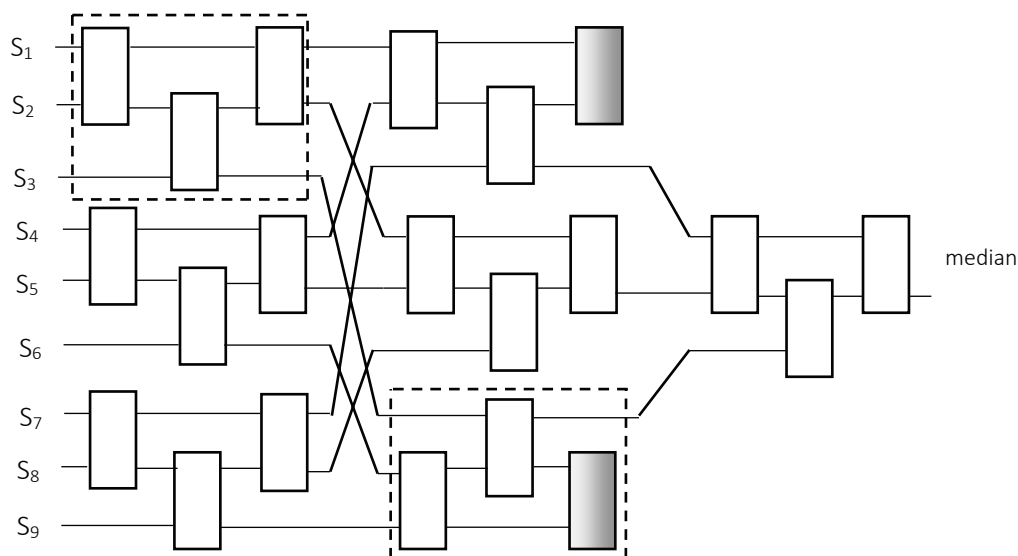


Figure 9: Pipelined design of 3*3 median filter using systolic array with **21** comparators after flip over the lower block.

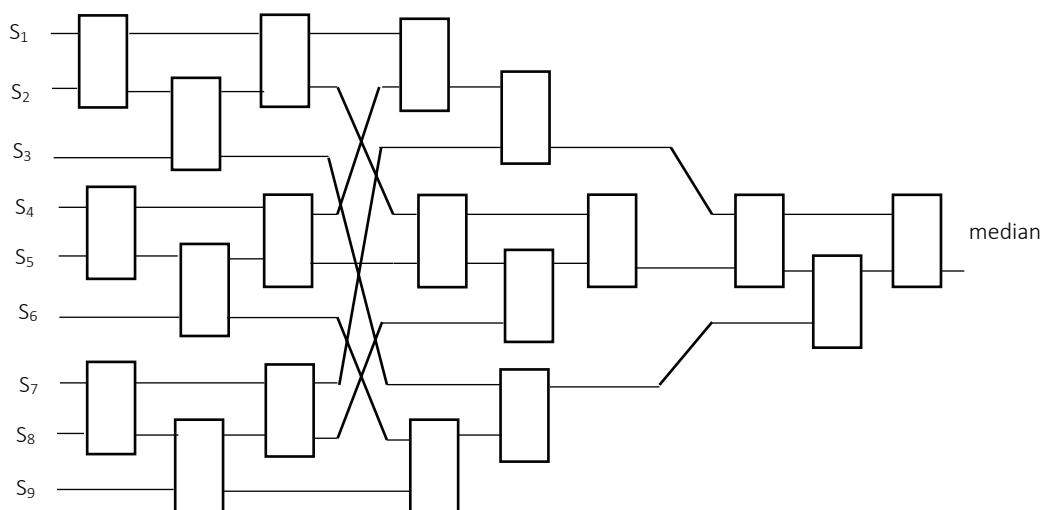


Figure 10: Pipelined design of 3*3 median filter using systolic array with **19** comparators

4.2. Traditional Simulink Design of Median Filter

The classical structure using one simulink block depending completely on S/W. In this structure the VHDL code completes all the design steps required for median filter implementation [14], [15] i.e. the comparison operations are accomplished by S/W processes. The Xilinx System Generator block set cannot serve our application. Then we can build our useful blocks, (as a user function in C language), by using a black box that is available in the XSG block set. This black box is empty and do not contain any predefined function. Thus, the black box can be filled with any required processes for any undefined function in the available XSG block set. Hence a sorting process can be built in the selected black box to be the core of hardware implementation of median filter design based on FPGA platform.

The Median of ($k \times k$) pixels can be calculated using the traditional sorting method, which is done, as illustrated previously, by arranging the pixels in ascending or descending order and picking the middle value as a median, see Figure 2, or it can be done by calculating the distance between the pixels using the distance norms. The pixel with the minimum distance to all the pixels represents the median [11].

An FPGA system of 3×3 median filter design using traditional method is shown in Figure 11 [14], [15]. The VHDL S/W inside the block box does all the sorting process to produce the median value as a filter output. Timing waveforms of 3×3 median filter was shown in Figures 12. The binary numbers in the wave diagram represents the values of the nine samples in the specific selected window with their computed median. It is clear, as shown in the table besides wave diagram, the output is really the median of the input nine samples.

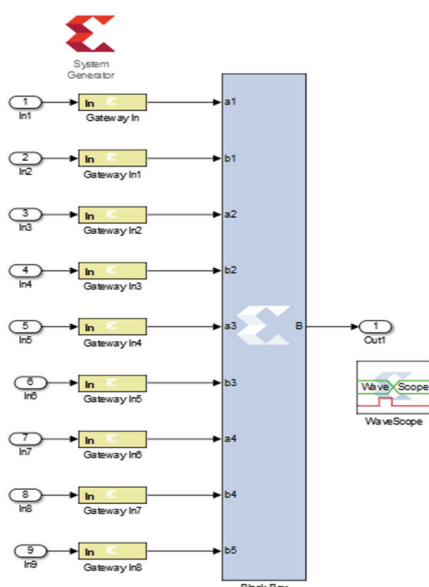


Figure 11: XSG-simulink design of 3×3 median filter design using traditional method [15].

This analysis can be extended easily for any filter length by changing the number of input samples according to the selected window size.

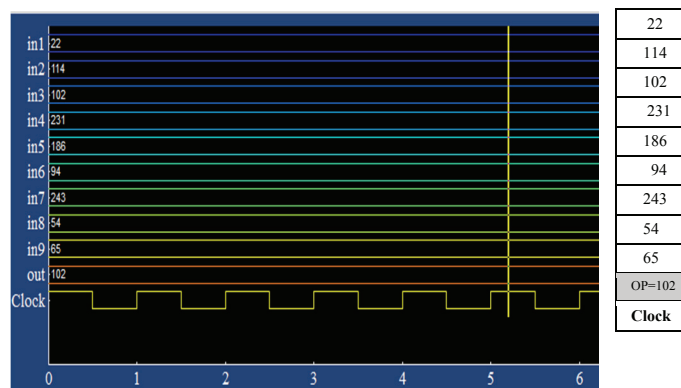


Figure 12: Timing diagram for 3×3 median filter using traditional method with sample of 9 pixels.

Performance characteristics are accomplished for 3×3 , 5×5 , and 7×7 median filter design using traditional method. For filter length of 9×9 the computation of execution time is done only.

4.3. Optimized Simulink Design of Median Filter

FPGAs are used for efficient median filter design based on different novel algorithms [16], [17], [18], [19]. Different optimized structure are dependant here using Simulink tools. All methods and different techniques tend to find the median value of the selected samples which is calculated in various ways using systolic algorithm [20], [21], [22].

The core of all these designs is the comparator element. However the comparator that available in the XSG block set cannot be useful here, because it cannot order the inputs, but only indicates to six different selected relations (equal, greater than, and less than.....etc) that describes the relation of the input samples as shown in Figure 13.

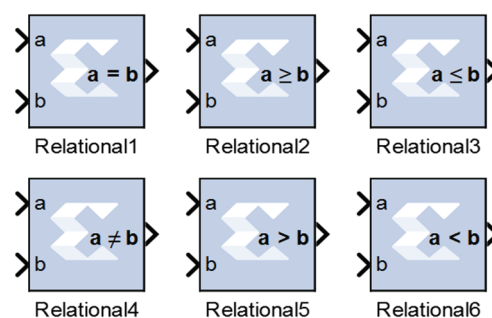


Figure 13: XSG comparator.

Thus, a special comparator must be built to be used as a basic node in all implemented systolic designs. This comparator is simply a black box with two inputs and two outputs and contain a very simple VHDL code that compares these two input samples and makes the greater one as the first O/P and the second output will be the sample with smallest value as shown in Figure 14.

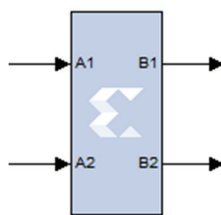


Figure 14: XSG black box used as a special comparator, the basic node of all median filter systolic design.

All the FPGA based systolic design of median filter implementation are a configuration of limited numbers of this special comparator but they usually arranged at each time in some different manner.

The optimized structures of median filter design are developed based on the parallelism of systolic array [21], [22]. The principle structure of three 2-inputs comparators is used as a basic block for large systolic filtering system, see Figure 15. Each comparator will be considered as a single node for all the used designs.

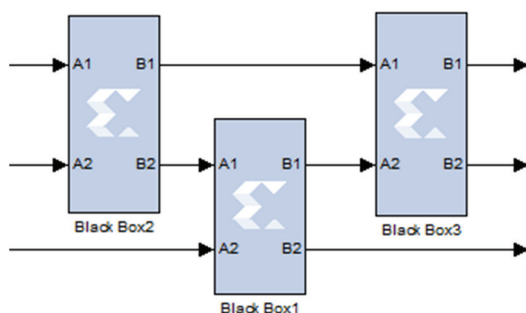


Figure 15: XSG Architecture, the core of 3*3 median filter design.

5. THE RESULTS

The median filter is implemented in different methods and techniques which can be classified mainly as:

- traditional method
- systolic architecture techniques.

Design utilization summary of a traditional design of 3*3 median filter is presented in Table 1.

Several systolic architecture are implemented and tested in the present work for 3*3 median filter. These optimized systolic architecture are differ mainly in number of nodes, each node is a 2-I/P comparator. The implemented systolic architecture starts with 36 node and then goes down to 27 nodes, 23 nodes, 21 nodes until reaching 19 nodes only. All these optimized systolic designs are implemented using FPGA platform to test the reduction in the FPGA CLB and internal digital components for each developed systolic design.

Design utilization summary of 36 nodes, 27 nodes, 23 nodes, 21 nodes, and 19 nodes designs are shown in Table 2, Table 3, Table 4, Table 5, and Table 6 respectively. These tables show that the number of bonded IOBs for all designs is 81. This emphasis that all designs have 9 stage each with 9 IOB.

The total number of IOBs can be determined by following formula:

$$Total\ IOBs = No.\ of\ stages * No.\ of\ IOBs/stage \dots \dots (2)$$

Thus total number of IOBs for implemented designs equal to 9 (no. of stages) multiplied by 9 and then equal to 81, as shown in the utilization summary tables. 9 locations are set for each stage; although some of these location may be terminated later.

The traditional method is applied for different filter length to compute the execution time for each one. Some important performance factors of median filter such as storage capacity, needed LUT and maximum clock cycle allowed for each filter length are computed and well presented in Table 7.

The implemented systolic array based on XSG of 21 nodes is shown in Figure 16, and the improved systolic design of 19 nodes is shown in Figure 17. For last two figures, each node is replaced by a two-I/P comparator, that previously illustrated in Figure 14. Figure 18, describes the rise of processing time with respect to increasing in filter length. While Figure 19 and Figure 20 show the variation in used LUT and storage memory respectively corresponding to various filter length of median filter.

The required time for 3*3 median filter, in which each selected window contains 9 pixels, with traditional and different optimized systolic techniques are summarized in Table 8. Number of used LUTs and required memory for each design of 3*3 median filter is also shown in Table 8.

Figure 21 displays the processing time form all implemented systolic designs. The minimization in the used LUT for all implemented designs are shown in Figure 22, while Figure 23 illustrates the storage memory required for each design, that shows the improved optimized systolic design needs less memory than traditional design method for median filter implementation.

For 3*3 median filter, it's clear that the execution time is effectively reduced for single window from 911.576 ns with traditional method to 686.495 ns with the proposed systolic architecture technique of 19 nodes as shown in Figure 21. And if the execution time for the whole image is obtained then the efficiency of the proposed technique will be very clear.

The implemented system was tested with the image of my daughter "Lulu" of size 250x400. The reduction in processing time of systolic design for a single window seems to be not so effective. But for the hole image the total processing time is effectively reduced from 91.157 ms for traditional design to 7.628 ms for improved systolic design. This illustrates the highly benefits of the proposed systolic technique for real time processing.

The tested image was exposed to salt and pepper noise with noise density: $d=0.1$. Figure 24-a shows the original clean image and the noisy image is shown in Figure 24-b. The median filter of two different lengths 3x3 and 5x5 are applied to the noisy image, and the o/p filtered images of each one are shown in Figure 24-c and Figure 24-d respectively. The images were resized here for presentation.

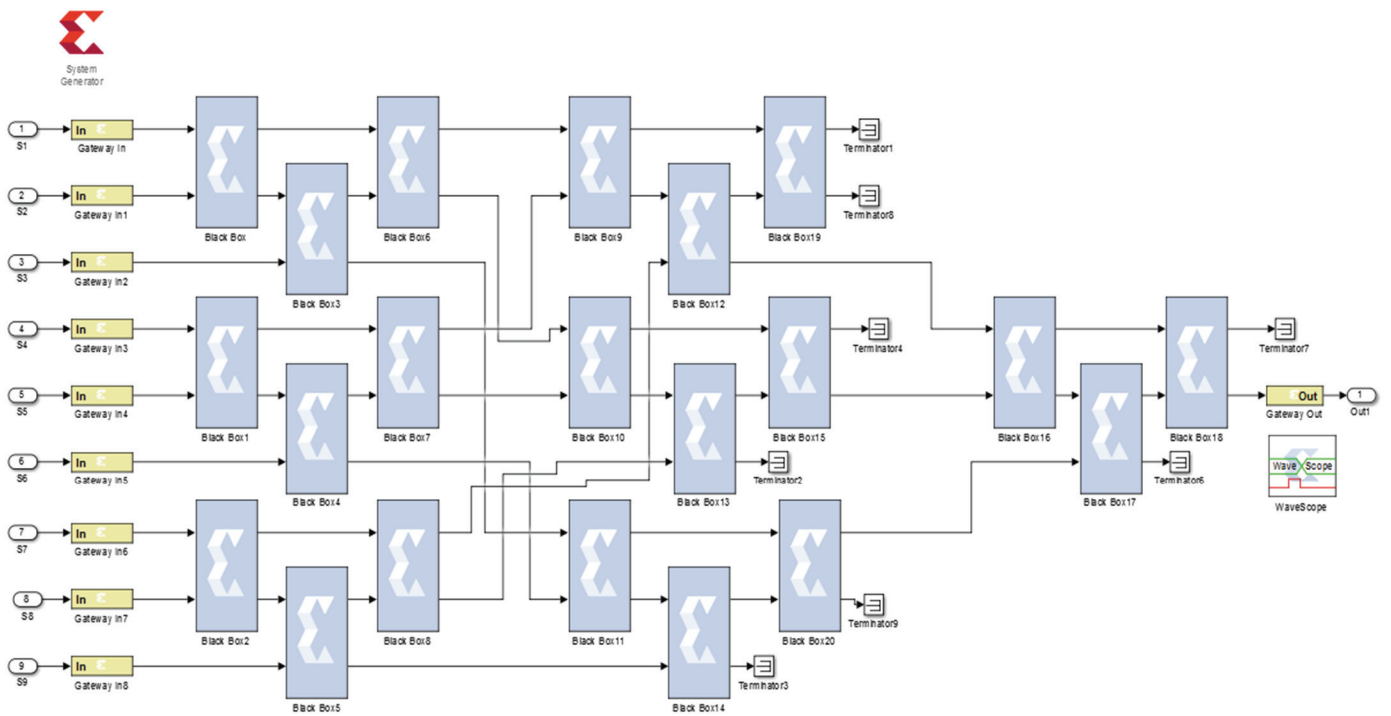


Figure16: Simulink design of 3*3 median filter using systolic array with 21 nodes.

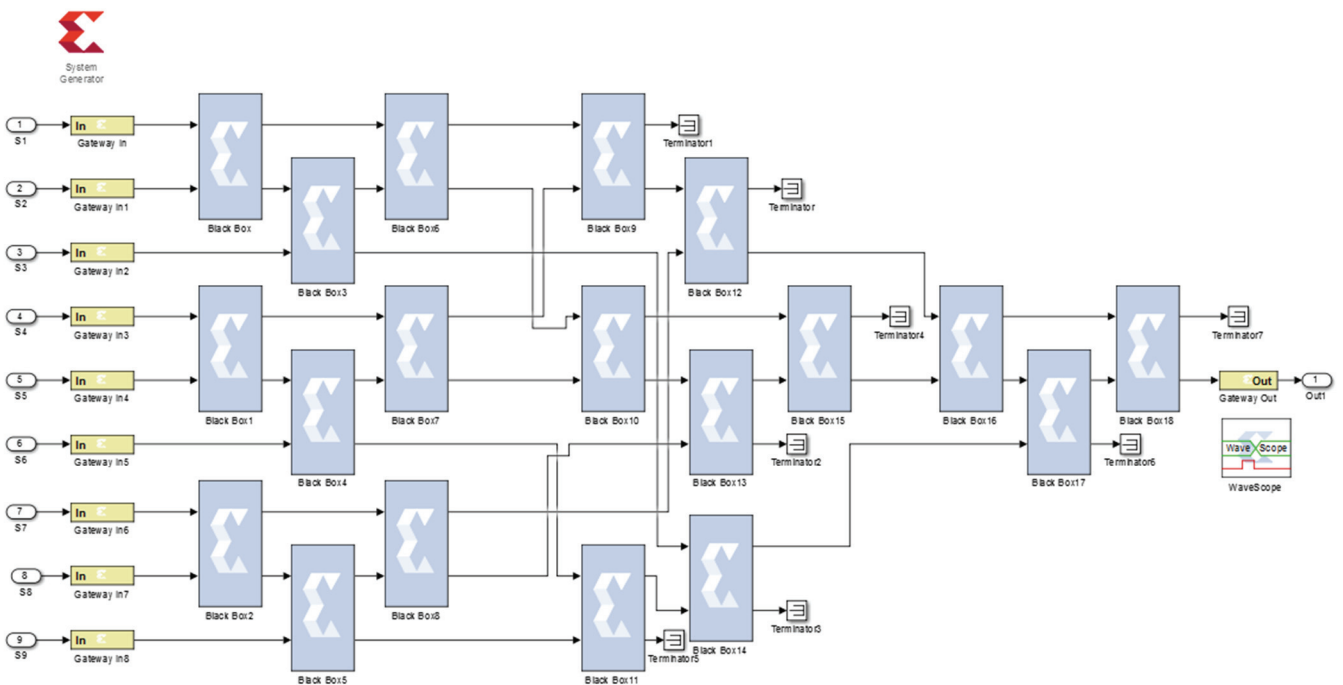


Figure17: Simulink design of 3*3 median filter using systolic array with 19 nodes.

Table 1: Traditional design of 3*3 median filter.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	0	54576	0%
Number of Slice LUT	728	27288	3%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 4: Systolic design of 3*3 median filter with 23 nodes.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	46	54576	< 1%
Number of Slice LUT	463	27288	1%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 2: Systolic design of 3*3 median filter with 36 nodes.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	73	54576	< 1%
Number of Slice LUT	704	27288	3%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 5: Systolic design of 3*3 median filter with 21 nodes

Logic Utilization	Used	Available	Utilizati .
Number of Slice Registers	43	54576	< 1%
Number of Slice LUT	439	27288	1%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 3: Systolic design of 3*3 median filter with 27 nodes

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	54	54576	< 1%
Number of Slice LUT	543	27288	2%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 6: Systolic design of 3*3 median filter with 19 nodes

Logic Utilization	Used	Available	Utiliz.
Number of Slice Registers	37	54576	< 1%
Number of Slice LUT	437	27288	1%
Number of bonded <u>IOBs</u>	81	296	27%
Number of BUFG/BUFGCTRLS	1	16	6%

Table 7: Design criteria of different window sizes of median filter (Traditional method).

Performance Criteria Window size	Used LUT	Storage capacity(MB)	T_p(μs)
3*3	728	142	0.911
5*5	6074	174	7.504
7*7	15768	202	21.912

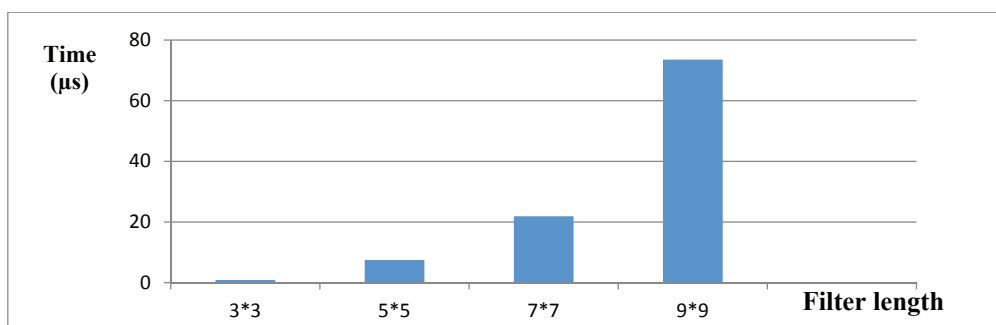


Figure 18 : Processing time for different filter length.

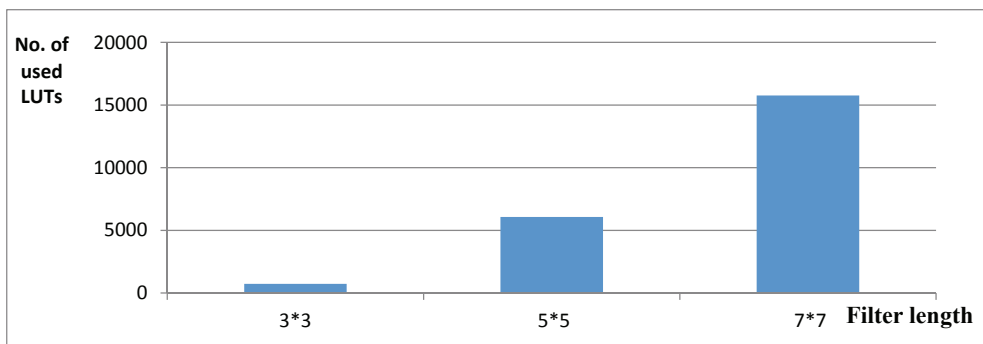


Figure 19: LUT for different filter length.

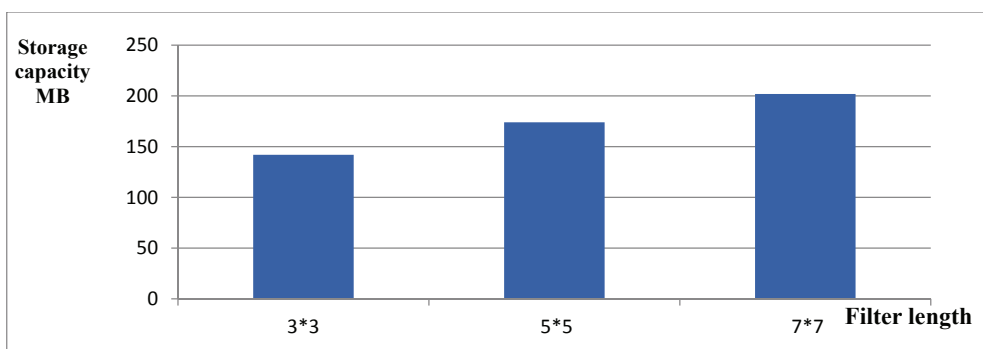


Figure 20: Storage capacity for different filter length.

Table 8: Design criteria of optimized systolic design of 3*3 median filter.

Performance Design Technique Criteria	Used LUT	Storage Memory MB	T _p (ns)
Traditional	728	142	911.576
36 nodes	704	140	734.190
27 nodes	543	139	721.865
23 nodes	463	139	704.025
21 nodes	439	139	698.257
19 nodes	437	139	686.495

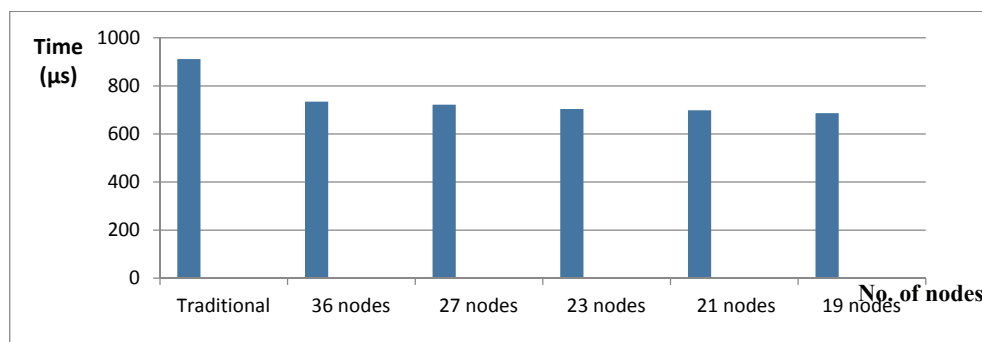


Figure 21: Processing time for different systolic design of 3*3 filter length.

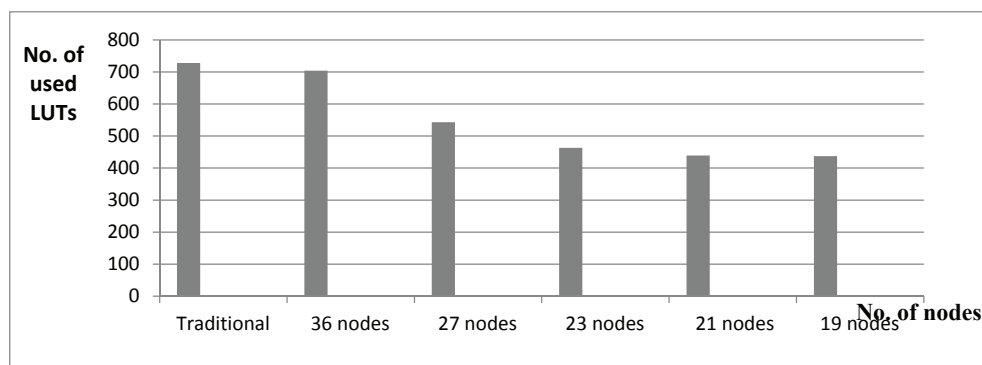


Figure 22: LUT for different systolic design of 3*3 filter length.

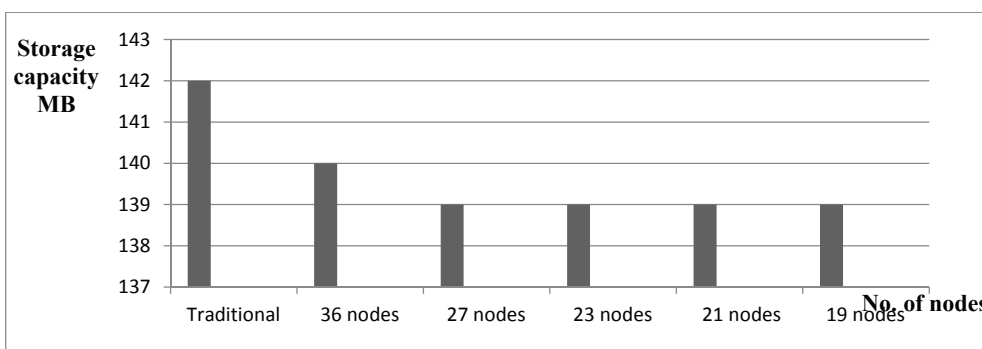


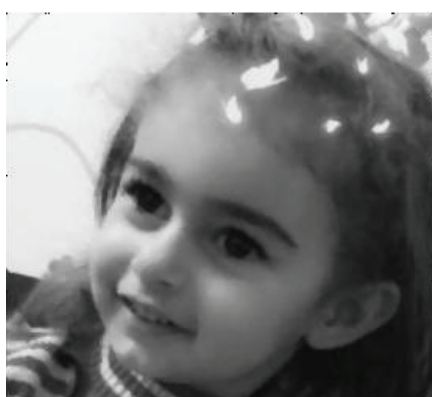
Figure 23: Storage capacity for different systolic design of 3*3 filter length.



(a): Original image.



(b): Noisy image, $d=0.1$.



(c): Filtered image of 3*3 filter length.



(d): Filtered image of 5*5 filter length.

Figure 24: The effect of filter length on image quality.

Important basic design principles must be established here. The researchers that interest with the development of median filter design must distinguish between the circuits that have single output which is the **median value** among several input data samples; and sorting circuits that their output is the sorted order of the values of all input samples. Hence, the design with traditional method and also systolic architecture designs of 36 nodes and 27 nodes are **sorting circuits**, produce all the sorted elements not only the median one; i.e, the output of these circuits is the sorted order of the input data. While the systolic architecture designs with 23 nodes, 21 nodes, and 19 nodes produce only the median elements as an output and cannot gives the sorted order of input data. Thus Table 1, Table2, and Table 3 reflect the hardware complexity of sorting circuits. But Table 4, Table 5, and Table 6 present the hardware required for median filtering circuits.

As illustrated previously, both of these two circuits types can be useful for median filter design, but the median value circuits are cost effective and give high performance parameters.

Finally, from the obtained results and graphs the following relationships have been proved :

$$T_p \propto L_f \quad \dots \dots \dots (3)$$

$$T_p \propto \frac{1}{D_p} \quad \dots \dots \dots (4)$$

$$C_{H/W} \propto L_f \quad \dots \dots \dots (5)$$

$$C_{H/W} \propto N_{node} \quad \dots \dots \dots (6)$$

where:

T_p : processing time,

L_f : filter length,

D_p : degree of parallelism,

$C_{H/W}$: hardware complexity and

N_{node} : number of required comparing nodes.

6. CONCLUSION

The median filter represents the most important part in many image denoising system, so a special attention must be given to its design and implementation.

The main weakness of the systolic array architectures design of median filter is the needing of more complex hardware compared with traditional design, because multi execution units must be run simultaneously. This problem solved easily in the present work by using an optimized architecture that cancel out any unimportant hardware components and also the components with repeated actions.

The cost problem of systolic array hardware is appeared at the first attempt only. Then, this problem can be highly eliminated for the improved and optimized systolic designs. So, for the first systolic design that includes 36 nodes for 3*3 median filter, the needed LUTs is 704, but this number would be efficiently reduced to 543 LUTs for systolic design with 27 nodes. Hardware components can be reduced further more for the design with 23 nodes, so the total number of the used LUTs become 463. The minimization of the needed hardware components was continued to be more effective. Thus, the number of needed LUTs could be reduced to 439 for the design with 21 nodes. Finally the needed LUTs is 437 only for systolic design with 19 nodes. These reductions are generalized to other hardware components such as occupied slices and registers. Hardware complexity can be highly enhanced, for median filter design, depending on the stage design approach. Thus the internal structure of each stage may be efficiently simplified.

The maximum used memory takes its chance of improvement also to be less and less with the optimized systolic array design, presented in this paper, since the number of intermediate results that must be stored will be decreased with decreasing the number of comparing nodes.

Execution time is a very important factor for median filter design, so its value must be reduced as minimum as possible while maintaining hardware simplicity. In fact simplifying the hardware complexity of systolic array architecture designs leads to an effective improvements in execution time, that depends mainly on the degree of parallelism of the developed system, to produce an excellent overall performance parameters of the implemented filter.

As its clear from the results, the execution time of median filter is considerably high with traditional design method, but with systolic array architecture the processing speed is highly improved and the execution time has been minimized to approximately 8% of its value with traditional method.

Looking at time charts, it is obvious that the execution time increases exponentially with the increasing of filter length, and also this exponential shape takes place when moving from traditional to systolic design for the same length. Finally, it is concluded that the speed of processing is related directly to filter length, as well as the needed hardware components are highly affected by this length, which may be increased rapidly to overcome the FPGA available components limits, but for high noise density large window size becomes essential demands.

REFERENCES

- [1] A. K. Jain, "Fundamentals of Digital Image Processing", *Prentice Hall of India*, First Edition, 1989.
- [2] R. Gonzalez and R.E. Woods, *Digital Image Processing*. Reading, MA: Prentice Hall, 3rd edition, 2007.
- [3] H. Hwang, R. Haddad, *Adaptive Median Filters: New Algorithms and Results*, *IEEE Trans. Image Processing*, Vol. 4, No. 4, 1995, pp. 499-502.
- [4] S. Singh and N. R. Prakash, "Modified Adaptive Median Filter for Salt & Pepper Noise", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 1, (2014) January.
- [5] S.S.Tavse, P.M. Jadhav, M.R. Ingle "Optimized Median Filter Implementation on FPGA Including Soft Processor" *IJEATE* Volume 2, Issue 8, August 2012).
- [6] V. Katkovnik, K. Egiazarian and J. Astola, "Adaptive Window Size Image De-noising Based on Intersection of Confidence Intervals (ICI) Rule", *Journal of Mathematical Imaging and Vision* vol. 16, (2002).
- [7] B. Draper, R. Beveridge, W. Böhm, C. Ross and M. Chawathe. "Implementing Image Applications on FPGAs," *International Conference on Pattern Recognition*, Quebec City, Aug. 11-15, 2002.
- [8] Linde, A., T. Nordstrom and M. Taveniku, "Using FPGAs to implement a reconfigurable highly parallel computer," *FPGA: Architectures and Tools for Rapid Prototyping; Selected papers from: Second International Workshop on Field-Programmable Logic and Applications (FPL'92)*, Springer-Verlag, pp. 199-210, 1992.
- [9] Introduction to Field Programmable Gate Arrays: <http://cas.web.cern.ch/cas/Sweden2007/Lectures/Web-versions/Serrano-1.pdf>
- [10] Xilinx Inc., FPGA vs. ASIC: <http://www.xilinx.com/company/gettingstarted/fpgavsasic>.
- [11] J. Villasenor and B. Hutchings, "The Flexibility of Configurable Computing," *IEEE Signal Processing Magazine*, Sept. 1998, pp. 67-84.
- [12] S. Hauck, "The Role of FPGAs in Reprogrammable Systems," in *Proceedings of the IEEE*, vol. 86, no. 4, April 1998, 615-638.

- [13] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm", IEEE Trans. Acoust., Speech, Signal Processing, vol. 27, no. 1, pp. 13–18, 1979.
- [14] Radhamadhab Dalai, "A Low Power Selective Median Filter Design", Master's Thesis, National Institute Of Technology, Rourkela, Department Of Computer Science And Engineering, June-2008.
- [15] A. H. Rasheed "FPGA Design and Implementation of Huge Data Systems", IJCA, Volume 174 – No.7, September 2017.
- [16] K. Benkrid, D.Crookes, A. Benkrid, "Design and Implementation of a Novel Algorithm for General Purpose Median Filtering on FPGAs", IEEE International Symposium on Circuits and Systems, Vol. 4, pp. IV-425-IV-428, 2002.
- [17] Gavin L.Bates and Saeid Nooshabadi, "FPGA Implementation of a Median Filter, "Proceedings of IEEE Speech and Image Technologies for Computing and Tele-communications, Vol. 2, pp. 437-440 vol.2, 1997
- [18] Miguel A. Vega-Rodriguez, "An FPGA Based Implementation for Median Filtering Meeting the Real-time Requirements of Automated Visual Inspection Systems," Proceedings of the 10th Mediterranean Conference on Control and Automation, July 2002.
- [19] Lu Gangl, Liang Yitao, "The Implementation and Analysis of Fast Median Filter Based on FPGA "Luoyang Institute of Science and Technology, Luoyang, China.
- [20] Siddarth Sharma, K. Pritamdas "FPGA Based Efficient Median Filter Implementation Using Xilinx System Generator", International Journal of Innovative Research in Science, Engineering and Technology, Vol. 5, Issue 5, May 2016.
- [21] S.S.Tavse, P.M.Jadhav, M.R.Ingle, "Optimized Median Filter Implementation on FPGA Including Soft Processor" IJETAE, Volume 2, Issue 8, August 2012.
- [22] A. N. Pimpale and Porf. Anoop Khambra, "Optimized Systolic Array Design for Median Filter in Image Filtration", International Journal of Electrical, Electron and Computer Engineering: 46-53(2016).