VLSI ARCHITECTURE OF DECISION BASED MODIFIED SELECTION SORT FILTER FOR SALT AND PEPPER NOISE REMOVAL

Vasanth K.

Sathyabama University, Chennai, Tamilnadu, India.
Email: vasanthece_k@yahoo.co.in

Abstract

A Novel architecture is proposed for the decision based modified selection sorting. The need for an optimized area, speed and power plays a vital role for any VLSI implementation of image processing hardware. The proposed architecture checks the given pixel is noisy or not and finds the median only if it is noisy. Under high noisy conditions the computed median might also be noisy hence arithmetic mean of uncorrupted pixels in the current window or replacement of neighborhood pixel over the processed pixel is done. The proposed architecture is compared with other decision based median finding architecture on the basis of power, speed, and area. The proposed architecture is targeted to Spartan 3e Device with gate capacity 5000 using Xilinx 7.1i compiler version. The proposed scheme is capable of operating at 233.318MHz requiring 2800 number of slices with a power dissipation of 100mw.

Keywords—Decision based median filters, modified selection sorting, salt and pepper noise

I. INTRODUCTION

Median filtering is a popular method of noise removal,employed extensively in applications involving speech and signal and image processing. This non-linear technique has proven to be a good alternative to linear filtering as it can effectively suppress impulse noise while preserve edge information.[1] [18]-[25]. These properties make it very popular filter in speech processing and image processing schemes. These are two types of linear filters. Non-recursive and recursive. In non-recursive median filtering, a window is moved along the sampled values of the image, and the center value of each window is replaced by the median of the values in the window. For instance in 2D non-recursive median filtering , the $(i,j)^{th}$ window of size(k*k), W_{ij}, is centered at (i,j) and the $(i,j)^{th}$ output y_{ij}= median $\{ w_{ij} \}$. In recursive median filtering, the window consists of recent median values as well as input values. In 1D recursive median filtering, the i^{th} window of size (2N+1), W_{ij} consists of (N+1) input values $x_i \ldots\ldots x_{i+N}$, and N output values $y_{i-N} \ldots\ldots y_{i-1}$; $wi = \{ y_i - N, \ldots\ldots y_{i-1}, x_i, \ldots\ldots x_{i+1} \}$ and the i^{th} output y_i=median$\{w_i\}$. The existing architecture for median filter can be broadly classified in to two classes. The array based architecture [2,3,4], and sorting network based architecture [5,6] gives an excellent survey of the existing architecture. The array based architecture consists of K processors of the windows is of size K, but they have a large sample period compared to sorting network based architecture. A good survey paper of VLSI median filters can be found in [7] where the author discussed hardware complexity in terms of number of samples 'N', word length 'I', and running size 'R'. In principle these digital algorithms and methods can be classified in to two categories: word-level and bit-level as discussed in[8] In this paper only word level median filters are studied since they offer high throughput capability as required in many real-time image/video systems. However a very cost-effective hardware solution to meet this goal is often difficult to achieve and hence system performance becomes degraded to allow trade-off between hardware cost and achievable performance. For example a fast median filter Based on the bubble sorting algorithm can be found in [9] By means of a set of processing elements or PE^s, the required values can be obtained with a latency of N cycles, where N is the number of input samples. Though this approach is fast, the size o the hardware implementation complexity is proportional to the square of the number of input samples. Hence hardware overheads increase rapidly with the number o input samples. In addition to this sorting kernel, it is necessary to provide extra hardware in the form of a data buffer to rearrange input samples for the parallel processing and hence increase the memory bandwidth. Another solution is a message passing method [10] Realized on a systolic array architecture [11] both deletion and insertion messages pass through the systolic arrays until certain conditions

are encountered. Although the hardware complexity depends on the number of input samples (N), the latency remains the same as that needed in the parallel bubble sorter. This latency of N cycles may not be allowed when real-time performance is concerned. Sorting is one of the most commonly used data processing applications as a fundamental operation on a computer system. Much effort has been devoted to find out faster sorting algorithms because of its practical importance as well as its theoretical interest [12] Proposed a parallel sorting algorithm based on a bitonic sequence and obtained an execution time of $O(log^2N)$ for N data using $O(log^2N)$ processors.[13] Also proposed a bitonic sorting with the perfect shuffle network and achieved an execution time of $O(log^2N)$ using $O(N\log^2N)$ processors.[14] Proposed another algorithm with $O(log^2N)$ time for $O(N\log N)$ processors. Horiguchi and sheigei [15] Proposed a parallel sorting algorithm with $O(N)$ time for $O(log N)$ linearly connected arrays. Thompson and Kung [16] And Nassimi and sahni [17] Extended Bacher's algorithm to a mesh connected arrays with N^2 Processors. They obtained the execution time of $O(N)$ for N^2 data. The revolutionary VLSI device technology has made practical and production of special purpose computing system with highly parallel structure. The systolic array is generally a set of relatively a simple processing unit of the same type, which are connected by a simple interconnection scheme and are able to be operated in parallel. The architecture serves a very high performance, because the primitive cells use data from neighbors without having to store and retrieve intermediate results. Section II deals with the implementation of proposed algorithm. Section III deals with simulation results Section IV deals with conclusions.

II. IMPLEMENTATION OF PROPOSED ALGORITHM

The proposed algorithm is given as follows:

Step 1. A 2-D window of size 3*3 is selected. Assume the pixel to be processed is P(X, Y).

Step 2. The pixel intensities of the window considered are converted into an 1D array of size 9.

Step 3. The pixel with maximum intensity is propagated to the final array position of the input data by the process of swapping as shown in figure 1. This gives P_{max}.

Step 4. The pixel with minimum intensity is propagated to the last but one position just next to P_{max} of the array by the process of swapping the array elements, excluding P_{max}. This gives Pmin as shown in figure 1.

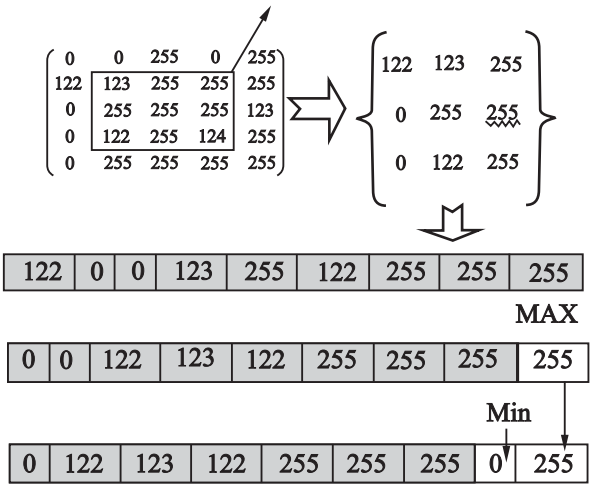
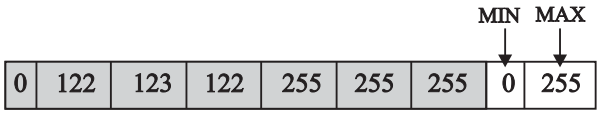


Fig. 1. Illustration of the proposed algorithm to find Pmin and P_{max}

Step 1: Convert the matrix into 1D array

Step 2: Propagate the maximum pixel intensity to the 9^{th} array position considering the above array as input

Step 3: Propagate the maximum pixel intensity to the 8^{th} array position considering the first 8 pixel intensities of the above array as input



Step 4:

Case 1. P(X, Y) is an uncorrupted pixel, if $P_{min} < P(X, Y) < P_{max}$; the pixel being Processed is left unchanged. This case does not involve the computation of the median. Otherwise, $P(X, Y)$ is a corrupted Pixel. The median is computed only when the processed pixel is noisy.

Case 2. If $P(X, Y)$ is a corrupted pixel, the median is computed as follows. To find the median Pmed, swap the remaining unsorted array elements obtained from step 3 , excluding P_{max} and P_{min} for four passes as shown in figure 2. After each pass, the smallest element encountered in the current pass will reside in the last position traversed. So each pass can be one step shorter than the previous pass, instead of every pass continuing to traverse all the elements at the end, which are already in their final positions and will not move in any case. After the 4^{th} pass, the pixel in the 4th position will give the median of the window as illustrated in the figure.2. The corrupted pixel is replaced by its median value.

The corrupted pixel is replaced by its median value. For high noise densities the manipulated median may also be noisy. So check the calculated median is noisy or not.If $P_{min} < P_{med} < P_{max}$ and $0 < P_{med} < 255$. then Pmed is a uncorrupted pixel, replace on the processed pixel.

Step 4 : Case (2)

Input array to find the median excluding min and max pixel intensities (i.e) the first seven elements of the array The median found in the 4^{th} pass, which is the final pass. The smallest element encountered in the current pass will reside in the last position traversed. So each pass is one step shorter than the previous pass.

122	123	122	255	255	255	0	0	255
123	122	255	255	255	122	0	0	255
123	255	255	255	122	122	0	0	255
255	255	255	123	122	122	0	0	255

MED

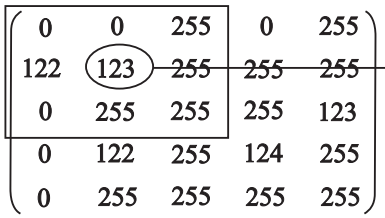
Fig. 2 Illustration of the proposed algorithm to find P_{med}.

Case 3. If $P_{min} < P_{med} < P_{max}$ is not satisfied or $255 < P_{med} < 0$, then P_{med} is a noisy pixel. In this case, the $P(X, Y)$ is replaced by the average of the non- noisy pixels in the window considered. These pixels must satisfy the condition, min < pixel intensity < max. only those pixels satisfying the above condition is considered as non noisy or noise free pixel of the current processing window. When no non noisy pixel is presented then go to case4

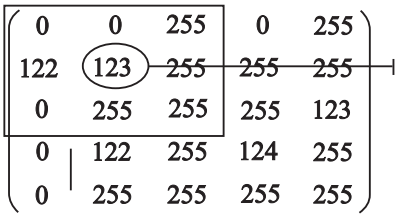
Case 4. If there are no uncorrupted pixels in the window,replace the corrupted pixel with the neighborhood pixel.

Step 5. Steps 1 to 5 are repeated until the processing is completed for the entire image. The proposed algorithm is illustrated in figure 1-3 [20].

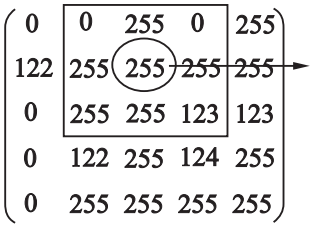
Case 1:



Case 2



Case 3



(0 < 123 < 255)
Median is also noisy. Replace processed pixel by arithmetic mean of non noisy pixel inside the window. Here only one non noisy pixel i.e. 123.

Case 4

0	0	255	0	255
122	123	123	255	255
0	255	255	255	255
0	122	255	124	255
0	255	255	255	255

$(0 < 255 < 255)$ center pixel is noisy, so the median is found 255.

$(0 < 255 < 255)$ Median is also noisy. There is no non noisy pixel in the window. Replace by left neighborhood from the last processed value.

III HARDWARE IMPLEMENTATION

We propose a sequential architecture for the proposed algorithm which uses 3x3 spatial window as shown in figure 3. The 2D data acquired from the window is converted to 1D.The architecture is split up into various modules such as max_unit, Min Unit, Median Unit and decision unit. Max and Min unit computes maximum and minimum values of the array in a 3x3 window.

Max_unit: The proposed architecture consists of max unit that manipulates the maximum number of the given array. This unit compares each input with each other and swaps subsequently to obtain the maximum value of the array. The manipulated maximum value is placed in the last position of the input array. It is termed as Pmax

Min_Unit: The proposed architecture consists of min unit that manipulates the minimum number of the given array. The max unit finds maximum value of the input array; hence to find minimum value of the sorted array we consider only the remaining 8 values. This unit compares each input with each other and swaps subsequently to obtain the minimum value of the array. The manipulated minimum value is placed in the last but one position of the input array. It is termed as Pmin

Median_Unit: This unit arranges the remaining five inputs in ascending by the process of compare and

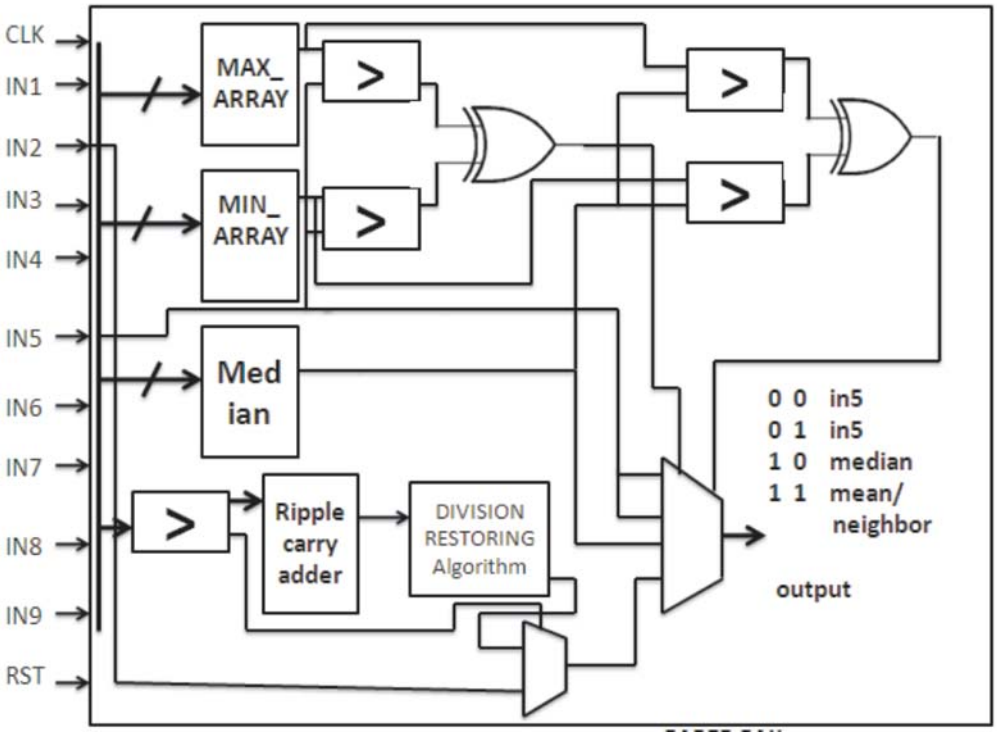


Fig. 3 Architecture for the proposed algorithm

swapping i.e to find the median Pmed, swap the remaining unsorted array elements obtained excluding Pmax and Pmin for four passes. After each pass, the smallest element encountered in the current pass will reside in the last position traversed. So each pass can be one step shorter than the previous pass, instead of every pass continuing to traverse all the elements at the end, which are already in their final positions and will not move in any case. After the 4th pass, the pixel in the 4^{th} position will give the median of the window. The median is evaluated only if the decision unit decides the processed pixels "in5" is noisy.

Decision unit: This unit compares the processed pixel "in5"with the maximum and minimum value of the given array. If the value lies within the range then the pixel is considered as non noisy. If the value lies outside this range then pixel is considered as noisy and a control signal is summoned to perform the median of the remaining output. Then the processed median is compared with the minimum and maximum values of the current processing window. If the median lies outside the range it is considered as noisy; hence evaluate the mean of the non noisy pixel inside the current processing window. At higher noisy conditions the processing window does not have any non noisy pixels hence the neighborhood pixel is replaced in the output as shown in the table 1.

Table 1. Combination of decision performed

Value from Decision	Value from II Decision	Action Performed	Explanation
0	0	In 5	Pixel is not noisy and the processed pixel is retained.
0	1	In 5	
1	0	Median	Processed pixel is Inoisty, but evaluated median is not noisy.
1	1	Mean/ Neighbor	Processed pixel is noisy, processed median is also noisy.

Mean unit: This unit checks each element of an array for the presence of the outliers. If pixel is not noisy then it is accumulated in an accumulator and latter fed as input to 8 bit ripple carry adder. After checking all the pixels of an array, division is implemented using non restoring algorithm. Initially there is a compare unit before this mean unit. It decides the number of noisy pixels. Thereby the number of noisy pixel is fed as an input to a decision unit to choose Arithmetic mean of non noisy pixels/ Neighborhood pixel that has to be replaced in the output. If noisy pixel is present then the noisy counter increments by 1. If this value is non zero then it means there are no non noisy pixels; hence the output is neighborhood value. If there are non noisy pixels then arithmetic mean of the non noisy pixels is sent out.

IV. SIMULATION RESULTS

The proposed architecture is implemented for XC3s5000-5fg900 using Xilinx 7.1 compiler tool for synthesis and modelsim 5.8lli for simulation as a third party tool using VHDL. All the median finding sorting algorithms have been implemented as described. First the individual sorting followed by the decision to estimate the pixel is noisy or not. But the proposed architecture finds median only when the pixel is noisy. Table 2 illustrates the comparison of other decision based sorting techniques with the proposed logic on the basis of area, speed, power. Figure 4 gives the utilization of number of slices for various algorithms. Figure 5 gives the utilization of 4 input look up table by various algorithms. Figure 6 denotes the comparison of number of slices utilized for various algorithms after place and route. Figure 7,8 illustrates the minimum period and operating frequency of various algorithms. Figure 9 implies the power used in each algorithms. Figure 10-14 gives the simulation result of various decision based sorting technique. Figure 15- 19 depicits the floor plan for the various decision based sorting techniques. Figure 20-24 denoted the routed FPGA for various decision based sorting logic.

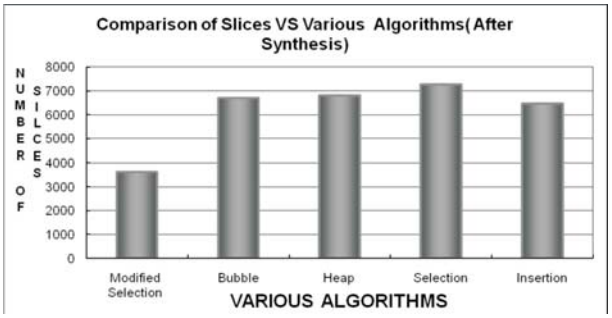


Fig. 4. Comparison of slices utilized vs various algorithms after synthesis

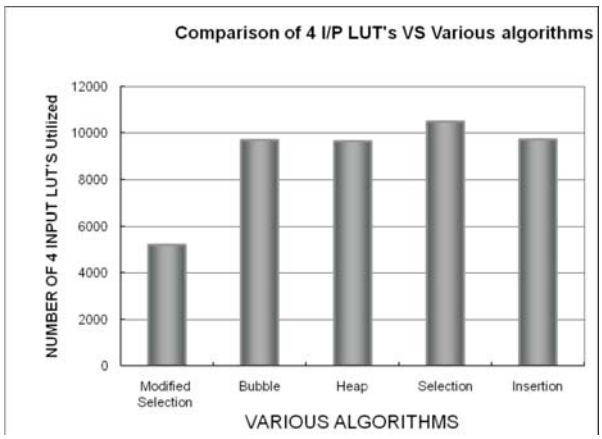


Fig. 5. Comparison of 4 input LUT's vs various algorithms after synthesis

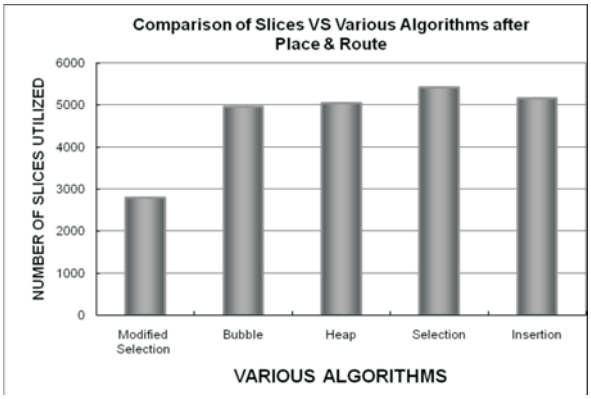


Fig. 6. Comparison of number of slices utilized vs various algorithms after place & Route

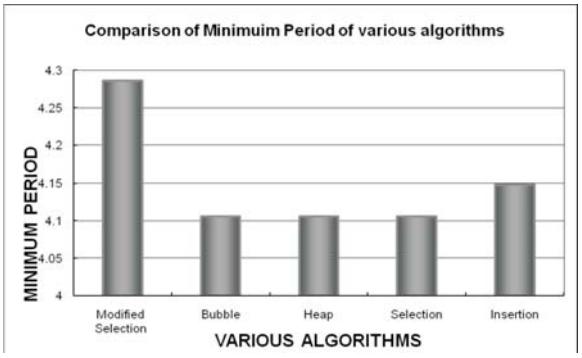


Fig. 7. Comparison of minimum period required by various algorithms

Table 2. Comparison of Decision based Median finding algorithms for the target FPGA Xc3s5000-5fg900

No.	Parameters	Insertion	Bubble	Heap	Selection	Modified Selection
1.	Slices	6479	6712	6814	727	3629
2.	4ip LUP	9738	9712	9635	10508	5209
3.	Bonded IOB	329	329	329	329	329
4.	Gdl	1	1	1	1	1
	Devicve Utilization Factor After place and Route					
1.	Cdl	–	–	–	–	–
2.	External GCLKIOBs	–	–	–	–	–
3.	External IOBs	329	329	329	329	329
4.	LOCed IOBs	0	0	0	0	0
5.	LOCed GCLKIOBs	0	0	0	0	0
6.	Slices	5151	4958	5049	5418	2800
	Triming Specifications before Place and Route					
1.	Minimum pepriod (ns)	4.148	4.106	4.106	4.106	4.286
2.	Operating Frequency	241.08	243.546	243.546	243.546	233.318
3.	Minimum inputarinal before the clock	257.78ns	192.590 ns	360.531ns	439.086ns	219.208ns
4.	Maximum output required time after clock	7.16ns	7.16ns	7.16ns	7.16ns	7.16ns
	Power Consumptioin					
1.	Power (unit)	298	298	298	298	100

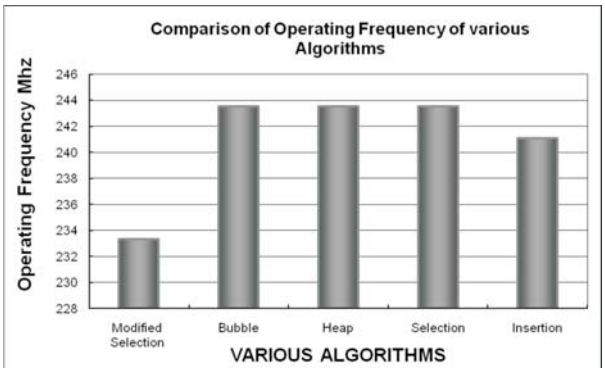


Fig. 8. Comparison of operating frequency required by various algorithms

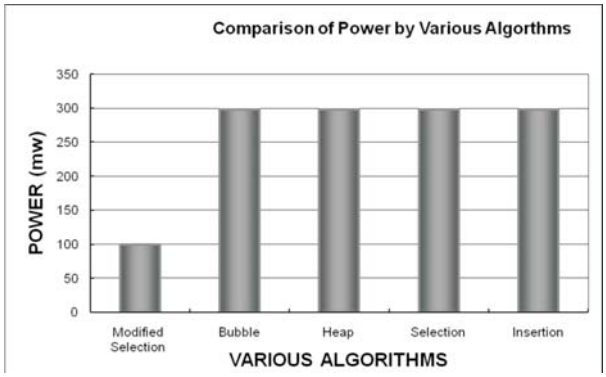


Fig. 9. Comparison of power vs various algorithms

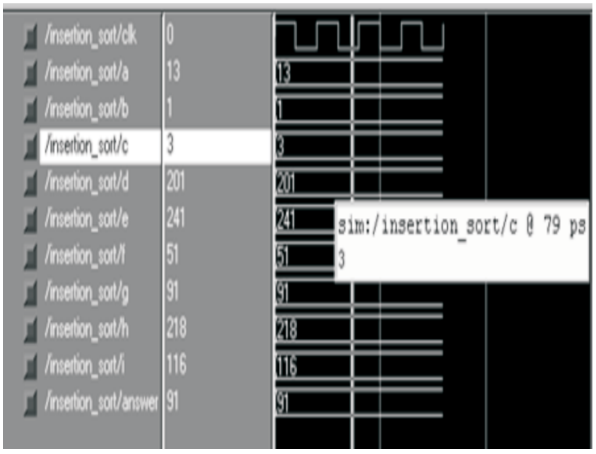


Fig. 10 Simulation result of decision based Insertion sorting

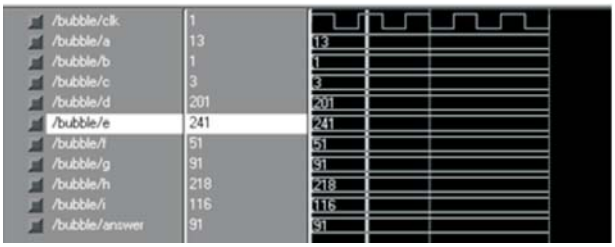


Fig.11 Simulation result of decision based bubble sorting

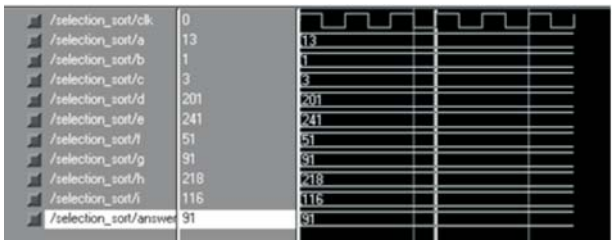


Fig.12 Simulation result of decision based Selection sorting

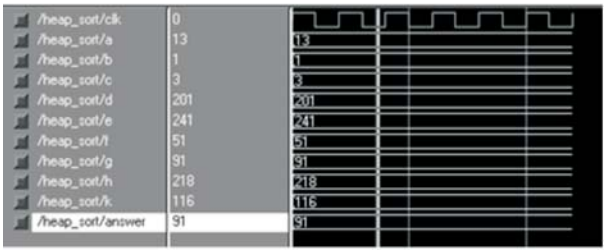


Fig.13 Simulation result of decision based Heap sorting

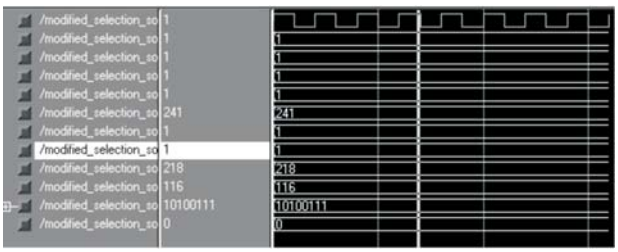


Fig.14 Simulation result of decision based Modified Selection sorting

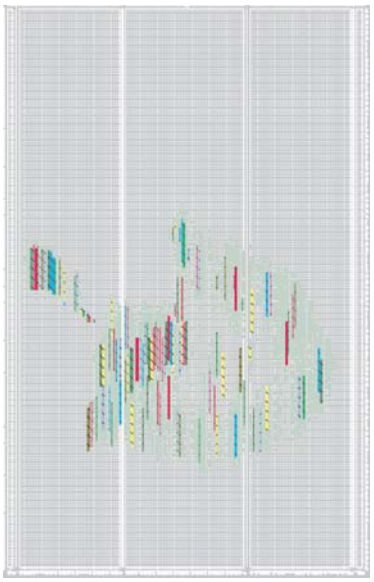


Fig.15 Floor plan of decision based Insertion sorting

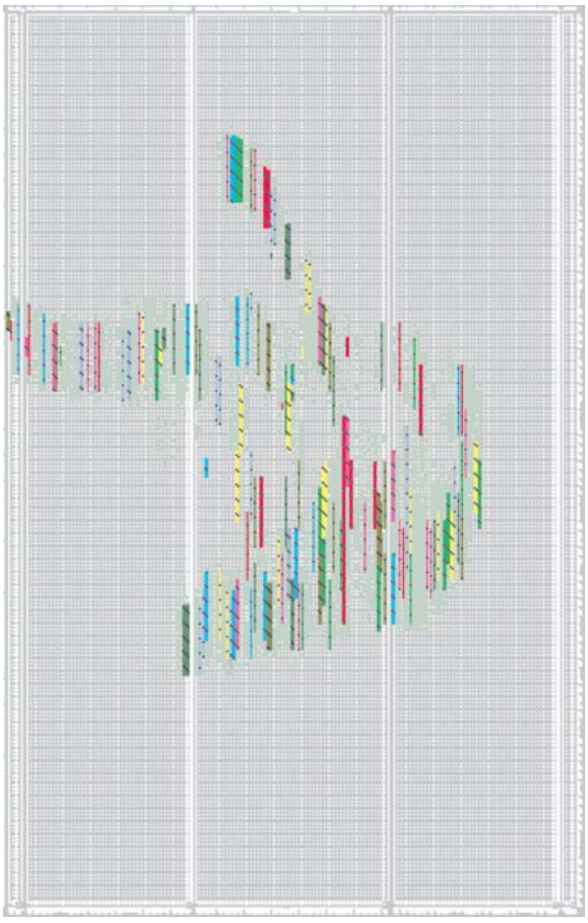


Fig.16 Floor plan of decision based Bubble sorting

Fig.17 Floor plan of decision based Heap sorting

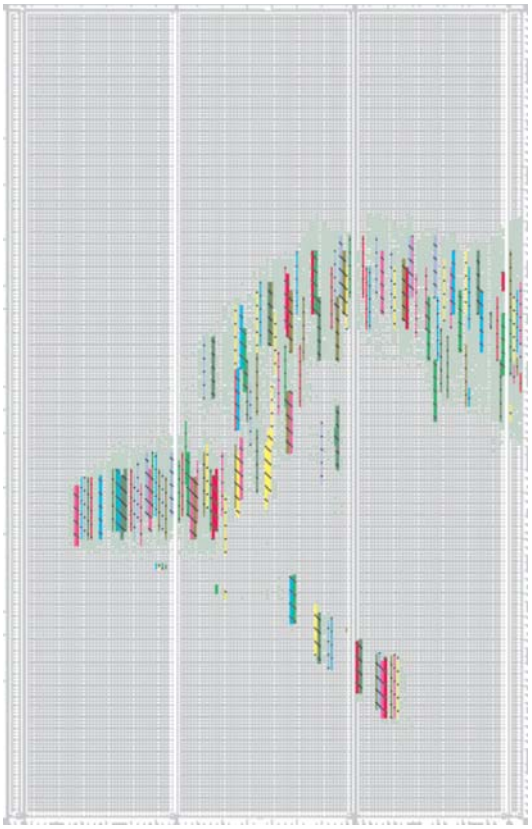


Fig.18 Floor plan of decision based Selection sorting

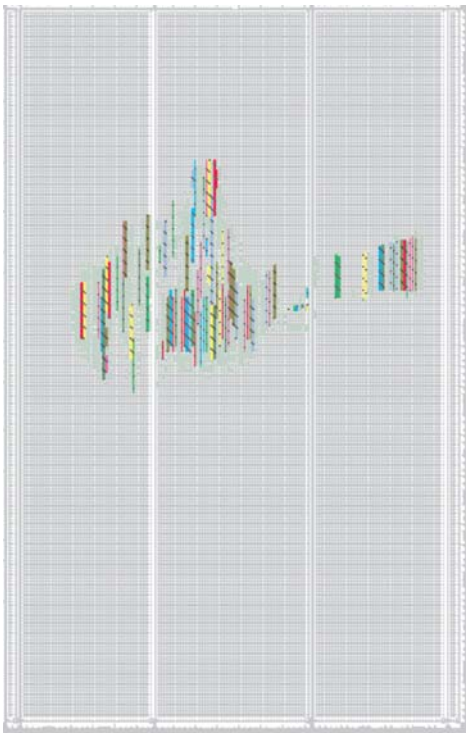


Fig.19 Floor plan of decision based Modified Selection sorting

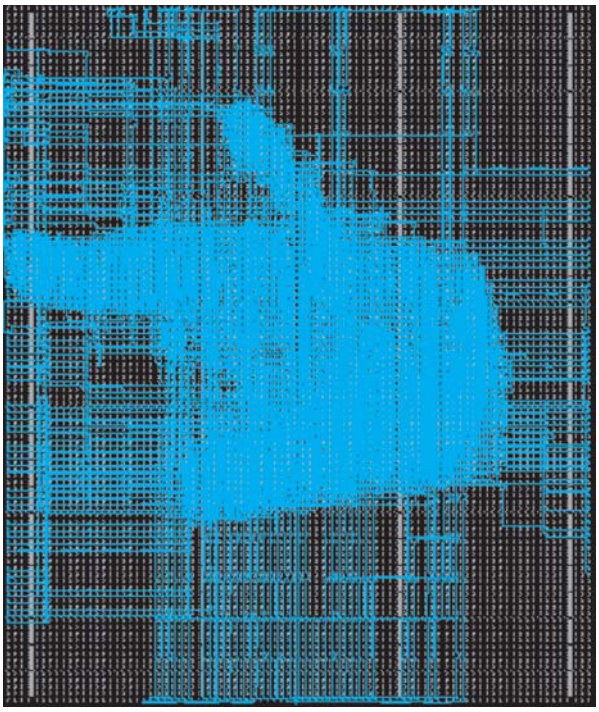


Fig.21 Routed FPGA of decision based Bubble sorting

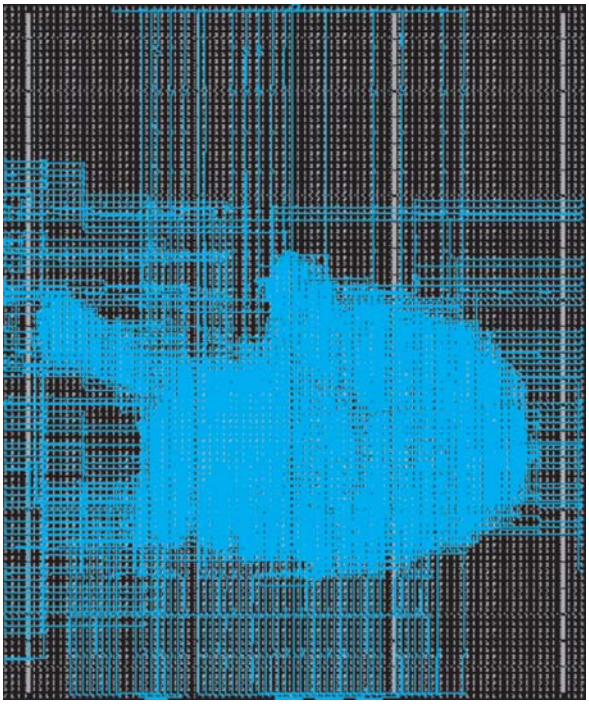


Fig.20 Routed FPGA of decision based insertion sorting

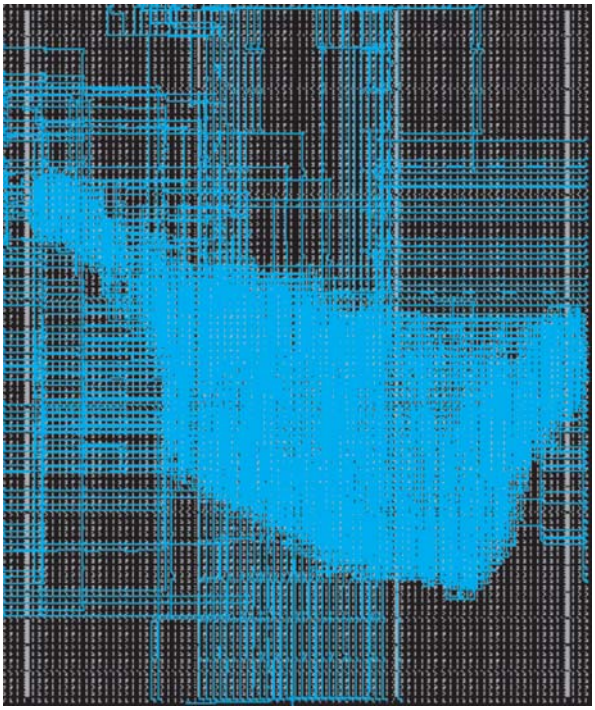


Fig.22 Routed FPGA of decision based Heap sorting

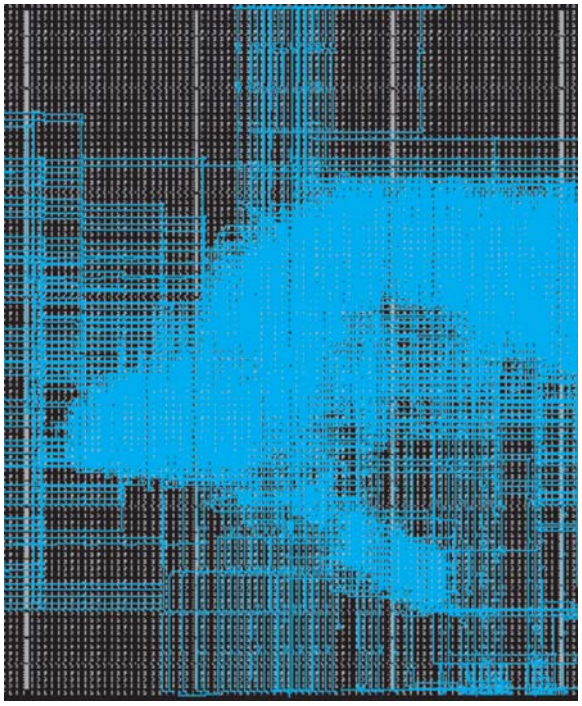


Fig.23 Routed FPGA of decision based Selection sorting

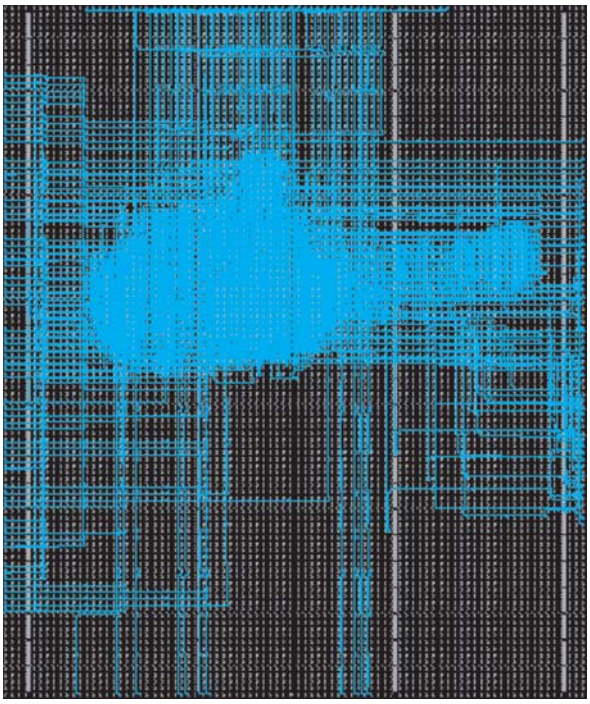


Fig.24 Routed FPGA of decision based Modified Selection sorting

The architecture employs modified selection sort to compute the median. The sum of non noisy pixels is done using a simple ripple carry adder and the result is divided by the number of non noisy pixels using non restoring algorithm. The decision unit employs a multiplexer which replaces the suitable output after classifying the pixel is noisy or not.

V. CONCLUSION

In this paper a novel architecture has been proposed which employs sequential processing of processing elements such as maximum finding minimum finding logic, comparator to check the processed pixel is noisy or not then a comparison logic to identify the non noisy pixels in the current window and adder to add the non noisy pixel and a divider to find the arithmetic mean of non noisy pixel in the given window or replacing the neighborhood pixel which results in switched median filter. The design of the proposed algorithm has the ability to exploit the certain features of 3×3 window. We have implemented the proposed switched median filter for the target device XC3s5000-5FG900 given in table1. the first row illustrates the number of slices utilized by the proposed algorithm. The proposed architecture requires 3269 slices for implementation after synthesis. The second row illustrates the number of 4 input look up table required by the proposed algorithm for implementation and it requires 2800 4 input look up table. Row ten of the table 1 gives the number of slices required by the proposed algorithm after place and route. The proposed architecture required 2800 slices for the logic. The row eleven illustrates the minimum period which is 4.26ns. The proposed logic works at 233.318 MHz. with the input arrival time of 219.208ns and a maximum output required time after clock is 7.165ns. The last row illustrates the low power required by the architecture which is 100mw. From the figure 4,5,6 we conclude the proposed logic requires 3269 slices, 5209 four input look up tables (After synthesis) and 2800 slices (after place and route) which is very less when compared to the other compared architectures. From the figure 7 we understand the minimum period required by the proposed logic is 4.286ns which is optimum when compared with the other architectures that are compared. Figure 8 it is evident that the operating frequency of the proposed logic is an optimum 233.318 MHz when compared to the other decision based sorting techniques discussed here. Figure 9 illustrates

the proposed architecture requires100mw. The required power is very less when compared to other architectures. From the table and the graph we conclude that a reduced area, low power architecture with optimum speed is proposed.

REFERENCES

[1] Oflazer K., 1983, "Design and implementation of a simple chip ID median filter, "IEEE trans on acoustics, speech and signal processing, Assp-30, pp1164-1168.

[2] Fisher A., 1982, "Systolic Algorithm for running order statistics in signal and image" , Digital system, volume 4, pp 251-264.

[3] Hwang J.N. et al. 1990, "Systolic architecture for 2-D rank order filtering , "Proc International conference Application specific Array processor, pp-90-99.

[4] Kung S.Y., 1989 "VLSI Array Processor", prentice Hall,.

[5] Karaman M., Onural L., Atalar A., 1988, "Design and implementation of general purpose median filter in VLSI," VLSI signal processing III, pp 111-119.

[6] Lucke L., :Parli K., 1992, "Parallel Structures for rank order & stack filters", Proc IEEE international conference. circuits & systems,.

[7] Richards. D.S., Jan 1982 "VLSI Median Filters", IEEE Computer, volume. 15, no.1,.

[8]Lee. C.L. and Jen C.W., Feb 1992 "Bit-sliced median filter design based on a majority gate", IEE proc-G V 139 no.1, pp.63-71,.

[9] Offen. J. and Raymond R., 1985. "VLSI Image Processing", McGraw-hill,

[10] Fisher. A.L., Systolic Algorithms for Running Order Statistics", in signal and image processing, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, Jul.1981.

[11] Kung. H.T., Jan 1982 "Why Systolic Architectures", IEEE Computer, vol. 15, no.1,.

[12] Batcher. K.E., "Sorting Network and their applications", Proc. AFIPS Conf. Vol. 32, pp.307-314 (1968).

[13] Stone. H.S.,1971 "Parallel Processing with the perfect Shuffle," IEEE Trans. Computer. Vol. C-20, No.2, pp.153-161.

[14] Preparata. F.P.,1978 "New Parallel Sorting Schemes," IEEE Trans. Computer, Vol.C-27, No.7, pp.669-673.

[15] Horiguchi. S. and sheigei Y., 1986). "Parallel sorting algorithm for a linearly connected multiprocessor system", Proc. Int'l Conf.Distributed computing systems, pp.111-118.

[16] Thompson. C.D. and Kung H.T., 1981 "sorting on a mesh-connected parallel computer," Comm. ACM, vol.20, pp.151-161.

[17] Nassimi. D. and Sahni S., 1979 "Bitonics sort on a mesh-connected parallel computer", IEEE Trans. Computer, volC-27, pp.2-7

[18] Srinivasan . K.S. et al., 2007. A New Fast and Efficient Decision- Based Algorithm for Removal of High-Density Impulse Noises, IEEE Signal Processing Letters, Vol.14, No.6,.

[19] Vasanth K., Karthik. S., "A New class of decomposition algorithm for the reduction of low density impulse noise", international conference on ARTCOM2009, kerala, India, pages 203-207.

[20] Vasanth K., Karthik. S., "A switched Algorithm using modified selection sort for the reduction of impulse noise", ICTACT international journal on Image and video processing, Vol1, issue1, pages 57-64.

[21] Vasanth K., Karthik.S., "Performance Analysis of modified decomposition filter for non identical noises", ICTACT international journal on Image and video processing, Vol1, issue2, pages 105-115.

[22] Vasanth K., Karthik.S., "FPGA implementation of modified decomposition filter", international conference on Signal and image processing, RMD Engg college, Tamilnadu, India

[23] Vasanth K., Karthik S., Preetha mol.P, "Hybrid cascaded algorithm for impulse noise removal", National conference on CSE,NCSE2010, Sathyabama university, Tamilnadu, India,pages 132-136

[24] Vasanth K., Karthik.S., " A Study of median filter and its variants for impulse noise removal", National conference on E.E.E,NEEE-2010, Sathyabama university, Tamilnadu, India.

[25] Vasanth K., Karthik S., Nirmal raj S., Preetha mol.P, "FPGA implementation of optimized sorting networks for median filter",INTERACT-2010,International conference on Robotics and automation", Sathyabama university, Tamilnadu, India