Journal of VLSI Design and Signal Processing
e-ISSN: 2581-8449
Volume 5 Issue 1

An Improved Fault Tolerant Technique of Median Filter

Linda Babu[1], Gnana Sheela K[2]*
[1]*PG scholar, Department of Electronics & Communication Engineering, APJ Abdul Kalam Technological University, Trivandrum, Kerala, India*
[2]*Professor, Department of Electronics & Communication Engineering, APJ Abdul Kalam Technological University, Trivandrum, Kerala, India*
***Email:**sheelabijins@gmail.com*
***DOI:** http://doi.org/10.5281/zenodo.2549248*

Abstract
Acquisition noises in the digital image processing system basically made out of imprudent clamors, for example, hot and dead pixels, and for the most part expelled utilizing middle channels. The median filtering algorithm can be speedup by FPGA implementation. Configuration memory cells in SRAM based FPGAs are susceptible to radiation effects such as SEUs which leads to configuration memory bit flips and hence a protective measure is required for the proper operation of median filtering algorithm.The fault tolerant implementations of median filter provides a range for median value with which the calculated median value is checked and find out error if the median is out of the provided range. The main aim of the project is to fasten up the fault tolerant implementation of median filter in FPGAs by adding a few resources. Experimental results show that the proposed technique significantly reduces the latency of the fault tolerant median filtering process.

***Keywords:** Acquisition noises, Impulsive noises, Median filter, FPGA, SRAM, SEUs, Configuration memory*

INTRODUCTION

Image processing is one of the rapidly growing technologies in this era, which is used to perform some operations on an image to get an enhanced image. Digital image processing is the commonly used method for different space as well as medical image processing applications because of their attractive features compared to analog image processing. It enables a wide scope of calculations to be connected on the info picture and can stay away from issues, for example, the impact of clamor and flag mutilation amid the handling of picture [1].

In digital image processing, the captured image may acquire undesirable noise during the image acquisition stage. These impulsive noises are commonly referred to as salt and pepper noise. It is important to remove these short noises from the image as soon as it captured, so as to avoid the transmission of the corrupted image to the subsequent stages. Median filters are the best remedial for the removal of salt and pepper noises because of their edge preserving ability while removing noise. Figure 1 shows the effect of noises on images and the effect of filtering on noise affected images [2–5].

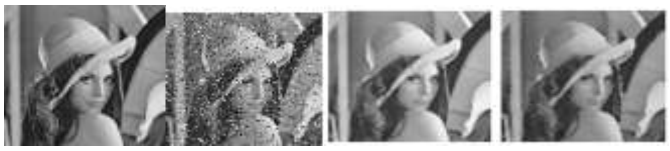

(a)(b)(c) (d)

Figure 1:*Effect of Salt and Pepper Noise on Images and Noise Removal using Median Filter. (a)Original Image, (b) Noise Affected Image, (c)Median Filtered Original Image, (d) Median Filtered Noise Affected Image.*

As these processes include a large number of mathematical operations on each image pixels, FPGAs become the best platform for implementing this algorithm on microprocessors. However, as SRAM based FPGAs are sensitive to radiation noises, the implemented median filtering algorithm can be altered by some SEUs thereby changing the functionalities of median filtering algorithm. It explains the need for a protection technique for the proper operation of median filter implementation in FPGAs. The fault tolerant implementation of median filter finds out the corrupted pixel present in the image. It is done by checking whether the calculated median value is within a prescribed range, or not. If not, an output error signal is generated as it detects a corrupted image pixel. This helps to speed up the overall image processing operations which in turn helps in large saving on time. This project is mainly focused on the speed of the processing system. As latency is an important factor of all systems, the proposed system is more efficient than the previous methods for median filtering.

METHODOLOGY

The proposed technique is an improved version of fault tolerant median filter, for providing low latency. The fault detection is performed similar to the fault tolerant method. The modification is done in the sorting and sliding section. To understand the basic fault tolerant median filtering process, a schematic diagram is shown in Figure 2.

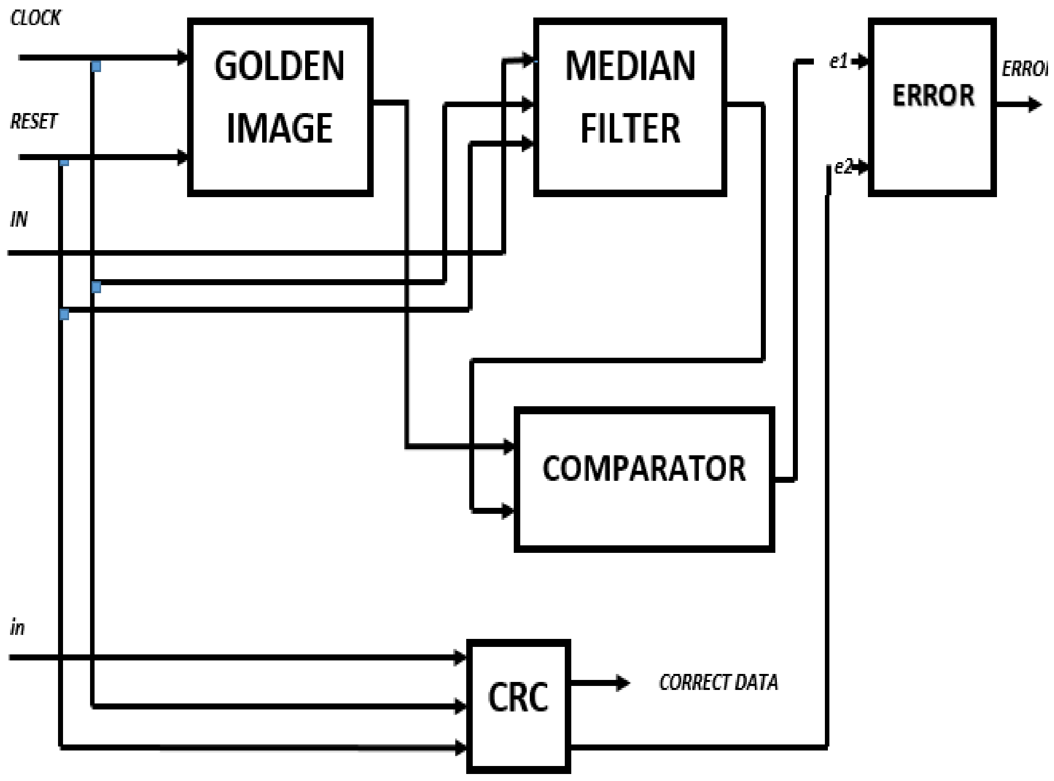


Figure 2:*Schematic Diagram of Fault Tolerant System.*

The fault tolerant median filter checks for two types of error. First, the median filtered image compares with the golden image i.e. the test image, and produce error ‘e1’ signal if any mismatches has occurred. The CRC error detection and correction module produces error signal ‘e2’ for any error in data transferring, to ensure the accuracy of data transmission. The basic blocks of median filer are shown in Figure 3.

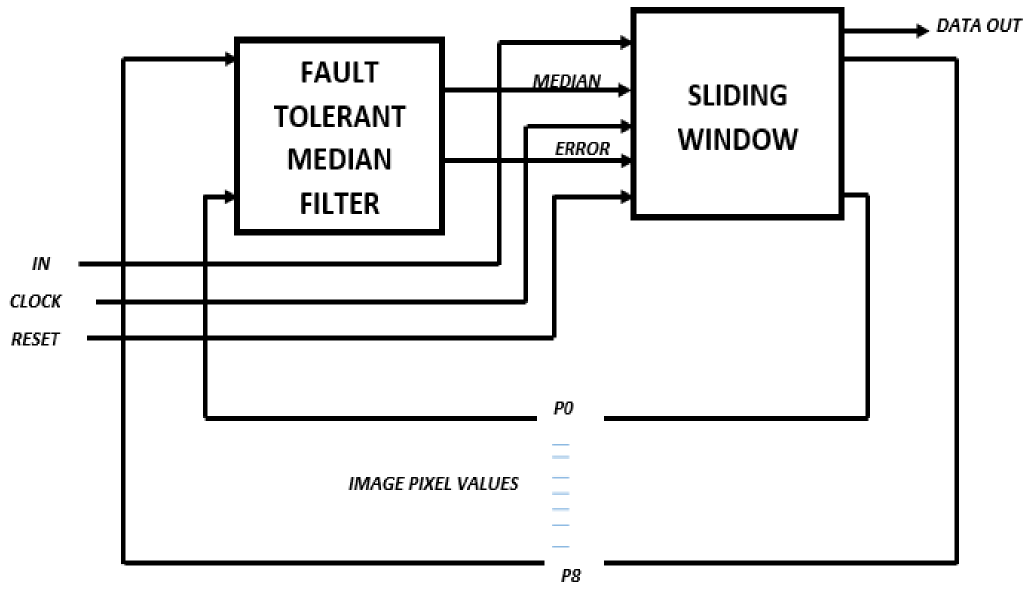


Figure 3:*Block Diagram of Median Filter.*

To understand the functionality of the system, it is required to learnabout the basic operations of each block. The theory and working of median filter and different sorting algorithms have explained below.

SORTING NETWORKS FOR MEDIAN FILTERING

A sorting network mainly consider as a multi-input, multi-output switching network. It requires less number of hardware to perform a large number of computations. The attractive part is that, it can be have thousands of inputs and outputs by using only a small number of hardware. Sorting memories have other applications such as a multi access memory, a switching network with buffering and as a multiprocessor. The proposed technique is used the sorting network with minimum number of exchange nodes, for sorting the 9 pixel values, obtainedfrom the 3×3 sliding window matrix [6,7].

The Figure 4 shows the basic exchange node for a sorting system. It includes two multiplexers and a comparator. The comparator is 8 bit and the multiplexers used here are 2:1 mux. At a time 2 inputs are compared and higher and lower values are generated.

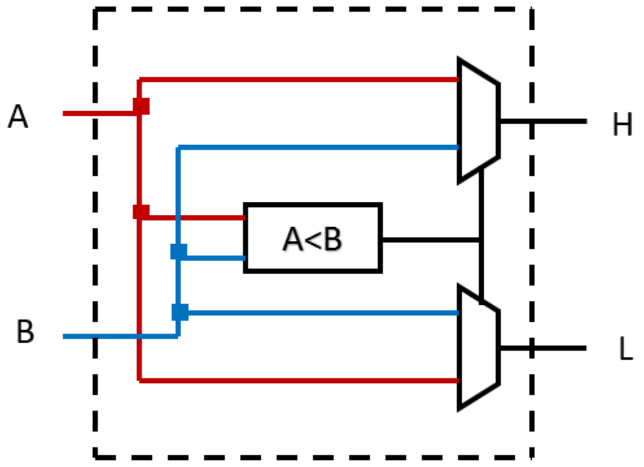


Figure 4:*Internal Diagram of Basic Exchange Node.*

As in figure, input image pixel values A and B are given to both the multiplexers and to the comparator. Based on the comparator output, each multiplexer selects either one input and gives as the output. Hence based on the comparison A<B, we get higher value as upper mux output and smaller value as lower mux output.

PROPOSED TECHNIQUE

Mainly this project is aimed on reducing the latency of the system to ensure the efficient performance of fault tolerant median filtering process. The proposed technique is an extension of the fault tolerant system which generates an output error signal whenever it detects an error. The overall working of the proposed system is same as that of fault tolerant method for median filtering [8]. The difference lies in the latency of the system. The proposed system has achieved low latency than the previous systems by reducing the number of clock cycles for sliding window formation.

The proposed technique has added a few registers for the efficient implementation of the fault tolerant technique. This technique has all the advantages of its precedent methods, and at the same time, it provides a low latency. To achieve this, two extra arrays and some registers are added.

At first, pixel values of the image are loaded to a 5×5 matrix as the input. Then a 3×3 sliding window applied to the image. Each time, the 3×3 window provides 9 pixel values to the sorting network. In the existing techniques, to form each 3×3 window consisting of 9 pixel values, 9 clock cycles are required. In the proposed technique, each set of 3 rows requires an initial 3 clock cycles and then 2 single clock cycles for forming sliding windows.

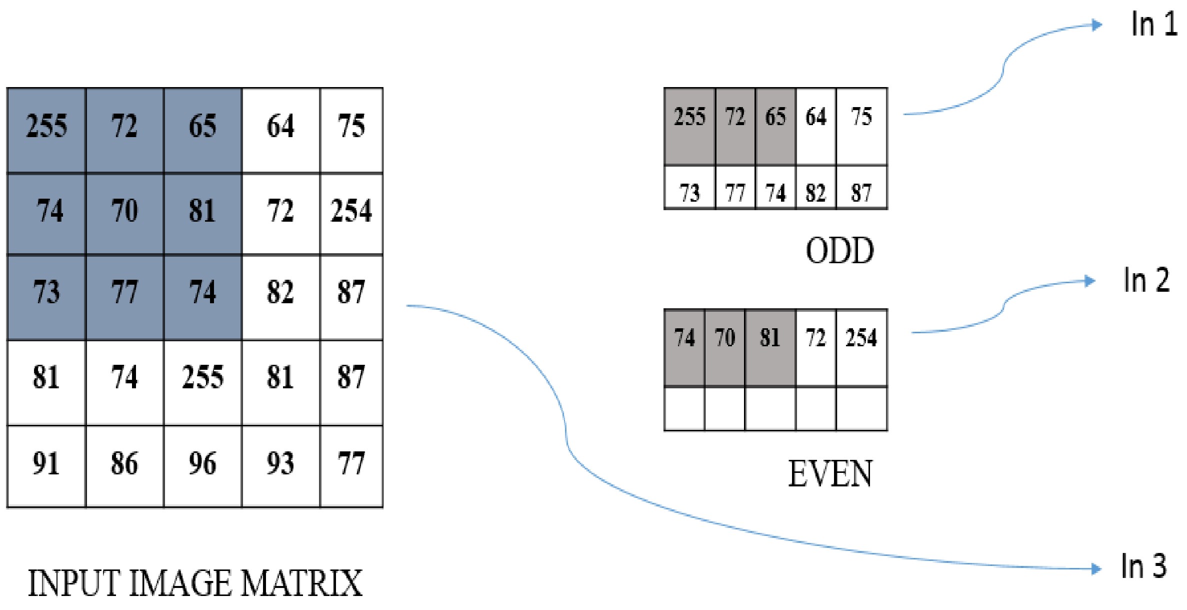


Figure 5:*Sliding Window Formation.*

Figure 10 shows a 5×5 matrix, the input image matrix which stores the pixel values of the image. The first 3×3 sliding window to be formed for sorting process, is shaded. The two arrays, odd and even, are used to store pixel values from the input matrix and from these arrays, the pixel values are taken to form sliding window. The first row of 'odd' formed by the elements of the 1st row of the input image matrix. Similarly, the elements of 2nd row of input image matrix are stored in the first row of 'even'.

In1 is a register which stores the first element in the 'odd' array, while register

In2 stores the first element of ‘even’ array. In3 directly stores values from 3rd row of the input matrix and at the same time, it stored in the 2nd row of the ‘odd’ too, for future use. In first clock cycle, values from In1, In2, and In3 are taken by the first columns of the 3 rows of the sliding window matrix [9, 10]. In next clock cycle, the entered values of sliding window matrix are shifted to the next columns of the same rows. The first columns are filled by the updated In1, In2, and In3 values as in the earlier manner. By the 3rd clock cycle, the 9 pixel values for the first sliding window have achieved. In the next clock cycle, the 2nd sliding window is formed and the 3rd is formed in next to that clock cycle. Figure 8 shows the formation of first set of sliding windows.

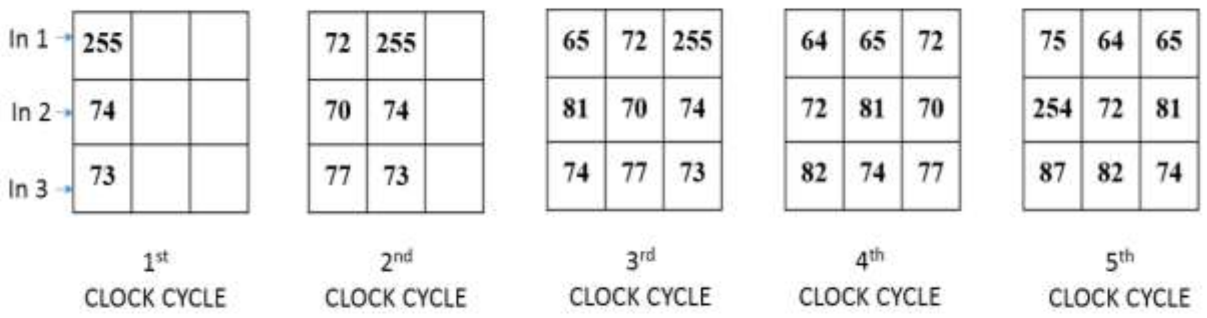


Figure 6:*Sliding Window Formation.*

Then it is the time to slide through the columns starting from the 2nd row of the input image matrix. For this, the values in ‘even’ is given to In1 and In2 is stored by the elements in second row of ‘odd’, which are filled by 3rd row elements of input image matrix during the initial clock cycle. Then In3 is stored directly from the 4th row of input matrix, and at the same time it stored in the 2nd row of ‘even’ too for future use [11–13]. Then the process is similar as described above. This process also requires an initial 3 clock cycles for the first sliding window matrix, and one single clock cycle for each subsequent sliding window matrices.

In similar way, each row requires an initial 3 clock cycles and then 2 single clock cycles for forming each sliding windows, since each row forms 3 sliding windows. A total of 9 sliding windows can be formed from a 5×5 matrix, the first sliding window being formed from first 3 elements of first 3 rows whereas the last sliding window formed by the 3rd, 4th and 5th elements of 3rd, 4th and 5th rows.

RESULTS AND DISCUSSION

An improved fault tolerant implementation of median filter is presented and performed. The typical fault tolerant system checks if the calculated median is within the range provided or not. If not, it set up an error signal to remove the noise and do necessary remedies [14, 15]. The proposed system improves this process by reducing the latency of the ordinary fault tolerant system. By reducing the latency, outputs are obtained fast as possible which in turn helps to do more processes in less amount of time.

In this project work, typical sorting networks used for median filter has been developed and simulated. The proposed technique has used the best sorting network, formed by Batcher’s Bitonic sorter with 19 basic nodes and extra 6 blocks for range formation. All the sorting networks are compared in terms of resource utilization, time, and power consumption. Error correction is done by CRC algorithm. Then the image is viewed by MATLAB software.

SIMULATION RESULTS

The proposed system and the fault tolerant system weresimulated using Xilinx ISE Design Suite. The simulation results are shown below.
For producing any sorting network, the first thing is to create a basic node. The basic node includes a comparator for comparing the input values and two multiplexers from where the output values are obtained.

8 bit Comparator

In each time, comparator checks whether ‘a<b’ is true or not where ‘a’&‘b’ are inputs. If it is true,the output ‘y’ becomes 1; otherwise ‘y’ is 0. Figure 7 shows the simulation result of 8 bit comparator.

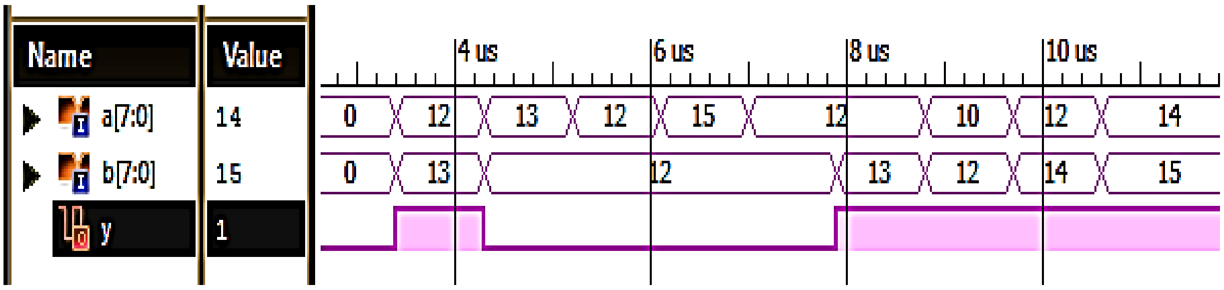


Figure 7:*8bit Comparator.*

8 bit Mux

Whenever the select input value‘s’ is low i.e. 0, the mux output ‘y’ selects the value of input ‘a’. When the select input‘s’ is high, i.e. 1, ‘y’ will be the value of ‘b’. Figure 8shows the simulation of 8 bit multiplexer [16].

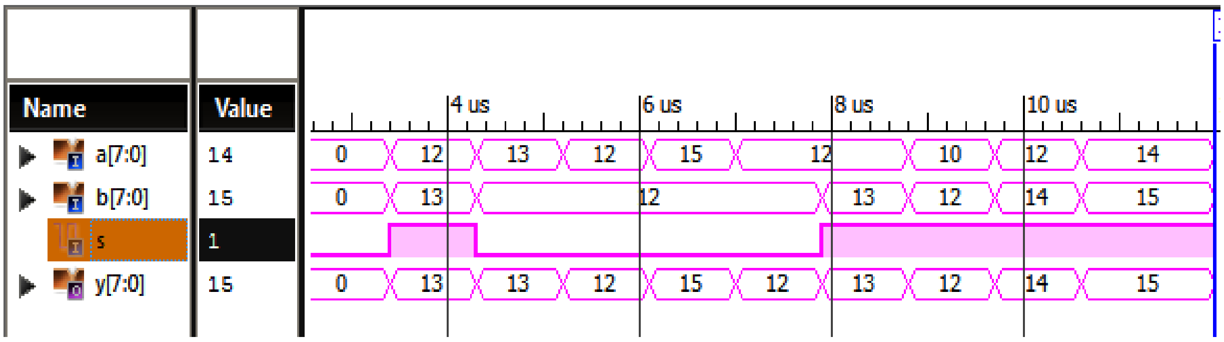


Figure 8:*8 bit Multiplexer.*

Basic Node

Basic node includes two multiplexers and a comparator. It compares the two inputs ‘a’&‘b’ at a time and produces the high and low outputs. Figure 14 shows the simulation result [17].

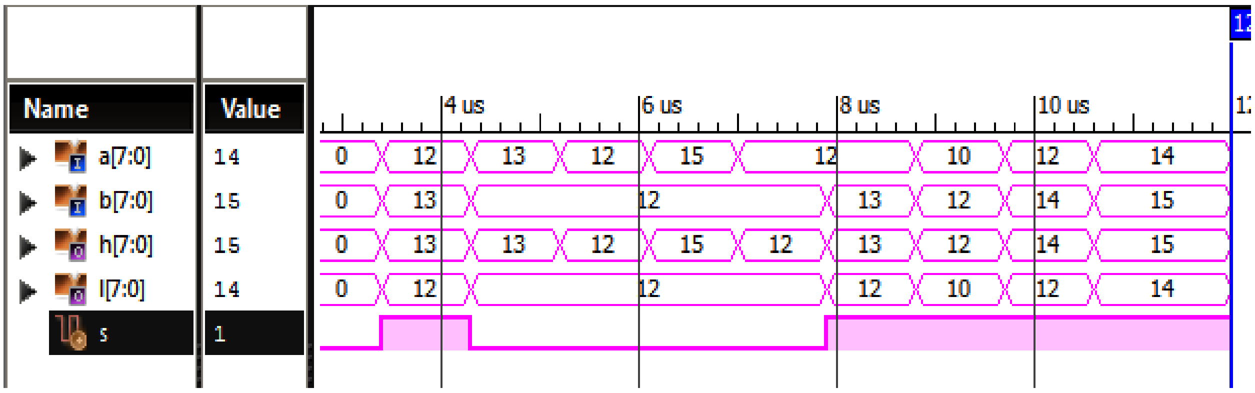
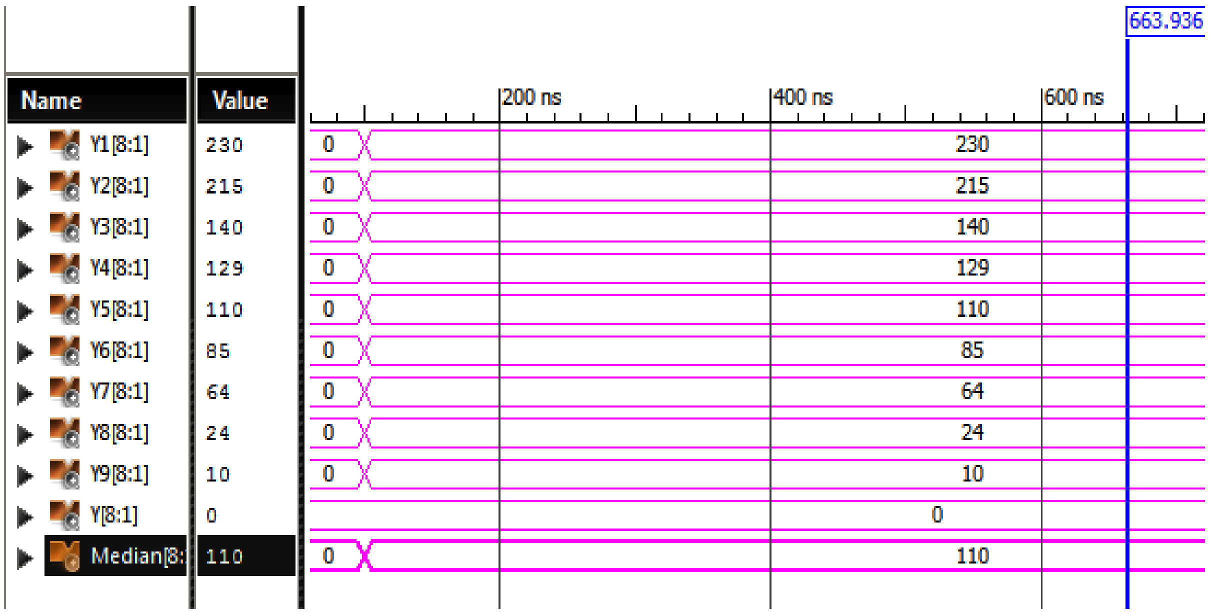


Figure 9:*Basic Exchange Node Simulation.*

Journal of VLSI Design and Signal Processing
e-ISSN: 2581-8449
Volume 5 Issue 1

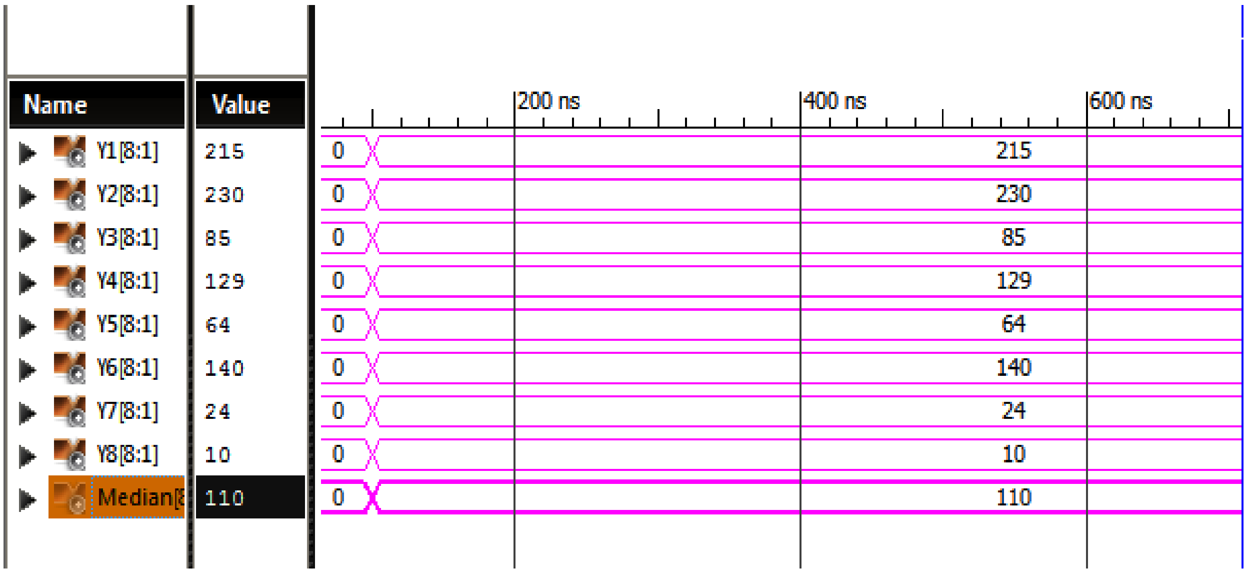
Classic Sorting Network
It includes 41 basic nodes. After sorting the 9 inputs, it provides the 9 outputs 'Y1-Y9' and obtained the 'Median' as in Figure 10 [18, 19].



Name	Value	Waveform
Y1[8:1]	230	0 → 230
Y2[8:1]	215	0 → 215
Y3[8:1]	140	0 → 140
Y4[8:1]	129	0 → 129
Y5[8:1]	110	0 → 110
Y6[8:1]	85	0 → 85
Y7[8:1]	64	0 → 64
Y8[8:1]	24	0 → 24
Y9[8:1]	10	0 → 10
Y[8:1]	0	0
Median[8:	110	0 → 110

Figure 10:*Classic Exchange Network Simulation.*

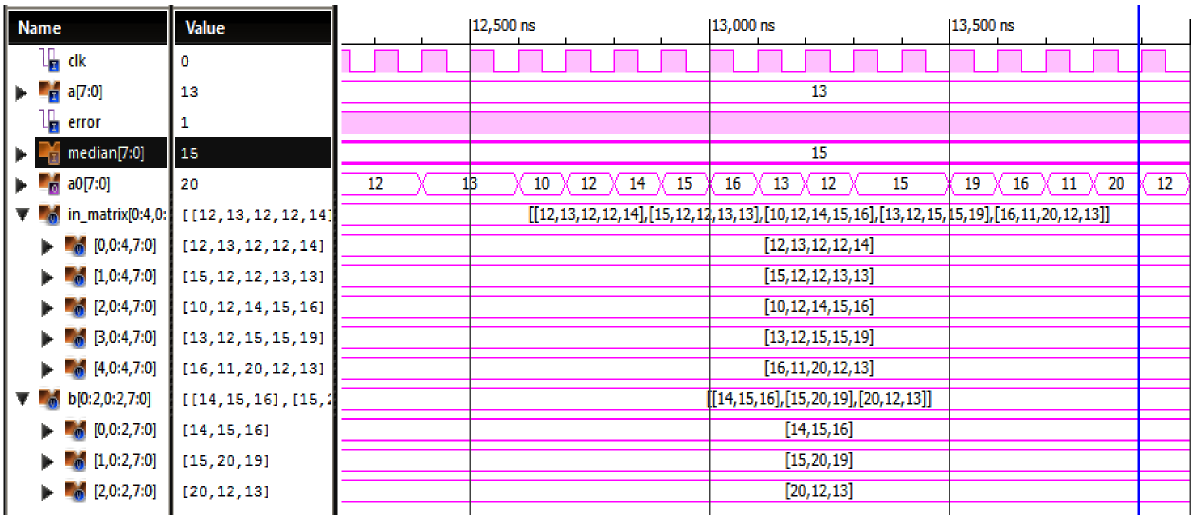
Minimum Exchange Network Scheme
It includes 19 basic nodes. This is the method which has used smaller number of exchange nodes. The 9 inputs are partially sorted and 'Median' is formed as in Figure 11.



Name	Value	Waveform
Y1[8:1]	215	0 → 215
Y2[8:1]	230	0 → 230
Y3[8:1]	85	0 → 85
Y4[8:1]	129	0 → 129
Y5[8:1]	64	0 → 64
Y6[8:1]	140	0 → 140
Y7[8:1]	24	0 → 24
Y8[8:1]	10	0 → 10
Median[8	110	0 → 110

Figure 11:*Minimum Exchange Network Scheme Simulation.*

Fault Tolerant Scheme
In addition to 19 exchange nodes as that of minimum exchange node scheme, the fault tolerant scheme needs 6 more additional exchange nodes for range calculation. It checks whether the median is between the range or not, so as to produce an error signal. Then the central pixel value of the sliding window matrix, is replaced by the median value. Figure 12 shows the simulation result of fault tolerant scheme.



```
Name                 Value
clk                  0
a[7:0]               13
error                1
median[7:0]          15
a0[7:0]              20
in_matrix[0:4,0:     [[12,13,12,12,14]
  [0,0:4,7:0]        [12,13,12,12,14]
  [1,0:4,7:0]        [15,12,12,13,13]
  [2,0:4,7:0]        [10,12,14,15,16]
  [3,0:4,7:0]        [13,12,15,15,19]
  [4,0:4,7:0]        [16,11,20,12,13]
b[0:2,0:2,7:0]       [[14,15,16],[15,2
  [0,0:2,7:0]        [14,15,16]
  [1,0:2,7:0]        [15,20,19]
  [2,0:2,7:0]        [20,12,13]
```

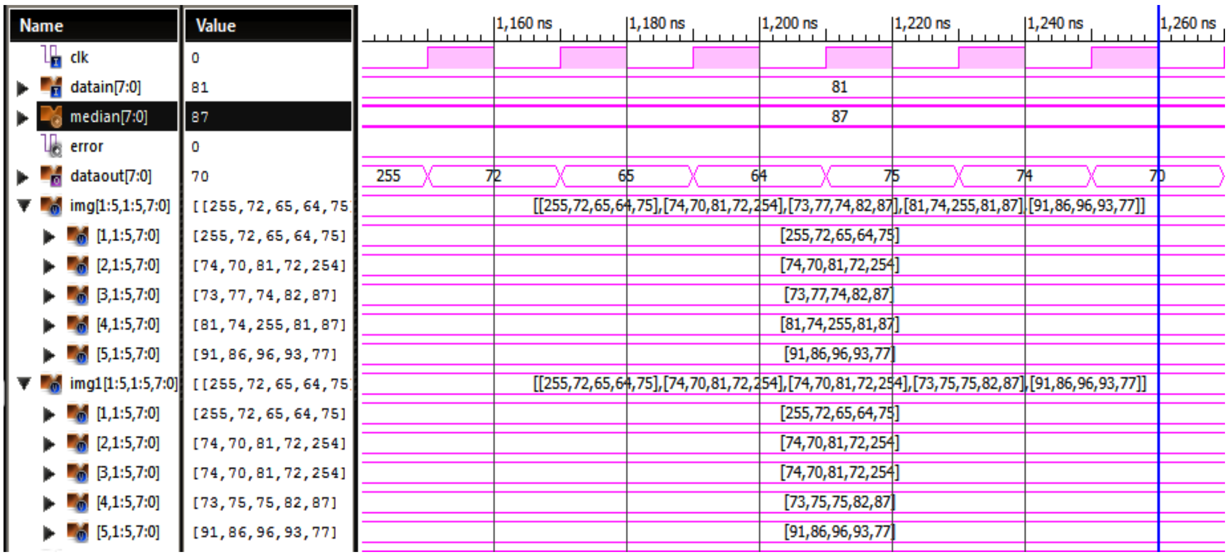
Figure 12:*Fault Tolerant Scheme Simulation.*

.

Improved Fault Tolerant Scheme

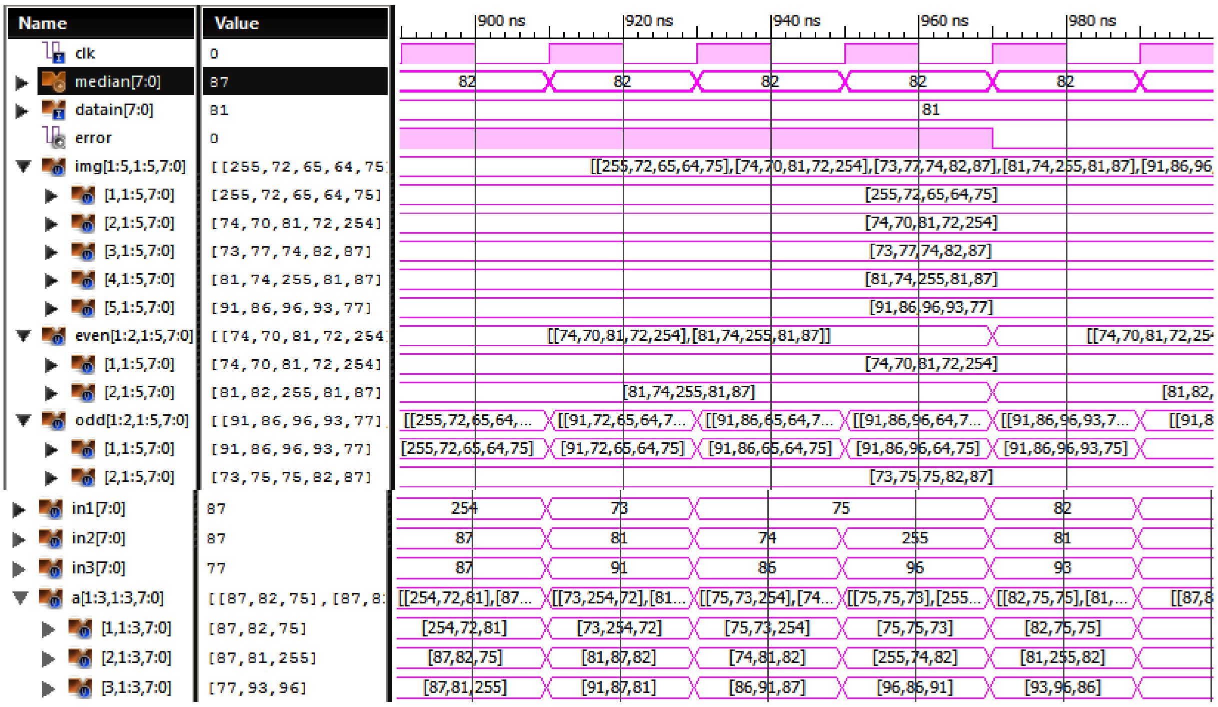
Our technique has the same working as fault tolerant scheme, which includes 25 basic exchange nodes for sorting algorithm. The change has made in sliding window section. For sliding window generation, it has used an additional 2 sets of arrays. The first matrix shows the input image ('img'). It is a 5×5 matrix to store 8 bit pixel values of input image. The pixel values are given to the system from a text file consisting of pixel values of the input image, by calling this file in the test bench. Error output signal shows high whenever the system detects an error. The median value ('median') for the final sliding window matrix values ('a') are shown in Figure 12. Also we can see that the median value replaces the original pixel value in the output image ('img1'). The values from the output image can be write into a text file, forming a text file consists of pixel values of the filtered image.

In Figure 13 (b), the formation of sliding window is shown. 'Even' and 'Odd' are the two extra arrays that are added to system to improve its performance. Also three registers in1, in2 and in3 are shown. 'a' is the sliding window for different clock cycles as shown in the Figure 13.



```
Name                 Value
clk                  0
datain[7:0]          81
median[7:0]          87
error                0
dataout[7:0]         70
img[1:5,1:5,7:0]     [[255,72,65,64,75
  [1,1:5,7:0]        [255,72,65,64,75]
  [2,1:5,7:0]        [74,70,81,72,254]
  [3,1:5,7:0]        [73,77,74,82,87]
  [4,1:5,7:0]        [81,74,255,81,87]
  [5,1:5,7:0]        [91,86,96,93,77]
img1[1:5,1:5,7:0]    [[255,72,65,64,75
  [1,1:5,7:0]        [255,72,65,64,75]
  [2,1:5,7:0]        [74,70,81,72,254]
  [3,1:5,7:0]        [74,70,81,72,254]
  [4,1:5,7:0]        [73,75,75,82,87]
  [5,1:5,7:0]        [91,86,96,93,77]
```

(a)

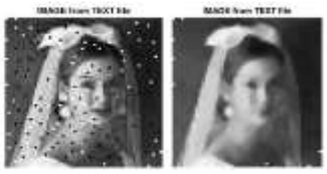


```
Name                Value
clk                 0
median[7:0]         87
datain[7:0]         81
error               0
img[1:5,1:5,7:0]    [[255,72,65,64,75
  [1,1:5,7:0]       [255,72,65,64,75]
  [2,1:5,7:0]       [74,70,81,72,254]
  [3,1:5,7:0]       [73,77,74,82,87]
  [4,1:5,7:0]       [81,74,255,81,87]
  [5,1:5,7:0]       [91,86,96,93,77]
even[1:2,1:5,7:0]   [[74,70,81,72,254
  [1,1:5,7:0]       [74,70,81,72,254]
  [2,1:5,7:0]       [81,82,255,81,87]
odd[1:2,1:5,7:0]    [[91,86,96,93,77]
  [1,1:5,7:0]       [91,86,96,93,77]
  [2,1:5,7:0]       [73,75,75,82,87]
in1[7:0]            87
in2[7:0]            87
in3[7:0]            77
a[1:3,1:3,7:0]      [[87,82,75],[87,8
  [1,1:3,7:0]       [87,82,75]
  [2,1:3,7:0]       [87,81,255]
  [3,1:3,7:0]       [77,93,96]
```

(b)

Figure 13:*Improved Fault Tolerant Method Simulation (a) Input and Output Image Matrices, (b)Sliding Window Formation.*

The simulation result shows the input and output images in terms of their pixel values. For easier understanding of the effect of median filtering, these images can be viewed in jpeg format. The text files from the Xilinx ISE simulation of improved fault tolerant implementation of median filter is converted in to image files using MATLAB software. From the obtained images, it is easy to understand the effectiveness of the median filter in noise removing. Figure 14 shows the MATLAB simulation of improved fault tolerant implementation of median filter.



(a) (b)

Figure 14:*MATLAB Simulation (a) Input Image, (b) Filtered Output Image, using MATLAB Simulation.*

SIMULATION ANALYSIS

Based on the simulation, some features are analysed. First of all, the sorting networks are simulated and compared their resource utilization and delay. Table 1 shows the comparative analysis of sorting methods.

Table 1:*Comparison of Sorting Methods.*

Method	No. of LUTs (Out of 46560)	Delay (ns)
Classic Exchange Network Scheme	703	28.325
Minimum Number Exchange Network Scheme	356	26.953

From Table 1, it is clear that minimum number of exchange network scheme uses lower number of resources and having low delay than classic network scheme. The minimum number of exchange network method is used in fault tolerant method also. The proposed system mainly aimed on increasing the speed of the system. Table 2 gives the comparison result of simulation of both fault tolerant and improved fault tolerant methods.

Table 2:*Comparison of Fault Tolerant and Improved Fault Tolerant Schemes.*

Method	No. of registers (out of 11440)	No. of LUTs (out of 5720)	No. of IO buffers (out of 102)	Delay (ns)
Fault tolerant method	387	1994	19	38.385
Improved fault tolerant method	505	2790	18	13.787

From Table 2, it is clear that improved fault tolerant scheme has reduced the delay efficiently. The additional number of registers and other resources are used to speed up the algorithm. As the aim of this project is to increase the speed of the fault tolerant median filtering process, the proposed technique has achieved sufficient speed for the algorithm by the addition of some resources.

CONCULSION

The fault tolerant implementation of median filter creates a range with which the calculated median value is checked, to detect the error present in a digital image. If the median value is out of the range, then the system generates an output error signal. The improved fault tolerant technique provides high speed to this algorithm by the addition of few resources. The proposed technique and other techniques simulated using Xilinx ISE design suite and the results shows that the proposed technique has less latency than the existing methods. This reduced latency improves the performance of the image processing by speeding up the filtering stage.

REFERENCES

1. Batcher K. E. Sorting networks and their applications. *AFIPS Spring Joint Computing Conference, San Francisco, CA*. 1968: pp. 307-314.
2. Yin L., Yang R., Gabbouj M. &Neuvo Y.Weighted median filters: a tutorial.*IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process*. March 1996; 43(3): pp. 157-192.
3. Bates G. L. &Nooshabadi S.FPGA implementation of a median filter.*Proceedings of IEEE Speech and Image Technologies for Computing and Telecommunications Conference (TENCON), Brisbane, Qld.*. 1997; 2: pp. 437-440.
4. Hopkinson G.R.Radiation effects in a CMOS active pixel sensor.*IEEE Trans. Nucl. Sci.* 2000; 47(6): pp. 2480-2484.
5. Eng H.-L.& Ma K.-K.Noise adaptive soft-switching median filter.*IEEE Trans. Image Process*.Feb 2001; 10(2): pp. 242-251.
6. Shim B.& Shanbhag N. R.Reduced precision redundancy for low-power digital filtering.*In Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and*

Computers, Pacific Grove, CA. 2001; 1: pp. 148-152.

7. Benkrid K., Crookes D.& Benkrid A.Design and implementation of a novel algorithm for general purpose median filtering on FPGAs.*IEEE International Symposium on Circuits and Systems (ISCAS)*. 2002; 4: pp. IV-425-IV-428.
8. Vega-Rodrıguez M. A., Sanchez-P´erez J. M. &Gomez-Pulido J. A.An FPGA-based implementation for median filter meeting the realtime requirements of automated visual inspection systems.*Proc. 10th Mediterranean Conf. Control and Automation, Lisbon, Portugal*. 2002.
9. Andre Roberto Guerra, J. M. Martins Ferreira & Manuel G. Gericota. Techniques to improve the reliability of fault-tolerant systems based on self-reconfigurable FPGAs. 2004.
10. Fahmy S. A., Cheung P. Y. K., &Luk W.Novel FPGA-based implementation of median and weighted median filters for image processing.*International Conference on Field Programmable Logic and Applications, Tampere, Finland*. 2005: pp. 142-147.
11. Sterpone L., Reorda M. S., Violante M., Kastensmidt F. L., &Carro L.Evaluating different solutions to design fault tolerant systems with SRAM-based FPGAs.*Journal of Electronic Testing*. 2007; 23(1): pp. 47-54.
12. Vasicek Z.,&Sekanina L.Novel Hardware Implementation of Adaptive Median Filters. *11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Bratislava, Slovakia*. 2008: pp. 1-6.
13. Reviriego P., Maestro J. A., L´opez-Calle I., &Agapito J. A. de. Soft Error Tolerant Infinite Impulse Response Filters Using Reduced Precision Replicas.*In Proc. of the Radiation Effects on Components and Systems (RADECS) conference, Sevilla, Spain*. 2011: pp. 493-496.
14. Marcela Simkova & Jan Kastil Verification of Fault-tolerant Methodologies for FPGA systems. 2012.
15. Malothu Nagu & N.Vijay Shanker. Image De-Noising By Using Median Filter and Weiner Filter.*In International Journal of Innovative Research in Computer and Communication Engineering*. Vol. 2, Issue 9, September 2014.
16. Rachna Mehta & Dr. Navneet Kumar Aggarwal. Comparative Analysis of Median Filter and Adaptive Filter for Impulse Noise – A Review.*In National Conference on Recent advances in Wireless Communication and Artificial Intelligence (RAWCAI-2014)*.
17. Ateng Eric. FPGA Implementation of Median Filter using an Improved Algorithm for Image Processing.*In IJIRST –International Journal for Innovative Research in Science & Technology*. May 2015; 1(12).
18. Garg Lovi, Garg Arushi&Kumar Amit. FPGA based design of median filter to reduce noise in imaging.*In International Journal of Electrical and Electronics Engineers*. Jan- June 2015; 7(1).
19. Aranda L.A., Reviriego P., &Maestro J. A.A Fault-Tolerant Implementation of the Median Filter.*presented at the Radiation Effects on Components and Systems (RADECS) Conference, Bremen, Germany*. 2016.

Cite this article as: *Linda Babu, & Gnana Sheela K. (2019). An Improved Fault Tolerant Technique of Median Filter. Journal of VLSI Design and Signal Processing, 5(1), 22–32. http://doi.org/10.5281/zenodo.2549248*