ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

An Efficient Decision Tree Based Architecture for Random Impulse Noise Removal in images

Sajan P. Philip[1], S. P. Prakash[2], Dr. S. Valarmathy[3]

P. G Sholar, Department of ECE, Bannari Amman Institute of Technology, Erode, India[1]

Assistant Professor, Department of ECE, Bannari Amman Institute of Technology, Erode, India[2]

Head of the Department, Department of ECE, Bannari Amman Institute of Technology, Erode, India[3]

ABSTRACT: Images are often degraded by impulse noise in the procedures of image acquirement and broadcast. In this paper, we propose an efficient VLSI Architecture of a Decision Tree Based Denoising algorithm which can be used for the effective removal of random-valued impulse noise in images. To accomplish the low cost, a low-complexity VLSI design is proposed. The main components of the proposed design are a Decision-Tree-Based Impulse Noise Detector to detect the noisy pixels in the corrupted image and an Edge Preserving Noise Filter to reconstruct the corrupted pixels. The Decision Tree Based Method includes a binary tree of three stages for the detection of the noisy pixel. The reconstructed pixels are written adaptively to the reconstructed image after the correction by the noise filter. The main feature of the proposed design is that it keeps the unaffected pixels untouched and thereby reducing the blurring effect in the reconstructed image. It requires only low computational complexity and two line memory buffers. The hardware cost of the proposed design is very less and therefore, the design is very appropriate to be applied to many real time scenarios.

Keywords: Noise Removal, Noise Filter, Decision Tree, VLSI Architecture, Median Filter, Impulse Noise

I. INTRODUCTION

In general, images are frequently corrupted by impulse noise in the events of image acquisition and broadcast. The efficiency of the Image processing techniques mainly depends on the noise in the images. Hence, a competent denoising technique becomes a very important issue in image processing [1], [2]. According to the distribution of noisy pixel values, impulse noise can be classified into two categories: Fixed-Valued impulse noise and Random-Valued impulse noise. The Fixed Values impulse noise is also known as "Salt and Pepper Noise" since the pixel value of a noisy pixel is either minimum or maximum value in grayscale images. The values of noisy pixels corrupted by random-valued impulse noise are uniformly distributed in the range of [0, 255] for gray-scale images. Removal of Random valued impulse noise is more complicated due to the random distribution of the noise pixels. In this paper, the main focus is on the detection and correction of the random-valued impulse noise from the corrupted image.

Recently many methods have been proposed for the removal of Impulse noise from the image. They are Standard Median Filter [3], modifications of Standard Median Filter [4], [5], Switching Method [6]-[8], Alpha Trimmed Mean Based Method (ATMBM)[9], Differential Rank Impulse Detector (DRID)[10], Directional Weighted Median (DWM)[11]. The main disadvantage of the standard median method is the blurring of the reconstructed image.

The complexity of denoising algorithms depends mainly on the local window size, memory buffer, and iteration times. The lower complexity techniques use a fixed-size local window, require a few line buffers and perform no iterations. Therefore, the computational complexity is low. However, the reconstructed image quality is not good enough. Hence achieving the higher quality reconstructed image with lower complexity method is a challenge. To achieve the goal of low cost, low power, less memory and easier computations, an efficient low complexity impulse noise removal method is essential.

Based on basic concepts available in the literature, here we present an efficient decision-tree-based denoising method (DTBDM) and its VLSI architecture for detecting random-valued impulse noise. The proposed design requires simple computations and two line memory buffers only, so its hardware cost is low. The main components of the Detector part are Isolation Module, Fringe Module and Similarity Module. The main component of the Corrector part is an Edge Preserving Median Filter.

The rest of this paper is organized as follows. The proposed DTBDM Impulse Detector is introduced briefly in Section II. Section III describes the Edge Preserving Filter.Section IV discuss the VLSI Design of the above architectures. Section V describes the Results andDiscussions and Section VI concludes the results.

II. PROPOSED DECISION TREE BASED IMPULSE DETECTOR

Here, the window size for the denoising process is 3x3. Assume the pixel tobe denoised is located at coordinate (i, j) and denoted as $p_{i,j}$,and its luminance value is $f_{i,j}$. The mask under consideration is shown in Fig. 1. We have divided the other pixels in the window as Top Half and Bottom Half. The overall design architecture of the proposed method is

ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

shown in Fig 2.Several methods are employed in the literature for the impulse detection. Based on the existing designs, we have designed three modules namely, Isolation Module (IM), Fringe Module (FM), and Similarity Module (SM). Three concatenating decisions of these modules make a decision tree.

A. Isolation Module

We use Isolation Module to make a decision whether the pixel value is in a smooth region. If the result is negative,we conclude that the pixel under consideration belongs to a noisy-free area. Otherwise if the result is positive, it indicates that the

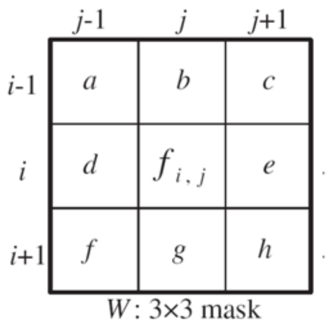


Fig.1. Window

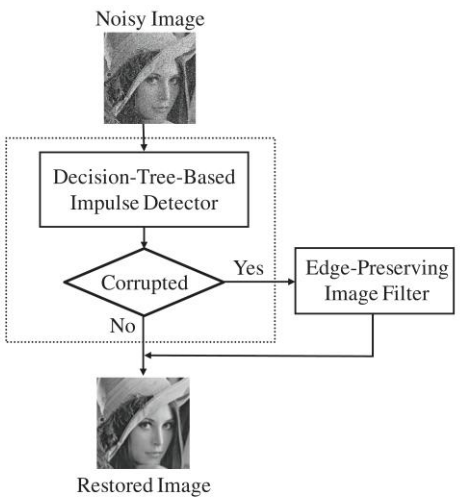

Noisy Image
Decision-Tree-Based Impulse Detector
Corrupted
Yes
No
Edge-Preserving Image Filter
Restored Image

Fig.2. The DTBDM Algorithm

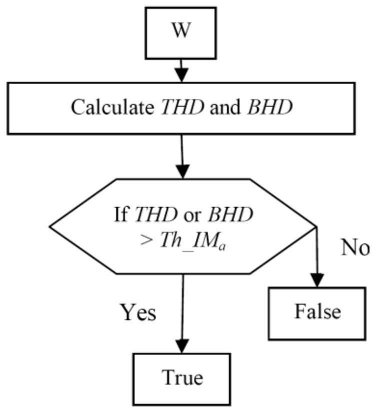


Fig. 3. Dataflow of Decision I in IM

pixel under consideration can be a noisy pixel or it is just situated on an edge of the object in the image. The Dataflow of the different components in Isolation Module is given in Fig. 3, Fig. 4 and Fig. 5. In order to avoid the complexity of the Design, the 3 x 3 window under consideration is divided into two different regions. Window Top Half and Window Bottom Half Where,

$W_{Top\ Half}=\{a,b,c,d\}$ (1)

$W_{Bottom\ Half}=\{e,f,g,h\}$ (2)

In the given dataflow diagrams, *THD, BHD, TH Max, TH Min, BH Max, BH Min* denotes*Top Half_difference, Bottom Half_difference, Top Half_Max, Top Half_Min, BottomHalf_Max, Bottom Half_Min*respectively.
Where,

$THD=TH\ Max - TH\ Min$ (3)

$BHD=BH\ Max - BH\ Min$ (4)

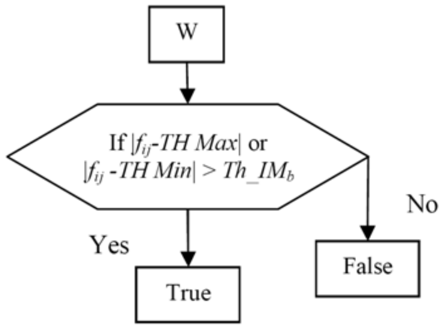


Fig.4. Dataflow of *IM_TH* in Decision I of IM

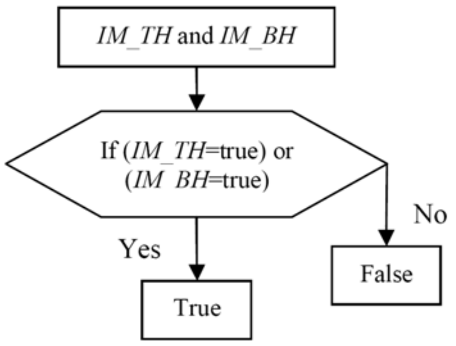


Fig. 5. The Dataflow of Decision II in IM

B. Fringe Module

In some cases, if the pixel is situated at the edge the Isolation Module may detect it as noisy pixel, In order to deal with this scenario; we define four directions, from E1 to E4, as shown inFig. 6. By calculating theabsolute difference between $f_{i,j}$and the other two pixel valuesalong the same direction, we can determine whetherthere is an edge or not. The Dataflow of the Fringe Module is given in Fig. 7 and Fig. 8.

ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

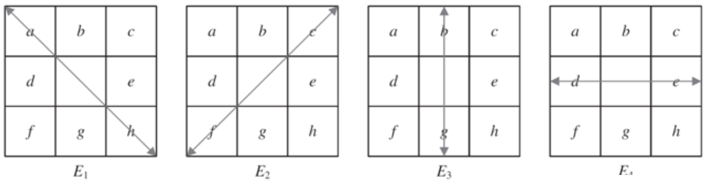


Fig.6 The Directions in the Fringe Module

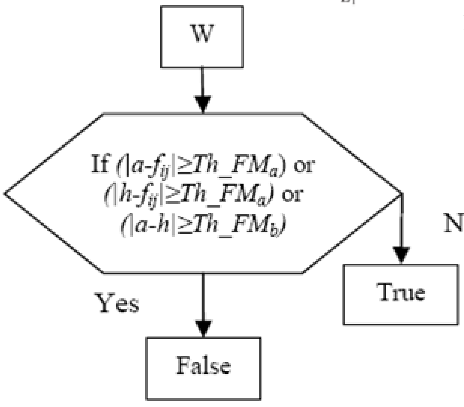
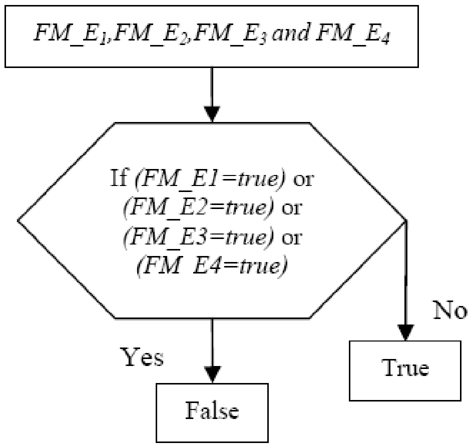


Fig.7. Dataflow of *FM_E1* in FM


FM_E1,FM_E2,FM_E3 and FM_E4
If (FM_E1=true) or (FM_E2=true) or (FM_E3=true) or (FM_E4=true)
No
True
Yes
False
Fig. 8. The Dataflow of Decision III in FM

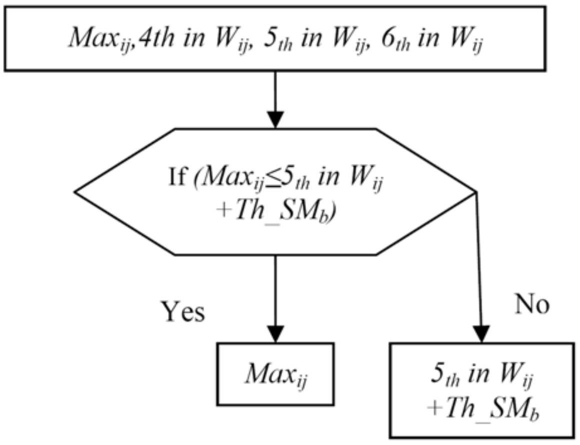
C. Similarity Module

The last step in impulse detection is the Similarity Module. The luminance valuesin mask W positioned in a noisy-free area might be close. The medianis always positioned in the center of the variation series,whereas the impulse is frequently located near one of its ends. Hence,if there are extreme big or small values, that shows the chanceof noisy signals. Based on this perception, we sort ninevalues in ascending order and obtain the 4th, 5th and 6th valueswhich are close to the median in mask W. In order to perform the operation, we need to define the following variables.

$$Max_{ij} = 6_{th} \text{ in } W_{ij} + Th\_SM_a \quad (5)$$

$$Min_{ij} = 4_{th} \text{ in } W_{ij} - Th\_SM_a \quad (6)$$

The Dataflow diagram of the Similarity module is given in Fig. 9 and Fig. 10. Calculation of Nmin is same as Nmax with some minor modifications. Threshold affects the performance of the denoising algorithms. The fixed values of the Thresholds make the algorithm simple and suitable for Hardware implementation. As per the experimental analysis and literature review, the values of Th_IM_a, Th_IM_b, Th_FM_a, Th_FM_b, Th_SM_a, Th_SM_b are 20, 25, 40, 80, 15 and 60 respectively.


Maxij,4th in Wij, 5th in Wij, 6th in Wij
If (Maxij≤5th in Wij +Th_SMb)
Yes
No
Maxij
5th in Wij +Th_SMb
Fig. 9. The Dataflow of N_{max} in Decision IV

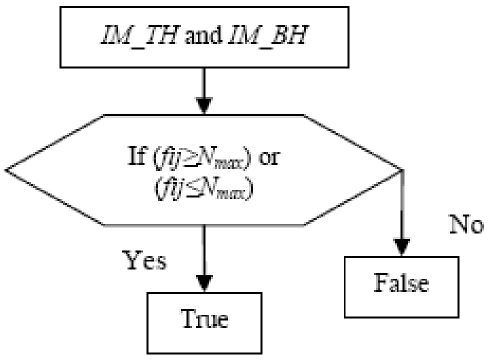


Fig. 10. The Dataflow of Decision IV

III. PROPOSED EDGE PRESERVING IMAGE FILTER

Edge preservation is one of the important consideration in Denoising algorithms. The similarity of the Reconstructed image with the Original image mainly depends on the Edges in the Image. Here, we consider eight directional differences, D1-D8, for the reconstruction of the Noisy pixel in the image as shown in the Fig. The main idea adopted here is to avoid the pixels, which are already known affected, for the reconstruction of the pixel *Pi,j*. This is to avoid the possible misdetection of the edges. This is accomplished by using $Max_{i,j}$ and $Min_{i,j}$, defined in similarity module (SM), to determine whether the values of *d, e, f, g* and *h* are likely corrupted respectively. If *d, e, f, g* and *h* are all suspected to be noisy pixels, and no edge can be processed, then the estimated value of $P_{i,j}$ is equal to the weighted

ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

averageof luminance values of three previously denoised pixels and calculated as *(a+b x 2+c)/4*. In other conditions, the edge filter calculates the directional differences of the chosen directions and locates the smallest one (D_{min}) among them. The overall Dataflow of the Edge Preserving Image Filter is given in the Fig. 11.

$$\left.\begin{array}{l} D_1=|d-h|+|a-e| \\ D_2=|a-g|+|b-h| \\ D_3=|b-g|\times 2 \\ D_4=|b-f|+|c-g| \\ D_5=|c-d|+|e-f| \\ D_6=|d-e|\times 2 \\ D_7=|a-h|\times 2 \\ D_8=|c-f|\times 2 \end{array}\right\} \quad (7)$$

$$\text{New } f_{ij}=\begin{cases} (a+d+e+h)/4, & \text{if } D_{min}=D_1 \\ (a+b+g+h)/4, & \text{if } D_{min}=D_2 \\ (b+g)/2, & \text{if } D_{min}=D_3 \\ (b+c+f+g)/4, & \text{if } D_{min}=D_4 \\ (c+d+e+f)/4, & \text{if } D_{min}=D_5 \\ (d+e)/2, & \text{if } D_{min}=D_6 \\ (a+h)/2, & \text{if } D_{min}=D_7 \\ (c+f)/2, & \text{if } D_{min}=D_8 \end{cases} \quad (8)$$

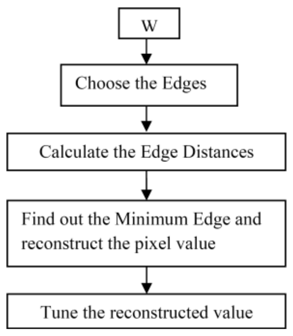

W
Choose the Edges
Calculate the Edge Distances
Find out the Minimum Edge and reconstruct the pixel value
Tune the reconstructed value

Fig.11. Edge preserving Filter Algorithm

IV. VLSI IMPLEMENTATIONOF THE PROPOSED METHOD

The reduction in cost and power is achieved by reducing the complexity in the computation and using simple computation elements in the overall design. The Detector part of the algorithm is implemented using simple Comparator, Subtractor, Adder and Mux. In the Filter part, the area and cost is reduced by using shifter instead of Multiplier and Divider.

A. Design of Isolation Module

Fig. 12 and Fig. 13 shows the Hardware architecture and Implementation of the Isolation Module for the Top Half of the Window. The overall architecture will be a combination of both the Top Half and Bottom Half. The comparator CMP_L is used to output the larger value from the two input values (8 bits) while the comparator CMP_S is used to output the smaller value from the two input values (8 bits). The SUB unit is used to subtract two 8 bit numbers. The |SUB| unit is used to determine the absolute difference between two 8 bit umbers. The GC is the greater comparator that will output ‘logic 1’ if the upper input value is greater than the lower one.*Decision I* is given as the selection line of the *Decision II* MUX.Finally, if the result of DecisionII is positive, $p_{i,j}$might be a noisy pixel or situate on anedge. The next module (FM) will be used to confirm the result.

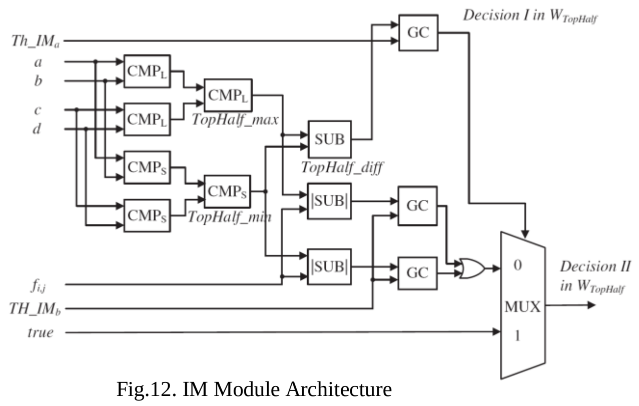


Fig.12. IM Module Architecture

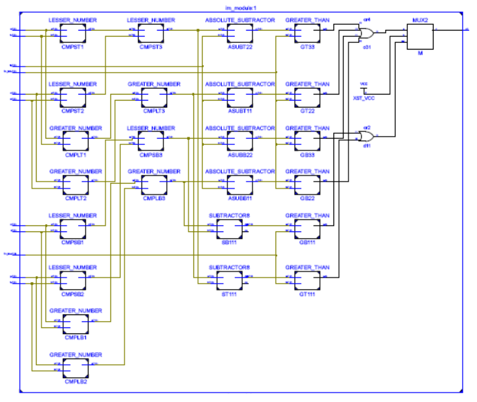


Fig.13. IM Module Hardware Implementation

B. Design of Fringe Module

Fig.14 shows the architecture of FM Module. The overall FM Module consists of four small structures called FM_1, FM_2, FM_3, and FM_4. Fig. 15 shows the detailed architecture of FM_1 structure. The |SUB| unit is used to calculate the absolute differences between the pixel values as given in the Dataflow of Fringe Module. The GC units will perform the comparisons and the NOR Gate will give the combined output. The combined output of each structure together selects the true or false value from the 2:1 MUX. The hardware Implementation of Fringe Module is given in Fig. 16.

ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

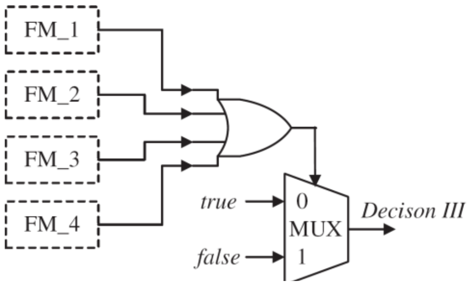


Fig.14. FM Module Architecture

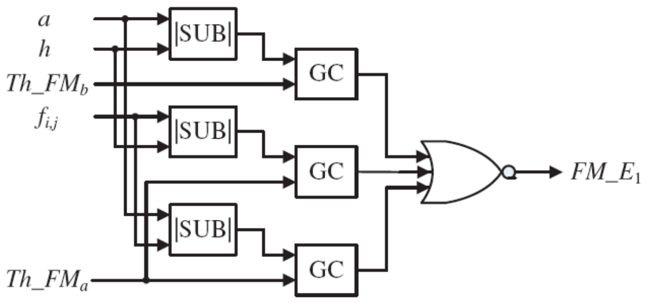


Fig.15. FM_1 Architecture

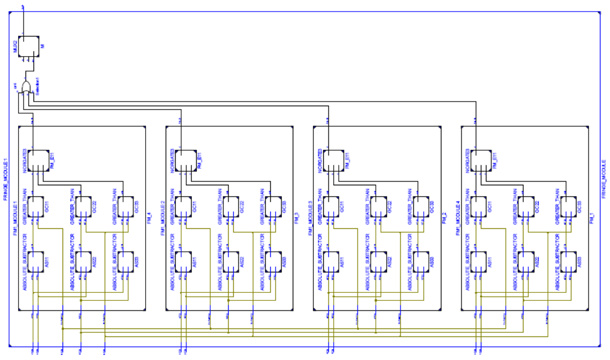


Fig.16. FM Module Hardware Implementation

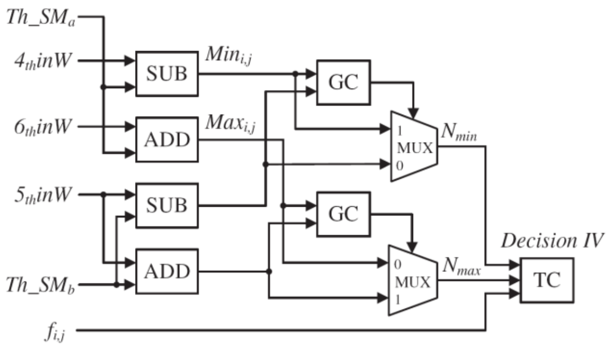


Fig.17. SM Architecture

C. *Design of Similarity Module*

Fig. 17 shows the architecture of the Similarity Module. Here the ADD and SUB units are used to calculate the $Max_{ij}$and Min_{ij}. An 8 Bit 2:1 MUX is used to select the N_{Max} and N_{Min}. The TC unit is a Triple Comparator which will output logic 1 if the lowest input is not in between the upper values. As shown in Fig.14, we need to find out the 4th, 5th and 6th value from the sorted pixel values in the window under consideration. We have proposed a fast and efficient architecture for the sorting process. The detailed implementation of the structures is shown in Fig. 18 and Fig. 19.

The idea behind the *M0* Module is as follows: If the value*a* is greater than*b*, *C01* is set to 1; or else, *C01* is set to 0. The eight *GC* unitsare used to determine the values from C01 to C08. After comparing,a combinational unit is used to combine the results of eachcomparator to obtain a number between 0 and 8. The number interprets the order of value in mask W under consideration. If *a*is the smallest valuein mask W, the output of the *M0* module is 0; if a is the biggestvalue in mask W, the output is 8. The architectures of othermodules *M1* to *M8* are approximately the same as *M0*, with only littledifference. By this approach in sorting, we can find outthe order of values efficiently by using simplecomparators andcombinational units. Comparing with conventional sorting algorithms,this method speeds up the sorting time, and reducesthe space which used to store the value to be exchanged. The Hardware Implementation of SM is given in Fig.20.

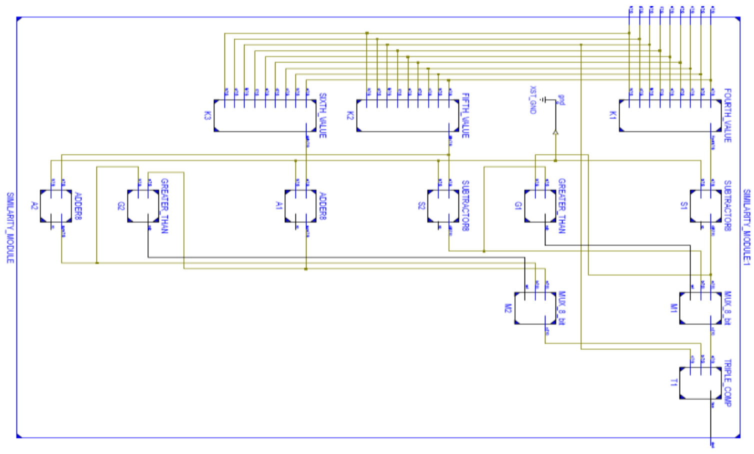


Fig.20. SM Hardware Implementation

ISSN (Print) : 2278 – 8875
ISSN (Online):2320 – 3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 4, April 2013

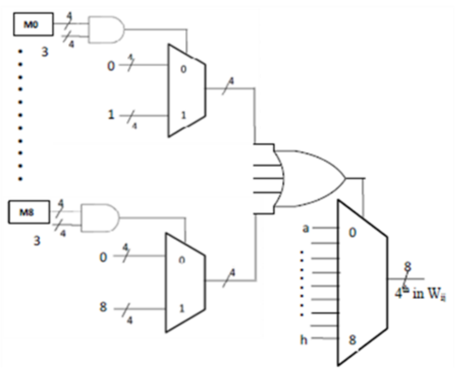


Fig.19. Sorting Architecture

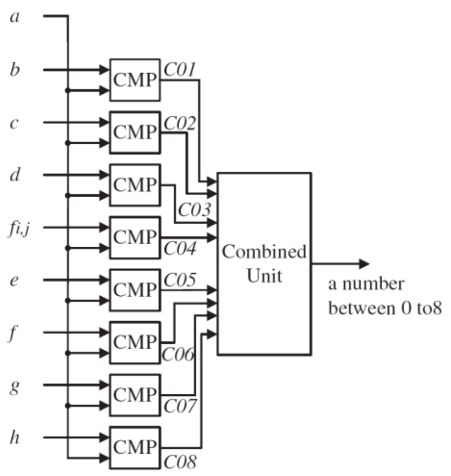


Fig.18. M0 Module Architecture

D. *Design of Edge Preserving Filter*

The Edge Preserving Filter consists of a Minimum Edge Generator and an Average Generator. The Edge Preserving Filter is used to find out the Edge with the smallest difference. Subtracter, Adder and Shifter are the main components of the Edge Preserving Filter. There is a MinTree used in the architecture to compare and find out the smallest Edge. Now the mean of the luminance values of the pixels which has the smallest directional difference can be obtained from the average generator. If all the pixels under consideration are suspected to be noisy pixels, the final MUX will outptu (a+bx2+c)/4. Otherwise, the MUX will output the mean of the pixel values whcih has minimum Edge difference. The architecture of MinED generator and Average Generator are given in Fig.21 and Fig 22.

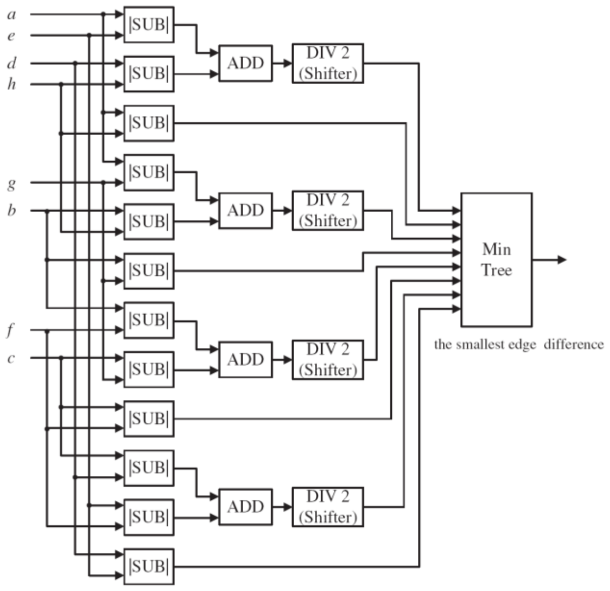


Fig.21. MinEdge Generator

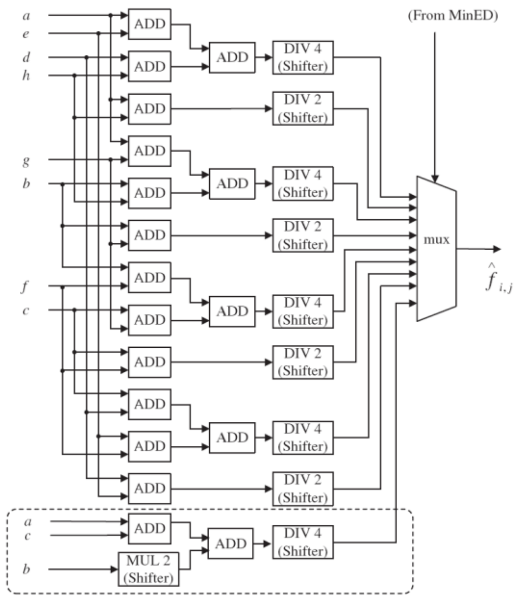
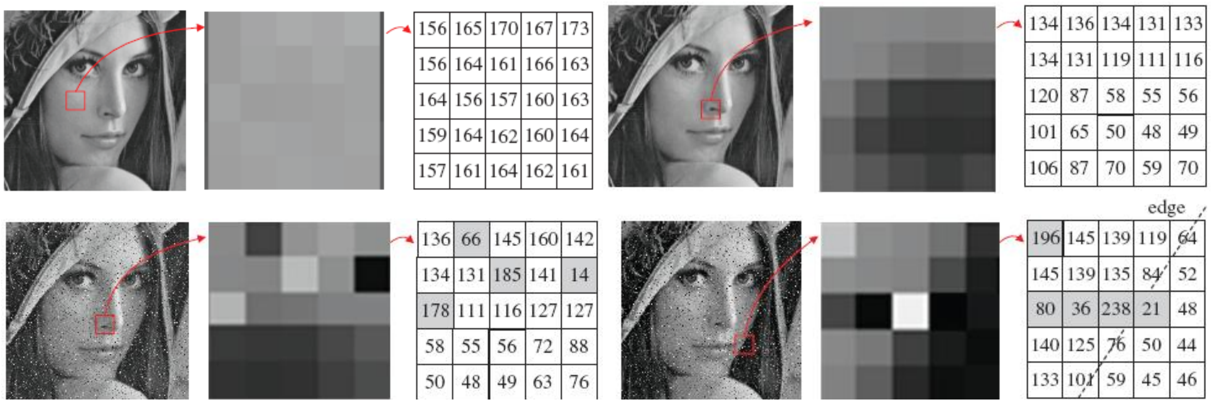


Fig.22. Average Generator

V. RESULTS AND DISCUSSIONS

The impulse detection process is a three stage process in the decision tree and it check for the smoothness of the area under consideration, position of the pixel in the edge of the image and the similarity of the pixel under consideration with the neighboring pixels. Here we have considered the Enlarged areas in the standard image “Lena”. In Fig. 23(a) a smooth area in the image is considered. In Fig. 23(c) an edge in the image is considered.Fig.23(d) is a noised added image.

In the same way the Edge preserving filter reconstruct the value by considering the neighbouring pixels. If all the neighbouring pixels are corrupted, then they will not be considered for the reconstruction of the pixel under consideration. In that case the reconstructed pixel value will be the average of the pixels *a*,*b* and *c*
.



(a)

156	165	170	167	173
156	164	161	166	163
164	156	157	160	163
159	164	162	160	164
157	161	164	162	161

(b)

134	136	134	131	133
134	131	119	111	116
120	87	58	55	56
101	65	50	48	49
106	87	70	59	70

(c)

136	66	145	160	142
134	131	185	141	14
178	111	116	127	127
58	55	56	72	88
50	48	49	63	76

(d)

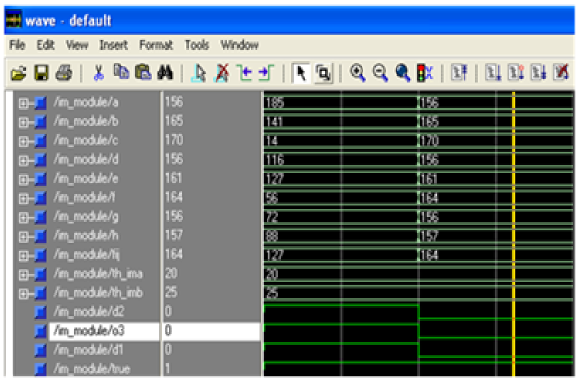
196	145	139	119	64
145	139	135	84	52
80	36	238	21	48
140	125	76	50	44
133	101	59	45	46

Fig.23. (a) Smooth Region (b) Non Smooth Region (c) Noisy pixels in the Edge (d) Edge in Noisy Image

The architecture is designed in VHDL and synthesized in the Xilinx 14.2 Simulation Tool and the simulation results are verified. Fig. 24 (a), (b), (c) shows the simulation results of the Isolation, Fringe and Similarity Modules respectively.

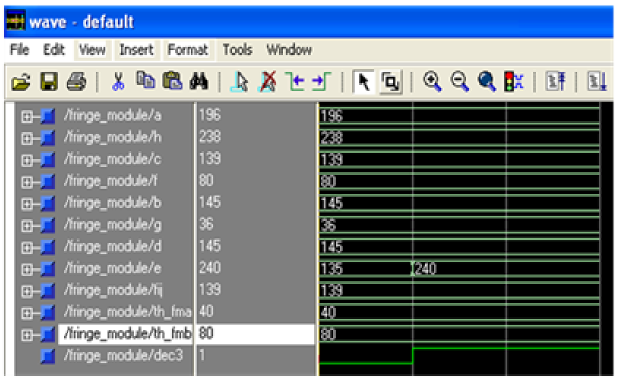
Table.1 shows the comparison Line Buffer and Iteration times of different algorithms with the proposed method. The proposed design requires simple computations and two line memory buffers only. Hence its hardware cost is low, which is an important concern in Hardware Design. For a 512×512 8-bit gray-scale test image, only two line buffer (512×2×8 bits) is required in our design. Most of the promising methods need to buffer a full image (512×512×8 bits). In our design, 99.6% of storage is reduced. Moreover, only simple arithmetic operations, such as addition and subtraction, are used in our Method.

Table.2 shows the summary of the design in terms of the LUTs, Slices, and Registers etc. We can see that the design is compact and efficient due to the simple components like Adder, Subtractor, and Comparator etc are the building blocks of the Architecture.



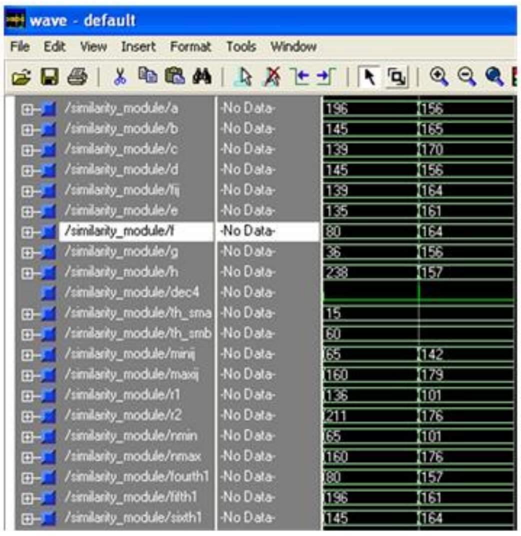
```
/im_module/a       156   185  156
/im_module/b       165   141  165
/im_module/c       170   14   170
/im_module/d       156   116  156
/im_module/e       161   127  161
/im_module/f       164   56   164
/im_module/g       156   72   156
/im_module/h       157   88   157
/im_module/fij     164   127  164
/im_module/th_ima  20    20
/im_module/th_imb  25    25
/im_module/d2      0
/im_module/o3      0
/im_module/d1      0
/im_module/true    1
```

(a)



```
/fringe_module/a       196   196
/fringe_module/h       238   238
/fringe_module/c       139   139
/fringe_module/f       80    80
/fringe_module/b       145   145
/fringe_module/g       36    36
/fringe_module/d       145   145
/fringe_module/e       240   135   240
/fringe_module/fij     139   139
/fringe_module/th_fma  40    40
/fringe_module/th_fmb  80    80
/fringe_module/dec3    1
```

(b)



```
/similarity_module/a        -No Data-   196  156
/similarity_module/b        -No Data-   145  165
/similarity_module/c        -No Data-   139  170
/similarity_module/d        -No Data-   145  156
/similarity_module/fij      -No Data-   139  164
/similarity_module/e        -No Data-   135  161
/similarity_module/f        -No Data-   80   164
/similarity_module/g        -No Data-   36   156
/similarity_module/h        -No Data-   238  157
/similarity_module/dec4     -No Data-
/similarity_module/th_sma   -No Data-   15
/similarity_module/th_smb   -No Data-   60
/similarity_module/minij    -No Data-   65   142
/similarity_module/maxij    -No Data-   160  179
/similarity_module/r1       -No Data-   136  101
/similarity_module/r2       -No Data-   211  176
/similarity_module/nmin     -No Data-   65   101
/similarity_module/nmax     -No Data-   160  176
/similarity_module/fourth1  -No Data-   80   157
/similarity_module/fifth1   -No Data-   196  161
/similarity_module/sixth1   -No Data-   145  164
```

Fig.24. Simulations results of (a) IM (b) FM (c) SM

Table 1: Comparison of Different Algorithms

Method	Mask Size	Line Buffer	Iteration Time
ATMBM[14]	3 x 3	Whole Image	4
DRID[15]	5 x 5	Whole Image	4
DWM[17]	7 x 7	Whole Image	3
Proposed	3 x 3	2	0
LCNR[36]	3 x 3	2	0
AMF[34]	3 x 3	4	0

Table 2: Hardware Implementation

Parameter	Metric
Slice LUTs	287
Registers	214
Delay	20.540ns
Frequency	129.32Mhz

(c)

VI. CONCLUSIONS

A low-cost VLSI architecture for competent removal of random- valued impulse noise is proposed in this paper. The method uses the decision-tree-based detector to detect the noisy pixel in the image, and employs an effective design to locate the edge. With adaptive skill, the quality of the reconstructed images is notable improved. Our extensive experimental results demonstrate that the performance of our proposed technique is better than the previous lower-omplexity methods and is comparable to the higher-complexity methods in terms It requires only low computational complexity and two line memory buffers. Therefore, it is very suitable to be applied to many real-time applications.

REFERNCES

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing. Pearson Education, Upper Saddle River, New Jersey, 2007.
[2] W. K. Pratt, Digital Image Processing. New York: Wiley-Interscience, 1991.
[3] T. Nodes and N. Gallagher, "Median filters: some modifications and their properties," IEEE Trans. ASSP, vol. ASSP-30, no. 5, pp. 739-746, Oct. 1982.
[4] S.-J. Ko and Y.-H. Lee, "Center weighted median filters and their applications to image enhancement," IEEE Trans. Circuits Syst., vol. 38, no. 9, pp. 984- 993,Sept. 1991.
[5] T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," Pattern Recognit. Lett., vol. 15, pp. 341–347, Apr. 1994.
[6] E. Abreu, M. Lightstone, S. K. Mitra, and K. Arakawa, "A new efficient approach for the removal of impulse noise from highly corrupted images," IEEE
Trans. Image Process., vol. 5, no. 6, pp. 1012–1025, Jun. 1996.
[7] T. Chen and H. R. Wu, "Adaptive impulse detection using center-weighted median filters," IEEE Signal Process.Lett., vol. 8, no. 1, pp. 1–3, Jan. 2001.
[8] T. Chen and H. R.Wu, "Space variant median filters for the restoration of impulse noise corrupted images," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 48, no. 8, pp. 784–789, Aug. 2001.
[9] W. Luo, "An efficient detail-preserving approach for removing impulse noise in images," IEEE Signal Process.Lett., vol. 13, no. 7, pp. 413-416, July 2006.
[10]I. Aizenberg, and C. Butakoff, "Effective impulse detector based on rank-order criteria," IEEE Signal Process. Lett., vol. 11, no. 3, pp. 363-366, Mar. 2004.
[11]Y. Dong and S. Xu, "A new directional weighted median filter for removal of random-valued impulse noise," IEEE Signal Process.Lett., vol. 14, no. 3, pp.193–196, 2007.
[12]N. I. Petrovic and V. Crnojevic, "Universal impulse noise filter based on genetic programming," IEEE Trans. Image Process., vol. 17, no. 7, pp. 1109- 1120, July 2008.
[13]C. Y Lien, C.H Huang, P.Y Chen, Y. F Lin An Efficient Denoising Architecture forRemoval of Impulse Noise in Images, IEEE Transactions on Computers, March 2012

Sajan P Philip has received B. Tech degree in Electronics and Communication from College of Engineering, Chengannur, Kerala, affiliated to Cochin University of Science and T echnology (2010). Currently his pursuing his Masters in Applied Electronics at Bannari Amman Institute of Technology, Tamil Nadu affiliated to Anna University Chennai. His research interest includes Computer Architecture, Computer Networks and VLSI Design.

S P Prakash has received BE degree in Electronics and Communication from Kongu Engineering College, Coimbatore, Tamil Nadu affiliated to Bharathiar University (2004). He received his ME degree in Applied Electronics from Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, affiliated to Anna University Chennai(2007). Currently he is pursuing P.hd in Low Power VLSI Design under Anna University Chennai. His research interest includes Network analysis and Synthesis and Low power VLSI Design.

Dr. S Valarmathy has received BE degree in Electronics and Communication from Bharathiar University, Coimbatore, Tamil Nadu (1989). She received her ME Degree in Applied Electronics from Bharathiar University, Coimbatore, Tamil Nadu(2000). She received her P.hd degree for her work in Biometrics. Currently she is working as the Head of the Department of Electronics and Communication, Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu. Her research interests includes Soft computing, Data mining, Image processing and Speech processing.