An FPGA Based Image Denoising Architecture with Histogram Equalization for Enhancement

Jayalakshmi S Nair*
ECE Dept. & M.G. University

Bibin Binu Simon
ECE Dept. &M.G.University

Abstract*— This work provides an insight into the design of an efficient denoising architecture for removal of impulse noise in images. Images are often corrupted by impulse noise in the procedures of image acquisition and transmission. In this work an efficient denoising scheme and its VLSI architecture for the removal of random-valued impulse noise is proposed. To achieve the goal of low cost, a low complexity VLSI architecture is proposed. Here a decision-tree-based impulse noise detector to detect the noisy pixels, and an edge-preserving filter to reconstruct the intensity values of noisy pixels are used. Furthermore, histogram equalization is used to enhance the effects of removal of impulse noise. Our extensive experimental results demonstrate that the proposed technique can obtain better performances in terms of both quantitative evaluation and visual quality than the previous lower complexity methods. Moreover, the performance can be comparable to the higher complexity methods. Its hardware cost is low and suitable to be applied to many real-time applications. These are implemented using Verilog Hardware Description Language. The Verilog code is synthesized on Spartan 3 FPGA using Xilinx ISE 14.7 software tool.

***Keywords*— FPGA, Image denoising, impulse noise, impulse detector, Xilinx**

I. INTRODUCTION

Image processing is widely used in many fields, such as medical imaging, scanning techniques, printing skills, license plate recognition, face recognition, and so on. In general, images are often corrupted by impulse noise in the procedures of image acquisition and transmission. The noise may seriously affect the performance of image processing techniques. Hence, an efficient denoising technique becomes a very important issue in image processing. 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 former is also known as salt-and-pepper noise because the pixel value of a noisy pixel is either minimum or maximum value in gray-scale 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. There have been many methods for removing salt and-pepper noise, and some of them perform very well. The random-valued impulse noise is more difficult to handle due to the random distribution of noisy pixel values. Here we only focus on removing the random-valued impulse noise from the corrupted image in this paper.

Recently, many image denoising methods have been proposed to carry out impulse noise suppression. Some of them employ the standard median filter or its modifications. However, these approaches might blur the image since both noisy and noise-free pixels are modified. To avoid the damage on noise-free pixels, an efficient switching strategy has been proposed in the literature. In general, the switching median filter consists of two steps: 1) impulse detection and 2) noise filtering. It locates the noisy pixels with an impulse detector, and then filters them rather than the whole pixels of an image to avoid causing the damage on noise-free pixels. In addition to median filter, there are other methods used to carry out impulse noise.

Generally, the denoising methods can be classified into two categories: lower complexity techniques and higher complexity techniques. 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. The higher complexity techniques yield visually pleasing images by using high computational complexity arithmetic operations, enlarging local window size adaptively or doing iterations. The higher complexity approaches require long computational time as well as full frame buffer.

Digital image are often corrupted by different types of noise, namely, additive white Gaussian noise, impulse noise and mixed (Gaussian and impulse) noise . Noises are added in the image during acquisition by camera sensors and transmission in the channel. Hence, image de-noising is one of the most common and important image processing operations in image and video processing applications. Today, in many practical real-time applications, the denoising process is included in end-user equipment, so there appears an increasing need of a good lower-complexity denoising technique, which is simple and suitable for low-cost VLSI implementation. Low cost is a very important consideration in purchasing consumer electronic products. To achieve the goal of low cost, less memory and easier computations are indispensable. In this paper, we focus only on the lower complexity denoising techniques because of its simplicity and easy implementation with the VLSI circuit.

Although the field of digital image processing is built on a foundation of mathematical and probabilistic formulation human eyes and analysis play a central role in the choice of one technique versus another, and this choice is often made based on subjective and visual judgments. The impulse noise called salt and pepper causes black and white points appears in digital gray scale images, which chaotically scattered along image area

In order to reduce noise in digital images, a great number of algorithms have been presented. One of the most common ways to remove noise is convolving the image with a mask that represents a low-pass filter (e.g. Gaussian filter), performing a smoothing operation through a weighted average of neighbours, where the weights decrease with distance from the central pixel on filter window. Often, the use of smoothing filters cause significant blurring of edges. Recently, Buades et al. proposed interesting work in which they uses the Non-Local Means algorithm, supported on the idea that images contain repeated structures, and averaging this structures, noise can be reduced. The decision tree is a simple but powerful form of multiple variable analysis . It can break down a complex decision-making process into a collection of simpler decisions, thus provide a solution which is often easier to interpret. There have been several methods using decision tree to deal with salt-and-pepper noise and some of them perform well. Based on above basic concepts, we present a novel adaptive decision-tree-based denoising method (DTBDM) and its VLSI architecture for removing random-valued impulse noise. To enhance the effects of removal of impulse noise, the results of reconstructed pixels are adaptively written back as a part of input data.

The proposed design requires simple computations and two line memory buffers only, so its hardware cost is low. For a 512 x 512 x 8-bit gray scale test image, only two line buffers (512 x 2 x 8 bits) is needed in our design. Most state-of-the-art methods need to buffer a full image (512 x 512 x 8) bits. In the proposed design, 99.6 percent of storage is reduced. Furthermore, only simple arithmetic operations, such as addition and subtraction, are used in DTBDM. Especially, it can remove the noise from corrupted images efficiently and requires no previous training. These experimental results demonstrate that the proposed technique can obtain better performances in terms of both quantitative evaluation and visual quality than other lower complexity denoising methods.

Moreover, the performance can be comparable to the higher complexity methods. The seven-stage VLSI architecture for the proposed design will be implemented and synthesized by using Verilog HDL and Xilinx ISE Tool using Spartan 3 FPGA device respectively. The content of this report is organized as follows. First a literature survey on existing methods is described briefly. The proposed DTBDM is introduced briefly in next section and describes the proposed VLSI architecture in detail. Next section illustrates the VLSI implementation and comparisons. The conclusion is provided in next section with advantages and disadvantages along with scope of the proposed work.

II. DECISION TREE BASED DENOISING METHOD

Many denoising schemes are "*decision-based*" median filters. This means that the noise candidates are first detected by some rules and are replaced by the median output or its variants. These schemes are good because the uncorrupted pixels will not be modified. However, the replacement methods in these denoising schemes cannot preserve the features of the images, in particular the edges are smeared. a decision based algorithm is proposed. In this, image is denoised by using a 3×3 window. If the processing pixel value is 0 or 255 it is processed or else it is left unchanged. The selected 3× 3 window elements are arranged in either increasing or decreasing order. Then the pixel values 0's and 255's in the image (i.e., the pixel values responsible for the salt and pepper noise) are removed from the image. Then the median value of the remaining pixels is taken. This median value is used to replace the noisy pixel.

A .Decision-Tree-Based Impulse Detector

The proposed method here is decision tree based denoising method (DTBDM). The noise considered in this paper is random-valued impulse noise with uniform distribution . Here, we adopt a 3 x 3 mask for image denoising. Assume the pixel to be denoised is located at coordinate *(i, j)* and denoted as $p_{i,j}$, and its luminance value is named as $f_{i,j}$, as shown in Figure.1. According to the input sequence of image denoising process, we can divide other eight pixel values into two sets: $W_{TopHalf}$ and $W_{BottomHalf}$. They are given as

$$W_{TopHalf} = \{a, b, c, d\} \quad (1)$$
$$W_{BottomHalf} = \{ e, f, g, h \quad (2)$$

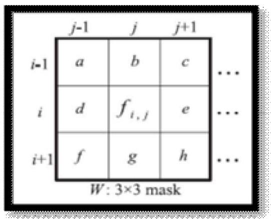


Fig. 1 A 3X 3mask centered on $p_{i,j}$

DTBDM consists of two components: decision-tree-based impulse detector and edge-preserving image filter. The detector determines whether $p_{i,j}$ is a noisy pixel by using the decision tree and the correlation between pixel $p_{i,j}$ and its neighbouring pixels. If the result is positive, edge preserving image filter based on direction-oriented filter generates the reconstructed value. Otherwise, the value will be kept unchanged. The design concept of the DTBDM is displayed in Figure.2.

In order to determine whether $p_{i,j}$ is a noisy pixel, the correlations between and its $p_{i,j}$ neighbouring pixels are considered . Surveying these methods, we can simply classify them into several ways—observing the degree of isolation at current pixel determining whether the current pixel is on a fringe or comparing the similarity between current pixel and its neighbouring pixels. Therefore,in the decision-tree-based impulse detector, we design three modules—isolation module (IM), fringe module (FM), and similarity module (SM). Three concatenating decisions of these modules build a decision tree. The decision tree is a binary tree and can determine the status of $p_{i,j}$ by using the different equations in different modules.

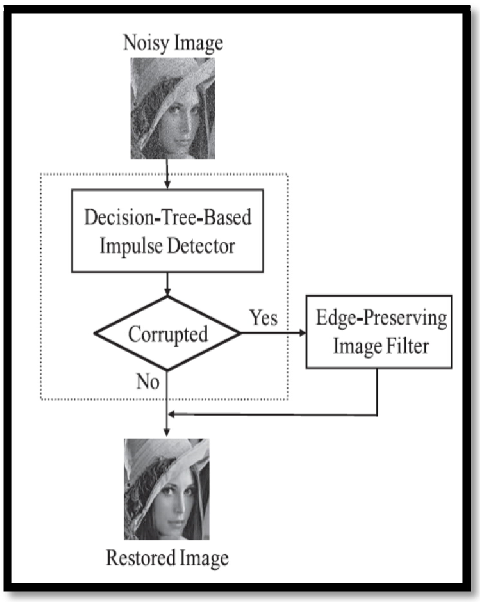

Noisy Image
Decision-Tree-Based Impulse Detector
Corrupted
Yes
No
Edge-Preserving Image Filter
Restored Image

Fig. 2 The Dataflow of DTBDM

First, we use isolation module to decide whether the pixel value is in a smooth region. If the result is negative, we conclude that the current pixel belongs to noisy free. Otherwise, if the result is positive, it means that the current pixel might be a noisy pixel or just situated on an edge. The fringe module is used to confirm the result. If the current pixel is situated on an edge, the result of fringe module will be negative (noisy free); otherwise, the result will be positive. If isolation module and fringe module cannot determine whether current pixel belongs to noisy free, the similarity module is used to decide the result. It compares the similarity between current pixel and its neighboring pixels. If the result is positive, $p_{i,j}$ is a noisy pixel; otherwise, it is noise free. The following sections describe the three modules in detail.

1) Isolation Module: The pixel values in a smooth region should be close or locally slightly varying, as shown . The differences between its neighbouring pixel values are small. If there are noisy values, edges, or blocks in this region, the distribution of the values is different, as shown in Figure. 3. Therefore, we determine whether current pixel is an isolation point by observing the smoothness of its surrounding pixels. Figure. 4 shows an example of noisy image. The pixels with shadow suffering from noise have low similarity with the neighbouring pixels and the so-called isolation point. The difference between it and its neighbouring pixel value is large. According to the above concepts, we first detect the maximum and minimum luminance values in $W_{TopHalf}$, named as *TopHalf_max*, *TopHalf_min*, and calculate the difference between them, named as *TopHalf_diff*. For $W_{\mathrm{BottomHalf}}$, we apply the same idea to obtain *BottomHalf_diff*. The two difference values are compared with a threshold *Th_IMa* to decide whether the surrounding region belongs to a smooth area. The equations are as

$$TopHalf\_diff = TopHalf\_max - TopHalf\_min \quad (3)$$

$$BottomHalf\ diff = BottomHalf\ max - BottomHalf\_min \quad (4)$$

$$\mathbf{DecisionI} = \begin{cases} true, & if(\ TopHalf\ diff \geq Th\_IM_a) \\ & or\ (\ BottomHalf\_diff \geq Th\_IM_a) \\ false, & otherwise: \end{cases} \quad (5)$$

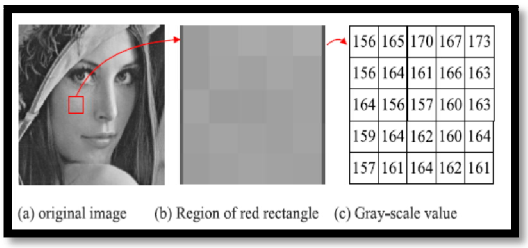
Next, we take $p_{i,j}$ into consideration. Two values must be calculated first. One is the difference between $f_{i,j}$ and *TopHalf_max*; the other is the difference between and $f_{i,j}$ and *TopHalf_min*. After the subtraction, a threshold ThIMb is used to compare these two differences. The same method as in the case of $W_{BottomHalf}$ is applied. The equations are as

$$IM\_TopHalf = \begin{cases} True, & \text{if } (|f\ i,j - TopHalf\_max| \geq Th\_IMb) \\ & \text{or } |f\ i,j - TopHalf\_min| \geq Th\_IMb) \\ false, & \text{otherwise.} \end{cases} \quad (6)$$

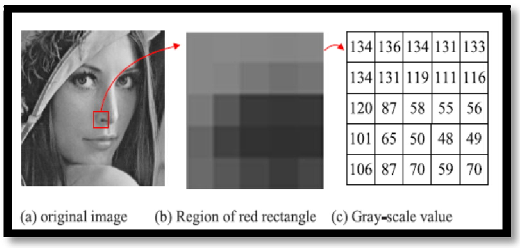
$$IM\_BottomHalf = \begin{cases} true, & \text{if } (|f_{i,j}\_BottomHalf\_max| \geq Th\_IMb) \\ & \text{or } (|f\ i,j\_BottomHalf\_min| \geq Th\_IMb) \\ false, & \text{otherwise.} \end{cases} \quad (7)$$

$$\textbf{\textit{DecisionII}} = \begin{cases} \textit{true}; \text{ if } (\textit{IM\_TopHalf} = \text{ true}) \\ \qquad \text{or } (\textit{IM\_ Bottom\_Half} = \text{ true}) \\ \textit{false}, \text{ otherwise.} \end{cases} \qquad (8)$$

Finally, we can make a temporary decision whether p*i*,*j* belongs to a suspected noisy pixel or is noisy free.

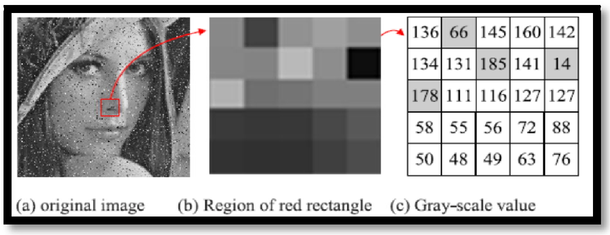


156	165	170	167	173
156	164	161	166	163
164	156	157	160	163
159	164	162	160	164
157	161	164	162	161



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

Fig. 3 A smooth and non smooth region in Lena



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

Fig. 4 The difference between noisy and neighbouring pixels in Lena

2) Fringe Module: If p*i*,*j* has a great difference with neighbouring pixels, it might be a noisy pixel or just situated on an edge, as shown in Figure. 5. How to conclude that a pixel is noisy or situated on an edge is difficult. In order to deal with this case, we define four directions, from E1 to E4, as shown in Figure. 6. We take direction E1 for example. By calculating the absolute difference between $f_{i;j}$ and the other two pixel values along the same direction, respectively, we can determine whether there is an edge or not. The detailed equations are as

$$\begin{array}{lll} \textit{FM\_E1} & = \textit{false}; & \text{if } (|\, a - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or } (|h - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or } (|a - h| \geq \textit{Th\_FMb}) \\ & = \textit{true}, & \text{otherwise.} \end{array} \qquad (9)$$

$$\begin{array}{lll} \textit{FM\_E2} & = \textit{false}; & \text{if } (|c - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or } (|f - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or}(|c - f| \geq \textit{Th\_FMb}) \\ & = \textit{true}; & \text{otherwise:} \end{array} \qquad (10)$$

$$\begin{array}{lll} \textit{FM\_E3} & = \textit{false}; & \text{if } (|b - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or } |g - f_{i,j}| \geq \textit{Th\_FMa}) \\ & & \text{or } (|b - g| \geq \textit{Th\_FMb}) \\ & \textit{true}; & \text{otherwise} \end{array} \qquad (11)$$

$$\begin{array}{lll} \textit{FM\_E4} & = \textit{false}; & \text{if } (|d - f_{i,j}| \geq \text{Th\_FMa}) \\ & & \text{or } (|e - f_{i,j}| \geq \text{Th\_FMa}) \\ & & \text{or } (|d - e| \geq \text{Th\_FMb}) \\ & = \textit{true}; & \text{otherwise} \end{array} \qquad (12)$$

$$\begin{array}{lll} \textbf{\textit{DecisionIII}} & = \textit{false}; & \text{if (FM\_ E1) or (FM\_ E2)} \\ & & \text{or (FM E3) or (FM\_E4)} \\ & = \textit{true}; & \text{otherwise} \end{array} \qquad (13)$$

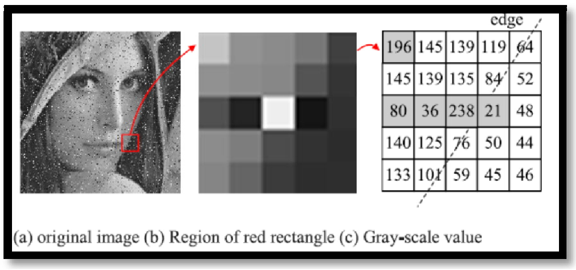


Fig. 5 The edge region in Lena

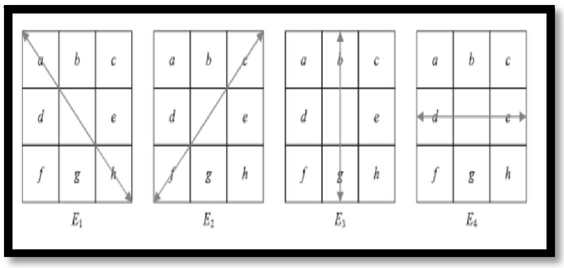


Fig. 6 Four directions in DTBDM

3)Similarity Module: The last module is similarity module. The luminance values in mask *W* located in a noisy-free area might be close. The median is always located in the center of the variational series, while the impulse is usually located near one of its ends. Hence, if there are extreme big or small values, that implies the possibility of noisy signals. According to this concept, we sort nine values in ascending order and obtain the fourth, fifth, and sixth values which are close to the median in mask *W*. The fourth, fifth, and sixth values are represented as 4*thin*W*i;j*, *MedianIn*W*i,j*, and 6*thin*W*i,j*. We define $Max_{i,j}$ and $Min_{i,j}$ as

$$Max_{i,j} = 6thinW_{i,j} + Th\_SM_a,$$

$$Mini,j = 4thinW_{i,j} - Th\_SM_a: \qquad (14)$$

$Max_{i,j}$ and $Min_{i,j}$ are used to determine the status of pixel *pi,j*. However, in order to make the decision more precisely, we do some modifications as

$$Nmax = \begin{cases} Max_{i,j}, & \text{if } (Max_{i,j} \le MedianInW_{i,j} + Th\_SMb) \\ MedianInWi;j + Th\_SMb, & \text{otherwise} \end{cases} \qquad (15)$$

$$Nmin = \begin{cases} Min_{i,j} & \text{if } (Mini,j \ge MedianInWi;j - Th\_SMb) \\ MedianInWi,j - Th\_SMb, & \text{otherwise:} \end{cases} \qquad (16)$$

Finally, if $f\,i_{,j}$ is not between *Nmax* and *Nmin*, we conclude that $p_{i;j}$ is a noise pixel. Edge-preserving image filter will be used to build the reconstructed value. Otherwise, the original value $f\,i_{,j}$ will be the output. The equation is as

$$\boldsymbol{DecisionIV} = \begin{cases} \text{true}, & \text{if } (f_{i,j} \ge \text{Nmax}) \text{ or } (f_{i,j} \le \text{Nmin} \\ \text{false}, & \text{otherwise.} \end{cases}$$

Obviously, the threshold affects the quality of denoised images of the proposed method. A more appropriate threshold contributes to achieve a better detection result. However, it is not easy to derive an optimal threshold through analytic formulation. The fixed values of thresholds make our algorithm simple and suitable for hardware implementation. According to our extensive experimental results, the thresholds *Th_ IMa, Th_ IMb, Th _FMa, Th _FMb, Th _SMa, and Th_ SMb* are all predefined values and set as 20, 25, 40, 80, 15, and 60, respectively.

B Edge-Preserving Image Filter

To locate the edge existing in the current W, a simple edge-preserving technique which can be realized easily with VLSI circuit is adopted. The dataflow of edge-preserving image filter is shown in Figure. 7. Here, we consider eight directional differences, from D1 to D8, to reconstruct the noisy pixel value. Only those composed of noise-free pixels are taken into account to avoid possible misdetection. Directions passing through the suspected pixels are discarded to reduce misdetection. Therefore, we use $Max_{i,j}$ and $Min_{i,j,}$ defined in similarity module, to determine whether the values of *d, e, f, g,* and *h* are likely corrupted, respectively. If the pixel is likely being corrupted by noise, we don't consider the direction including the suspected pixel. In the second block, if *d, e, f, g,* and *h* are all suspected to be noisy pixels, and no edge can be processed, so $f_{\,i,j}$ (the estimated value of $p_{\,i,j}$) is equal to the weighted average of 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 in the third block.The equations are as follows:

$$D1 = |d - h| + |a - e|,\ D2 = |a - g| + |b - h|,\ D3 = |b - g| \times 2,\ D4 = |b - f| + |c - g|,\ D5 = |c - d| + |e - f|,\ D6 = |d - e| \times 2$$
$$D7 = |a - h| \times 2, D8 = |c - f| \times 2 \qquad (17)$$

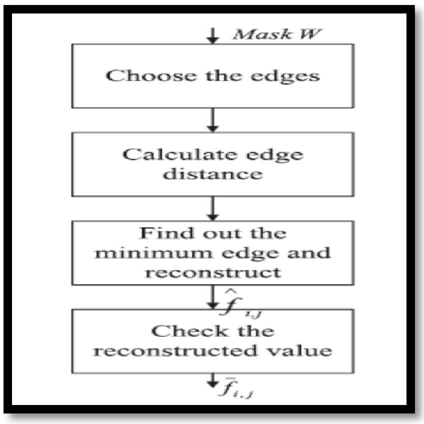
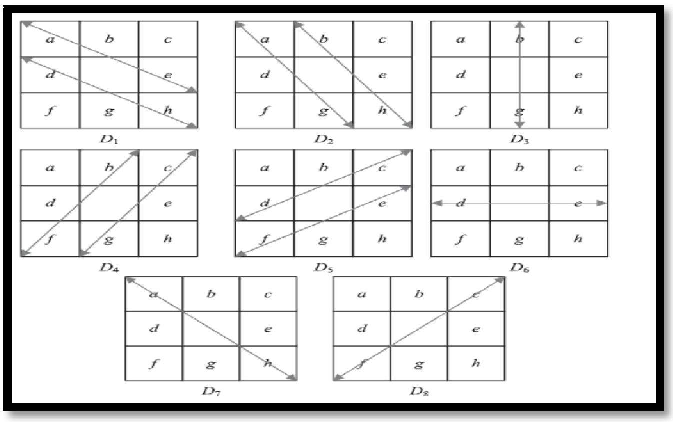

Mask W
Choose the edges
Calculate edge distance
Find out the minimum edge and reconstruct
Check the reconstructed value


Fig. 7 Dataflow of edge-preserving image filter & its eight directional differences

Also $\hat{f}_{i,j}$ =

$$\hat{f}_{i,j} = \begin{cases} (a+d+e+h)/4, & \text{if } D_{min} = D1; \\ (a+b+g+h)/4, & \text{if } D_{min} = D2; \\ (b+g)/2, & \text{if } D_{min} = D3; \\ (b+c+f+g)/4, & \text{if } D_{min} = D4; \\ (c+d+e+f)/4, & \text{if } D_{min} = D5; \\ (d+e)/2, & \text{if } D_{min} = D6; \\ (a+h)/2, & \text{if } D_{min} = D7; \\ (c+f)/2, & \text{if } D_{min} = D8. \end{cases} \quad (18)$$

In the last block the smallest directional difference implies that it has the strongest spatial relation with $p_{i,j}$, and probably there exists an edge in its direction. Hence, the mean of luminance values of the pixels which possess the smallest directional difference is treated as $\hat{f}_{i;j}$. After $\hat{f}_{i;j}$ is determined, a tuning skill is used to filter the bias edge. If $\hat{f}_{i,j}$ obtain the correct edge, it will situate at the median of *b, d, e,* and g because of the spatial relation and the characteristic of edge preserving. Otherwise, the values of $\bar{f}_{i;j}$ will be replaced by the median of four neighbouring pixels (*b, d, e, and g*). We can express $\bar{f}_{i;j}$ as

$$\bar{f}_{i;j} = \text{Median}(\hat{f}_{i;j}, b, d, e, g) \quad (19)$$

III. VLSI IMPLEMENTATION OF DTBDM

DTBDM has low computational complexity and requires only two line buffers instead of full images, so its cost of VLSI implementation is low. For better timing performance, we adopt the pipelined architecture to produce an output at every clock cycle .The architecture adopts an adaptive technology and consists of two main blocks:decision-tree-based impulse detector, edge-preserving image filter. The proposed method employs a histogram equalization to improve the quality of reconstructed image. The decision-tree-based impulse detector is composed of three modules (isolation module, fringe module, and similarity module). Each of them is described in the following sections.

1)Isolation module:

The architecture of IM consists of a comparator CMP_L which is used to output the larger value from the two input values while the comparator CMP_S is used to output the smaller value from the two input values. The first two-level comparators are used to find *TopHalf_max* and *TopHalf_min*. The SUB unit is used to output the difference which is subtracted the lower input *(TopHalf_min)* from the upper one *(TopHalf_max)*, and the |SUB| unit is used to output the absolute value of difference of two inputs. The GC is the greater comparator that will output logic 1 if the upper input value is greater than the lower one. The OR gate is employed to generate the binary result for *IM_TopHalf*. Finally, if the result of Decision II is positive, $p_{i;j}$ might be a noisy pixel or situate on an edge. The next module (FM) will be used to confirm the result.

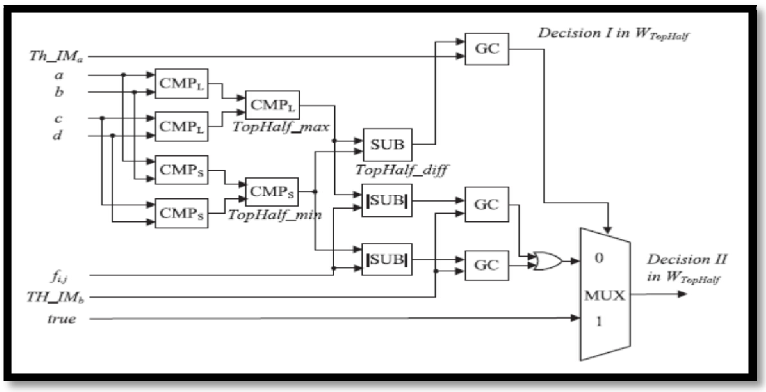


Fig. 8. Architecture of IM

International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 7 (August 2014) http://ijirae.com

2)Fringe Module

The architecture of FM is composed of four small modules, from FM_1 to FM_4, and each of them is used to determine its direction, as mentioned in Section 2.1.2. Fig 5.2.2.2 is a detailed implementation of FM_1. Since E1 is the direction from *a to h* , the relation between *a, h,* and $f_{i,j}$ must be referenced. The three |SUB| units are used to determine the absolute differences between them. The GC is described in the above section and the NOR gate is used to generate the result of *FM _E1*. If the result is positive, we consider that $f_{i,j}$ is on the edge *E1* and regard it as noise free.

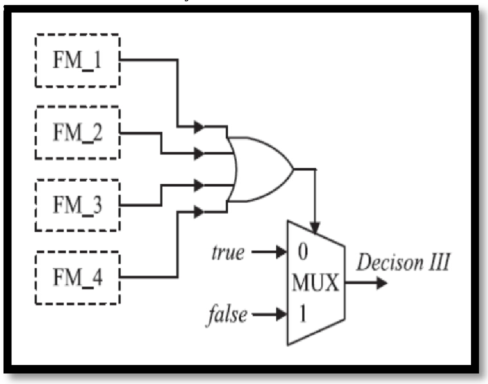


Fig 9 Architecture of FM

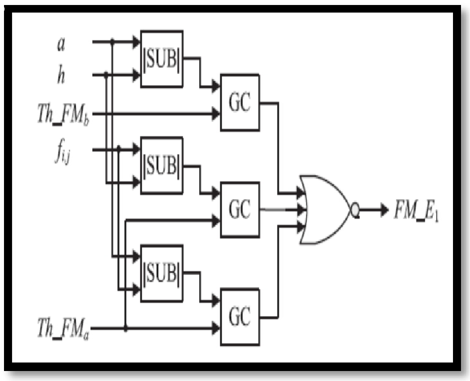


Fig. 10 Architecture of FM_1 module

3)Similarity Module:

If IM and FM can't determine whether *f i,j* belongs to a noisefree value or not, SM is used to confirm the result. This architecture that is designed to accelerate the, sorting speed to obtain the fourth value in mask W. The detailed implementation of module M0 is also shown. If a is greater than b, *C01* is set to 1; otherwise, *C01* is set to 0. The eight GC units are used to determine the values from *C01 to C08*. After comparing, a combined unit is used to combine the results of each comparator to obtain a number between 0 and 8. The number indicates the order of value in mask W. If a is the smallest value in mask W, the output of the M0 module is 0; if a is the biggest value in mask W, the output is 8. The architectures of other modules (M1 to M8) are almost the same as M0, with only little difference.

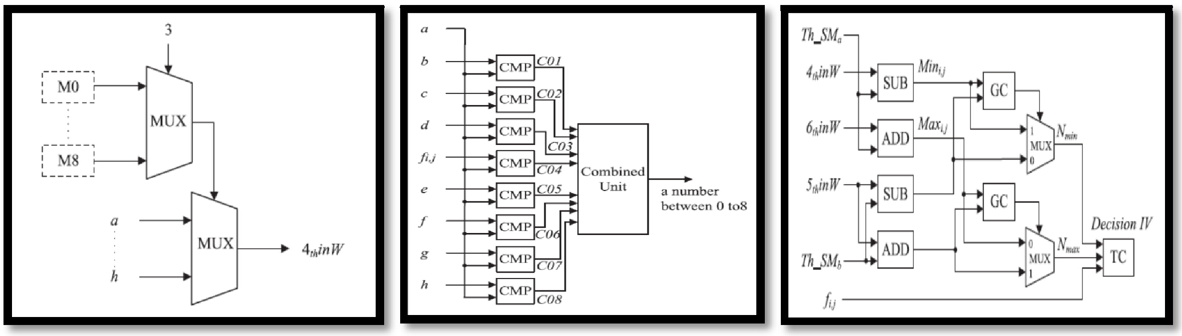


Fig 11 Architecture of SM

4)Edge-Preserving Image Filter

The Edge-Preserving Image Filter is composed of two modules, minED generator and average generator (AG). The architecture consists of minED generator which is used to determine the edge that has the smallest difference. Eight directional differences are calculated with twelve |SUB|, four ADD, and four shifter units. Then, the smallest one is determined by using the Min Tree unit. Min Tree is made up of a series of comparators. After that, the mean of luminance values of the pixels which process the smallest directional difference (Dmin) can be obtained from the average generator.

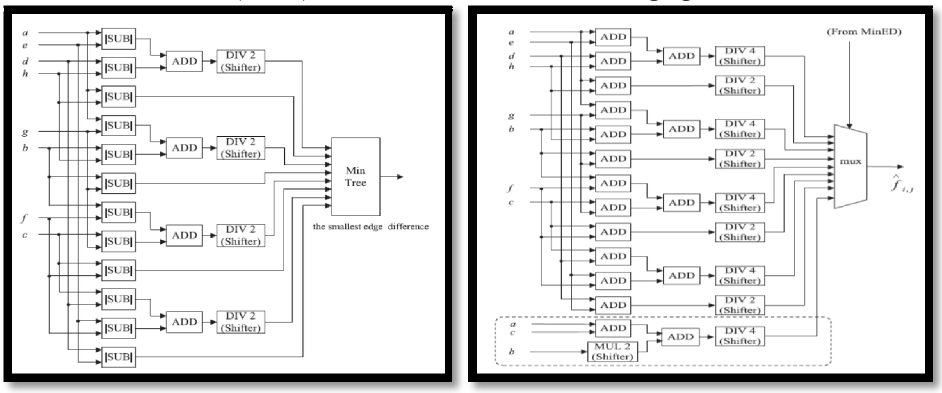


Fig 12 Architecture of minED Generator & Average generator

As mentioned already if $p_{i;j-1}$; $p_{i;j+1}$; $p_{i+1;j-1}$; $p_{i+1;j}$ and $p_{i+1,\, j+1}$ are all suspected to be noisy pixels, the final MUX will output (a +b x 2 +c)/4. Otherwise, the MUX will output the mean of the pixel values which process Dmin. Some directional differences are determined according to four pixel values, so its reconstructive values also need four pixel values. Two-level ADD and shifter units are used to complete our calculation. As for the chip implementation, the gate counts or silicon area of one multiplier or division is much larger than one shifter. In our design, all multipliers or divisions will be replaced by shifter units in order to lower the hardware cost. After above computations, we sort *b, d, e, and g* in order. The reconstructed value $\hat{f}_{i;j}$ obtained from edge-preserving filter will be compared with the second and third values, named as *SortFour2, SortFour3,* and the final value $\bar{f}_{i;j}$ is obtained from the equation as

$$\bar{f}_{i;j} = \begin{cases} SortFour2; & \text{if } (SortFour2 > \hat{f}_{i;j}) \\ SortFour3; & \text{if } (SortFour3 < \hat{f}_{i;j}) \\ \bar{f}_{i;j}, & \text{otherwise.} \end{cases} \qquad (20)$$

5) *Histogram Equalization*

Image enhancement is a mean as the improvement of an image appearance by increasing dominance of some features or by decreasing ambiguity between different regions of the image. Image enhancement processes consist of a collection of techniques that seek to improve the visual appearance of an image or to convert the image to a form better suited for analysis by a human or machine Numerous enhancement methods have been proposed but the enhancement efficiency, computational requirements, noise amplification, user intervention, and application suitability are the common factors to be considered when choosing from these different methods for specific image processing application. Histogram processing is the act of altering an image by modifying its histogram. Common uses of histogram processing include normalization by which one makes the histogram of an image as flat as possible. This is also known as contrast enhancement. Intensity transformation functions based on information extracted from image such as enhancement, compression, segmentation and description.

Histogram Equalization is a technique that generates a gray map which changes the histogram of an image and redistributing all pixels values to be as close as possible to a user –specified desired histogram. HE allows for areas of lower local contrast to gain a higher contrast. Histogram equalization automatically determines a transformation function seeking to produce an output image with a uniform Histogram. Histogram equalization is a method in image processing of contrast adjustment using the image histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. Histogram equalization automatically determines a transformation function seeking to produce an output image with a uniform Histogram.

IV. RESULTS AND OBSERVATION

A. *Simulation Results*

The performance analysis of all architectures are analysed and it is simulated using ModelSim simulator .

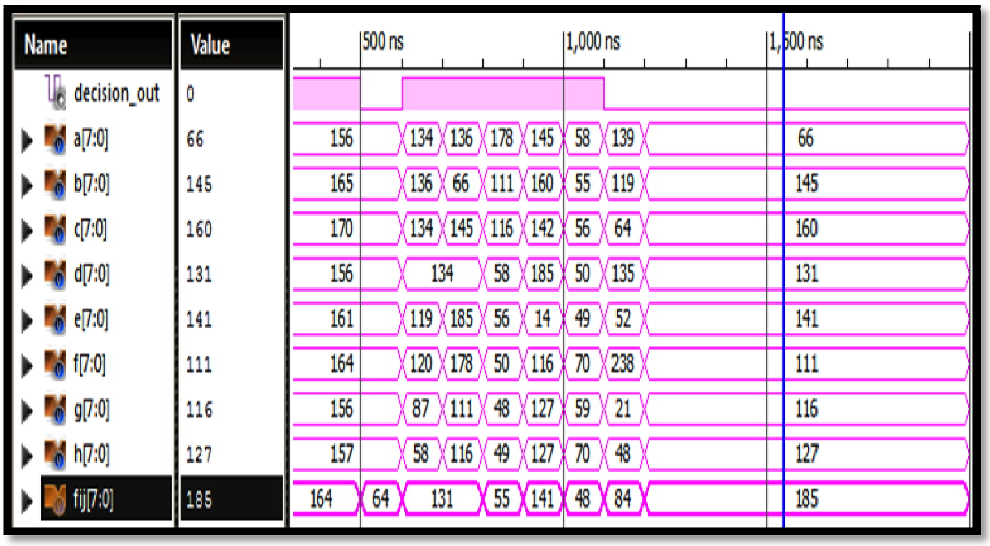
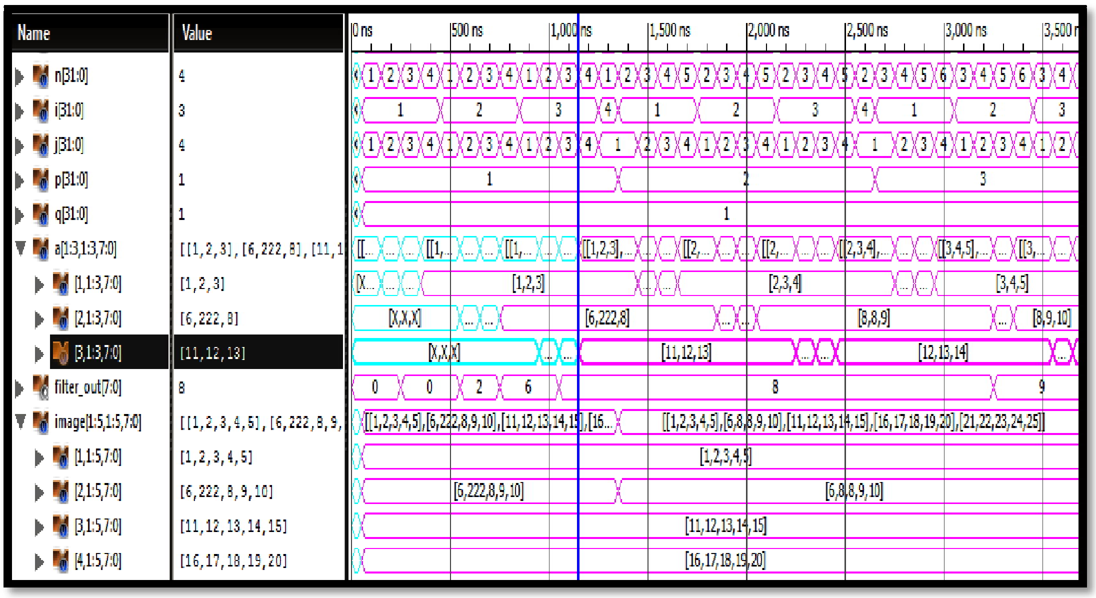
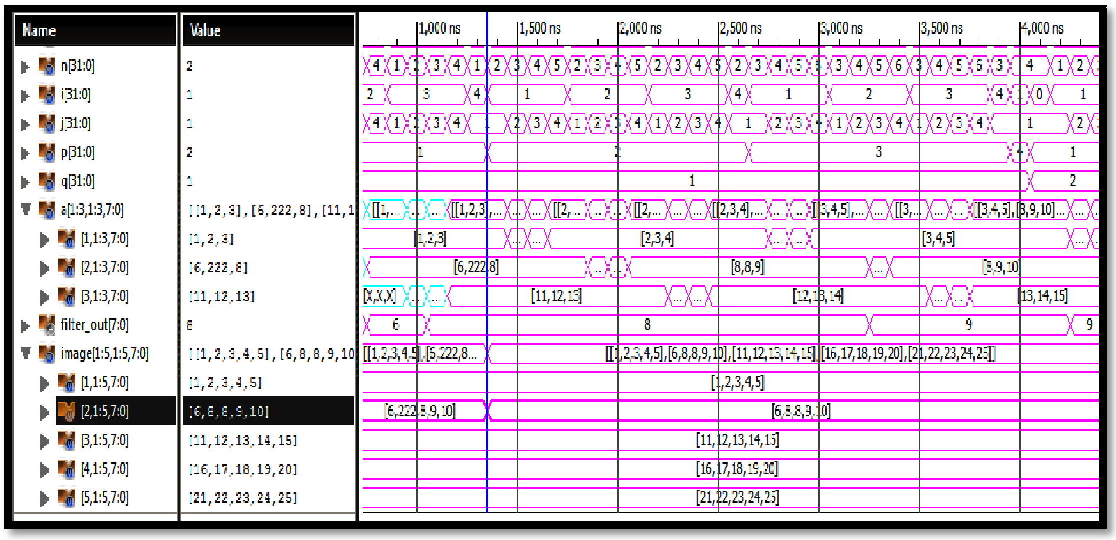


Fig.13 Simulation waveform of Decision Tree



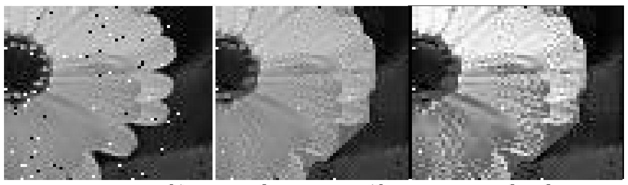
Name	Value
n[31:0]	4
i[31:0]	3
j[31:0]	4
p[31:0]	1
q[31:0]	1
a[1:3,1:3,7:0]	[[1,2,3],[6,222,8],[11,1
[1,1:3,7:0]	[1,2,3]
[2,1:3,7:0]	[6,222,8]
[3,1:3,7:0]	[11,12,13]
filter_out[7:0]	8
image[1:5,1:5,7:0]	[[1,2,3,4,5],[6,222,8,9,
[1,1:5,7:0]	[1,2,3,4,5]
[2,1:5,7:0]	[6,222,8,9,10]
[3,1:5,7:0]	[11,12,13,14,15]
[4,1:5,7:0]	[16,17,18,19,20]

Fig.14 Simulation waveform of DTBDM



Name	Value
n[31:0]	2
i[31:0]	1
j[31:0]	1
p[31:0]	2
q[31:0]	1
a[1:3,1:3,7:0]	[[1,2,3],[6,222,8],[11,1
[1,1:3,7:0]	[1,2,3]
[2,1:3,7:0]	[6,222,8]
[3,1:3,7:0]	[11,12,13]
filter_out[7:0]	8
image[1:5,1:5,7:0]	[[1,2,3,4,5],[6,8,8,9,10
[1,1:5,7:0]	[1,2,3,4,5]
[2,1:5,7:0]	[6,8,8,9,10]
[3,1:5,7:0]	[11,12,13,14,15]
[4,1:5,7:0]	[16,17,18,19,20]
[5,1:5,7:0]	[21,22,23,24,25]

Fig.15 Simulation waveform of DTBDM



a)Noisy image b)Denoised image c)histogram equalized image

Fig.16 Resultsof denoising method and image restoration with image enhancement

V. Conclusions

A low-cost VLSI architecture for efficient removal of random-valued impulse noise is proposed in this paper. The approach uses the decision-tree-based detector to detect the noisy pixel 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 complexity methods and is comparable to the higher complexity methods in terms of both quantitative evaluation and visual quality. It requires only low computational complexity and two line memory buffers. Therefore, it is very suitable to be applied to many real-time applications. Furthermore, histogram equalization is used to enhance the effects of removal of impulse noise.

ACKNOWLEDGMENT

I am grateful to Mr.Bibin Binu Simon, Assistant Professor, Department of Electronics and Communication Engineering, Mangalam College of Engineering, for his innovative ideas in proceeding with this proposed system.

REFERENCES

[1] Chih-Yuan Lien,Chien-Chuan Huang,Pei-Yin Chen, and Yi-Fan Lin “An Efficient Denoising Architecture for Removal of Impulse Noise in Images”,IEEE Transactions on computers,vol.62,no.4,April 2013.

[2] S. Zhang and M.A. Karim, “A New Impulse Detector for Switching Median Filter,” IEEE Signal Processing Letters, vol. 9, no. 11, pp Nov. 2002.

[3] R.H. Chan, C.W. Ho, and M. Nikolova, “Salt-and-Pepper Noise Removal by Median-Type Noise Detectors and Detail-Preserving Regularization,” IEEE Trans. Image Processing, vol. 14, no. 10, pp. 1479-1485, Oct. 2005..

[4] P.-Y. Chen and C.-Y. Lien, “An Efficient Edge-Preserving Algorithm for Removal of Salt-and-Pepper Noise,” IEEE Signal Processing Letters, vol. 15, pp. 833-836, Dec. 2008.

[5] T. Sun and Y. Neuvo, “Detail-Preserving Median Based Filters in Image Processing,” Pattern Recognition Letters, vol. 15, pp. 341-347, Apr. 1994.

[6] W. Luo, “An Efficient Detail-Preserving Approach for Removing Impulse Noise in Images,” IEEE Signal Processing Letters, vol. 13, no. 7, pp. 413-416, July 2006.