Denoising Techniques In VLSI For Impulse Noise Removal

Sherin N R[1]
PG Scholar
ECE Department
Younus College of Engineering and Technology
Kerala

Mr. Riyas A N[2]
Asst. Professor
ECE Department
Younus College of Engineering and Technology
Kerala

Mr. Rajeev S K[3]
Head of the Department
ECE Department
Younus College of Engineering and Technology
Kerala

***Abstract*— During the process of signal acquisition and transmission image and video signals might be affected by impulse noises. Nowadays many image denoising methods exists for impulse noise suppression. Here an efficient VLSI denoising technique is introduced. The method is a low complexity one and therefore its hardware cost is low. Moreover it will result in better image quality than any other technique. In addition to this another approach called decision-tree-based denoising method (DTBDM) is also included.**

***Keywords*— Image denoising, impulse noise, impulse detector, pipeline architecture,VLSI.**

I. INTRODUCTION

IN image processing many applications such as image segmentation, face recognition, printing skills, medical imaging, scanning techniques, etc has to face the problem that images are corrupted by noise in the process of image acquisition and transmission. The noise may seriously affect the performance of image processing techniques. Hence, in such situations an efficient denoising technique is very necessary. Recently, many image de-noising methods have been proposed to carry out the impulse noise suppression [2]–[17]. Some of them employ the standard median filter [2] or its modifications [3], [4] to implement denoising process. 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.There have been many methods for removing salt-and-pepper noise, and some of them perform very well [4]-[21]. The random-valued impulse noise is more difficult to handle due to the random distribution of noisy pixel values. We only focus on removing the random-valued impulse noise from the corrupted image in this paper.

In general, the switching median filter [5] consists of two steps: 1)impulse detection and 2)noise filtering. It locates the noisy pixels with an impulse detector, and then filters them .In addition to median filter, there are other methods used to carry out impulse noise. In [9], Wenbin Luo proposed an alpha-trimmed mean based method (ATMBM). It used the alpha-trimmed mean in impulse detection and replaced the noisy pixel value by a linear combination of its original value and the median of its local window. A differential rank impulse detector (DRID)was presented in [7]. The impulse detector of DRID is based on a comparison of signal samples within a narrow rank window by both rank and absolute value.In [17], Petrović NI and Crnojević proposed a method that employed genetic programming for impulse noise filter construction.

Generally, the denoising methods can be classified into two categories: lower-complexity techniques [8]-[13] and higher-complexity techniques [14]-[18]. 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. 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. Based on these two factors, we propose a simple edge-preserved denoising technique (SEPD) and its simulation for removing fixed-value impulse noise. The storage space needed for SEPD is two line buffers rather than a full frame buffer. Only simple arithmetic operations, such as addition and subtraction, are used in SEPD. We proposed a useful impulse noise detector to detect the noisy pixel and employ an effective design to locate the edge of it. The simulation results demonstrate that SEPD can obtain better performances in terms of both quantitative evaluation and visual quality. This simulation can also be obtained by using another technique called decision-tree-based denoising method

(DTBDM). 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.

The rest of this paper is organized as follows. In Section II, the SEPD is introduced. The VLSI architecture of SEPD is described briefly in Section III. In Section IV, the architecture of reduced SEPD is introduced. In section V the new technique DTBDM is presented.The simulation result is provided in Section VI. Conclusion is presented in Section VII.

II. Simple Edge Preservsed Denoising (SEPD)

Consider the current pixel to be denoised which is located at coordinate $(i,j)$and denoted as $p_{i,j}$and its luminance values before and after the denoising process are represented as $f_{i,j}$ and $\bar{f}_{i,j}$, respectively. The luminance value of $p_{i,j}$will jump to the minimum or maximum value in gray scale if it is corrupted by fixed-value impulse noise. Here a 3 ×3 mask W centering on $p_{i,j}$is considered for image denoising. In the current W,the three values at coordinates $(i-1,j-1)$, $(i-1,j)$ and $(i-1,j+1)$ are denoised at the previous denoising process, and the six pixels at coordinates $(i,j-1)$, (i,j), $(i,j+1)$,$(i+1,j-1)$, $(i+1,j)$ and $(i+1,j+1)$ are not denoised yet, as shown in Fig. 1. A pipelined hardware architecture is adopted in the design. Using the 3 x 3 values in W,SEPD will determine whether $p_{i,j}$ s a noisy pixel or not. If positive, SEPD locates a directional edge existing in W and uses it to determine the reconstructed value $\bar{f}_{i,j}$otherwise, $\bar{f}_{i,j} = f_{i,j}$.

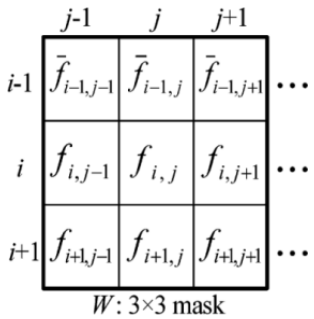

Fig. 1. 3× 3 mask centered on $p_{i,j}$.

SEPD consists of three components: extreme data detector, edge-oriented noise filter and impulse arbiter. The extreme data detector detects the minimum and maximum luminance values in W, and determines whether the luminance values of $p_{i,j}$ and its five neighboring pixels are equal to the extreme data. By observing the spatial correlation, the edge-oriented noise filter pinpoints a directional edge and uses it to generate the estimated value of current pixel. Finally, the impulse arbiter brings out the proper result. Fig. 2 shows the pseudo code of SEPD. The three components of SEPD are described in detail in the following subsections.

A. Extreme Data Detector

The extreme data detector detects the minimum and maximum luminance values (MINinW and MAXinW) in those processed masks from the first one to the current one in the image. The luminance value of a pixel will jump to the minimum or maximum value in gray scale, if it is corrupted by the fixed-value impulse noise. If $f_{i,j}$is not equal to MINinW/MAXinW, then conclude that $p_{i,j}$ is a noise-free pixel and the steps for denoising are skipped. If $f_{i,j}$ is equal to MINinW or MAXinW, then set p to1, check whether its five neighboring pixels are equal to the extreme data, and store the binary compared results into B , as shown in Fig. 2.

B. Edge-Oriented Noise Filter

To locate the edge existed in the current W, a simple edge catching technique is adopted. To decide the edge, consider 12 directional differences, from D_1 to D_{12}, as shown in Fig. 3. Here,
only noise-free pixels are taken into account to avoid misdetection. If a bit in B is equal to 1, it means that the pixel related to the binary flag is suspected to be a noisy pixel. Directions passing through the suspected pixels are not considered in order to reduce misdetection. In each condition, at most four directions are chosen for low-cost hardware implementation. Fig. 4 shows the mapping table between B and the chosen directions adopted in the design.

```
for(i = 0; i < row; i = i + 1)   /* input image size : row(height) × col(width) */
{
   for(j = 0; j < col; j = j + 1)
   {

/* Extreme Data Detector */
      Get W, the 3 × 3 mask centered on (i, j);
      Find MINinW and MAXinW in W;
      /* the minimum and maximum values from the first W to current W */
      φ = 0;                                  /* initial values */
      if ((f_{i,j} = MINinW) or (f_{i,j} = MAXinW))
            φ = 1;       /* p_{i,j} is suspected to be a noisy pixel */
      if (φ = 0)
            {f̄_{i,j} = f_{i,j}; break;}          /* p_{i,j} is a noise - free pixel */
      B = b1b2b3b4b5 ="00000";                /* initial values */
      if ((f_{i,j-1} = MINinW) or (f_{i,j-1} = MAXinW))
            b1 = 1;       /* p_{i,j-1} is suspected to be a noisy pxiel */
      if ((f_{i,j+1} = MINinW) or (f_{i,j+1} = MAXinW))
            b2 = 1;       /* p_{i,j+1} is suspected to be a noisy pxiel */
      if ((f_{i+1,j-1} = MINinW) or (f_{i+1,j-1} = MAXinW))
            b3 = 1;       /* p_{i+1,j-1} is suspected to be a noisy pxiel */
      if ((f_{i+1,j} = MINinW) or (f_{i+1,j} = MAXinW))
            b4 = 1;       /* p_{i+1,j} is suspected to be a noisy pxiel */
      if ((f_{i+1,j+1} = MINinW) or (f_{i+1,j+1} = MAXinW))
            b5 = 1;       /* p_{i+1,j+1} is suspected to be a noisy pxiel */

/* Edge - Oriented Noise Filter */
      Use B to determine the chosen directions across p_{i,j} according to Fig. 4;
      if (B = "11111")
            f̂_{i,j} = (f̄_{i-1,j-1} + 2 × f̄_{i-1,j} + f̄_{i-1,j+1}) / 4;      /* no edge is considered */
      else
            { Find D_min (the smallest directional difference among the chosen directions);
            f̂_{i,j} = the mean of luminance values of the two pixels which own D_min ; }

/* Impulse Arbiter */
      if (| f_{i,j} − f̂_{i,j} | > T_s)
            f̄_{i,j} = f̂_{i,j};                  /* p_{i,j} is judged as a noisy pixel */
      else
            f̄_{i,j} = f_{i,j};                  /* p_{i,j} is judged as a noise - free pixel */
   }
}
```

Fig. 2. Pseudo code of our scaling method.

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 (B= "11111") no edge can be processed, so $\hat{f}_{i,j}$ (the estimated value of $p_{i,j}$) is equal to the weighted average of luminance values of three previously denoised pixels and calculated as $(\bar{f}_{i-1,j-1} + 2 \times \bar{f}_{i-1,j} + \bar{f}_{i-1,j+1})/4$. In other conditions except when B= "11111 " the edge filter calculates the directional differences of the chosen directions and locates the smallest one (D_{min}) among them, as shown in Fig. 2. The smallest directional difference shows that it has the strongest spatial relation with $p_{i,j}$, and probably ther exists an edge in its direction. Hence, the mean of luminance values of the two pixels which possess the smallest directional difference is treated as $\hat{f}_{i,j}$.

For example, if B is equal to "10011," it means that $f_{i,j-1}$, $f_{i+1,j}$ and $f_{i+1,j+1}$ are suspected to be noisy values. Therefore $D_2 - D_5$, D_5 and $D_9 - D_{11}$ are discarded because they contain those suspected pixels (see Fig.3). The four chosen directional differences are D_1, D_6, and D_{12} (see Fig. 4). Finally, $\hat{f}_{i,j}$ is equal to the mean of luminance values of the two pixels which possess the smallest directional difference among D_1, D_6, D_8 and D_{12}.

C. Impulse Arbiter

Since the value of a pixel corrupted by the fixed value impulse noise will jump to be the minimum/maximum value in gray scale, then we can conclude that if $p_{i,j}$ is corrupted, $f_{i,j}$ is equal to MINinW or MAXinW . However, the converse is not true. If $f_{i,j}$ is equal to MINinW or MAXinW , $p_{i,j}$ may be corrupted or just in the region with the highest or lowest luminance value. In other words, a pixel whose value is MINinW or MAXinW might be identified as a noisy pixel even if it is not corrupted. To avoid this condition, we add another condition that, if $p_{i,j}$ is a noise-free pixel and the current mask has high spatial correlation, $f_{i,j}$ should be close to $\hat{f}_{i,j}$ and $|f_{i,j} - \hat{f}_{i,j}|$ is small. That is, $p_{i,j}$ might be a noise- free pixel but the pixel value is MINinW or MAXinW if $|f_{i,j} - \hat{f}_{i,j}|$ is small. So, in this case we measure $|f_{i,j} - \hat{f}_{i,j}|$ and compare it with a threshold to determine whether $p_{i,j}$ is corrupted or not. The threshold, denoted as T_s, is a predefined value. Here the threshold is set to be 20. If $p_{i,j}$ is judged as a corrupted pixel, the reconstructed luminance value $\bar{f}_{i,j}$ is equal to $\hat{f}_{i,j}$; otherwise, $\bar{f}_{i,j} = f_{i,j}$.

III. ARCHITECTURE OF SEPD

SEPD has low computational complexity and requires only two line buffers, so its cost of VLSI implementation is low. For better timing performance, we adopt the pipelined architecture which can produce an output at every clock cycle.

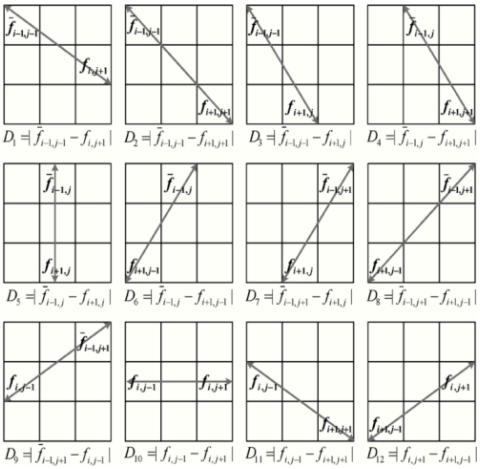

Fig. 3. Twelve directional differences of SEPD.

Fig. 5 shows block diagram of the 7-stage pipeline architecture for SEPD. The architecture consists of five main blocks: line buffer, register bank, extreme data detector, edge-oriented noise filter and impulse arbiter. Each of them is described briefly in the following subsections.

A. Line Buffer

SEPD adopts a 3x3 mask, so three scanning lines are needed. If $p_{i,j}$ are processed, three pixels from row_{i-1}, row_i and row_{i+1} are considered. With the help of four crossover multiplexers (see Fig. 5), we realize three scanning lines with two line buffers. As shown in Fig. 5, Line Buffer-odd and Line Buffer-even are used to store the pixels at odd and even rows, respectively.

Once the denoising process for $p_{i,j}$ is completed, the reconstructed pixel value $\bar{f}_{i,j}$ generated by the arbiter is outputted and written into the line buffer storing rowi to replace $f_{i,j}$. When

B	the chosen directions	B	the chosen directions
00000	D_2, D_5, D_8, D_{10}	10000	D_2, D_5, D_8, D_{12}
00001	D_3, D_5, D_8, D_{10}	10001	D_1, D_5, D_8, D_{12}
00010	D_2, D_8, D_{10}, D_{12}	10010	D_2, D_4, D_8, D_{12}
00011	D_1, D_6, D_8, D_{10}	10011	D_1, D_6, D_8, D_{12}
00100	D_2, D_5, D_7, D_{10}	10100	D_1, D_2, D_5, D_7
00101	D_3, D_5, D_7, D_{10}	10101	D_1, D_5, D_7
00110	D_2, D_4, D_9, D_{10}	10110	D_1, D_2, D_4
00111	D_1, D_9, D_{10}	10111	D_1
01000	D_2, D_5, D_8, D_{11}	11000	D_2, D_5, D_6, D_8
01001	D_3, D_5, D_7, D_9	11001	D_3, D_5, D_6, D_8
01010	D_2, D_6, D_8, D_{11}	11010	D_2, D_4, D_6, D_8
01011	D_6, D_8, D_9	11011	D_6, D_8
01100	D_2, D_5, D_9, D_{11}	11100	D_2, D_4, D_5, D_7
01101	D_3, D_5, D_9	11101	D_3, D_5, D_7
01110	D_2, D_4, D_9, D_{11}	11110	D_2, D_4
01111	D_9	11111	N/A

N/A: not available

Fig. 4. Thirty-two possible values of B and their corresponding directions in SEPD.

B. Register Bank

The register bank (RB), consisting of 9 registers, is used to store the 3x3 pixel values of the current mask. Fig. 6 shows its architecture where Reg4 keeps the luminance value $(f_{i,j})$ of the current pixel to be denoised. Obviously, the denoising process for $p_{i,j}$ does not start until $f_{i+1,j+1}$enterers from the input device. The nine values stored in RB are then used simultaneously by subsequent extreme data detector and noise filter for denoising.

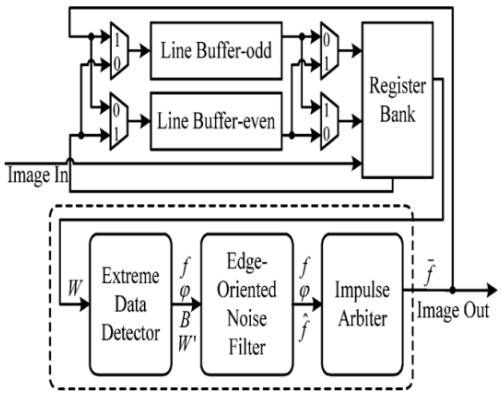

Fig. 5. Block diagram of VLSI architecture for SEPD.

Once the denoising process shifts from $p_{i,j}$ to $p_{i,j+1}$, only 3 new values $(f_{i-1,j+2}, f_{i,j+2}, f_{i+1,j+2})$ are needed to be read into RB (Reg2, Reg5 and Reg8, respectively) and other 6 pixel values are shifted to each one's proper register. At the same time, the previous value in Reg8 (the previous input value from the input device, $f_{i+1,j+1}$

) is written back to the line buffer storing row_{i-1} for subsequent denoising process. The selection signals of the four multiplexers are all set to 1 or 0 for denoising the odd or the even rows, respectively. Two examples are shown in Fig. 7 to illustrate we denoise row_2, set all four selection signals to 0, and those samples of row_1 and row_2 are stored in Line Buffer-odd and Buffer-even respectively. The samples of row_3 are inputted from the input device, as shown in Fig. 7(b). The previous interconnections between the two line buffers and RB. When value in Reg8 and the denoised results generated by the arbiter are written back to Line Buffer-odd and Line Buffer-even, respectively. After the denoising process of row_2 has been completed, Line Buffer-odd is filled with the whole samples of row_3, while those denoised samples of row_2 are all stored in Line Buffer-even. To denoise row_3, we set all selections signals to 1. Thus the previous value in Reg8 and the denoised results generated by the arbiter are written back to Line Buffer-even and Line Buffer-odd, respectively, as shown in Fig. 7(a). After the denoising process of row_3 has been completed, Line Buffer-even is now full with the whole samples of row_4, while those denoised samples of row_3 are stored in Line Buffer-odd.

C. Extreme Data Detector

Fig. 8 shows the 3-stage pipeline architecture of the extreme data detector in which P represents the pipeline register and EC is the equality comparator. The 2-stage min-max tree module is used to find MINinW and MAXinW. Two columns of EC units are used to determine whether the lower six pixels in W are equal to MINinW or MAXinW, respectively. W' represents the eight neighboring pixels values of $p_{i,j}$ in W . The six OR gates are employed to generate the binary comparison results p and b_1-b_5 When p=0, it means $p_{i,j}$ is a noise-free pixel and the following operations can be skipped.

D. Edge-Oriented Noise Filter

Fig. 9 shows the 2-stage pipeline architecture of the edge-oriented noise filter in which the $|SUB|$ unit is used to output the absolute value of difference of two inputs. Using B from the detector, the mapping module implements the table shown in Fig. 4. Four directional differences are calculated with the four $|SUB|$units. Then the smallest one is determined by using the minimum tree unit. After that, the mean of luminance values of the two pixels which possess the smallest directional difference (D_{min}) can be obtained. When "B=11111 " the final multiplexer willoutput$\left(f_{i-1,j-1} + 2 \times f_{i-1,j} + f_{i-1,j+1}\right)/4$.When "B $\neq$ 11111"
the multiplexer will output the D_{min} .

E. Impulse Arbiter

Fig. 10 shows the architecture of the impulse arbiter in which comparator CMP is used to output logic 1 if the upper input value $\left|f_{i,j} - \hat{f}_{i,j}\right|$is greater than the lower one (T_s). The final multiplexer is used to output $\hat{f}_{i,j}$ generated by the noise filter when $p_{i,j}$ is corrupted, or $f_{i,j}$ when $p_{i,j}$ is noise-free.

IV. ARCHITECTURE OF REDUCED SEPD

In SEPD, we consider 12 directional differences to decide the proper edge. When more edges are considered, more complex computations are required. To further reduce the cost of implementation, we modify SEPD and propose another design, named as reduced SEPD (RSEPD). Only three directional differences D_a, D_b and D_c as shown in Fig. 11, are considered in RSEPD. As demonstrated in Section V, RSEPD offers slightly poorer image quality but requires much lower cost than SEPD. Fig. 12 shows the pseudo code of RSEPD.

The VLSI architecture of RSEPD is also composed of five main blocks. RB, line buffer and impulse arbiter of RSPED are identical to those of SEPD; however, there are some difference between the extreme detector and noise filter and those of SEPD:

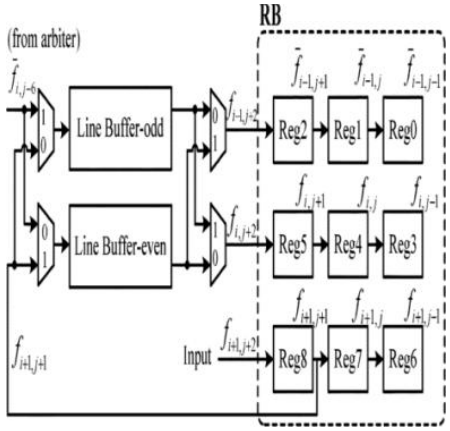

Fig. 6. Architecture of register bank in SEPD.

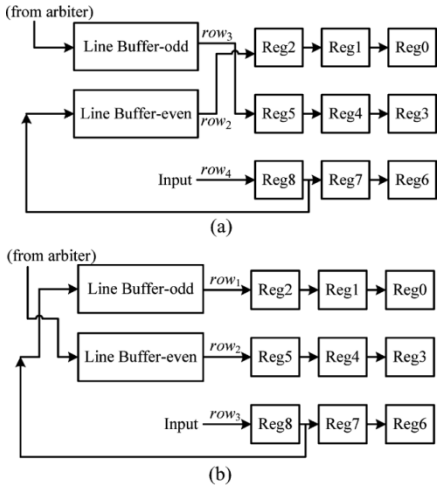

Fig. 7. Two examples of the interconnections between two line buffers and RB.

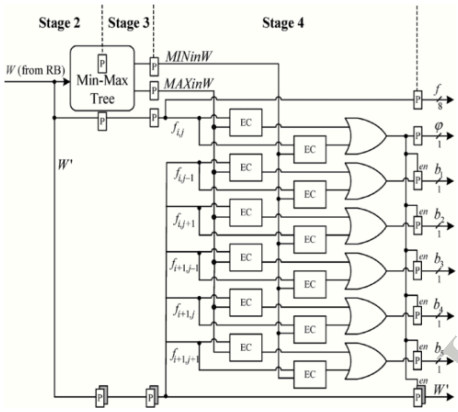

Fig. 8. Architecture of extreme data detector in SEPD.

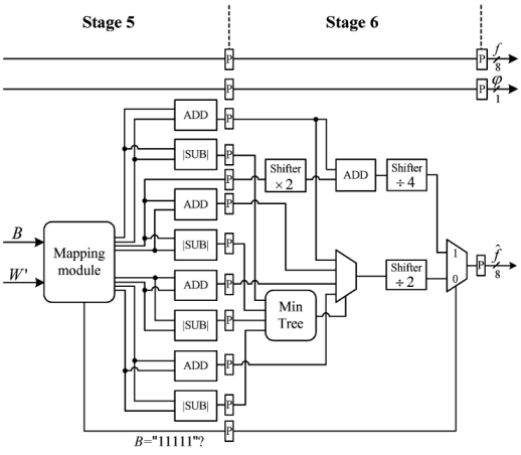

Fig. 9. Two-stage pipeline architecture of edge-oriented noise filter.

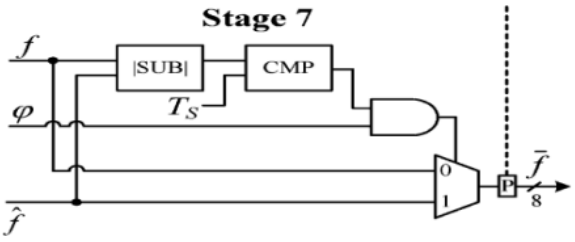

Fig. 10. Architecture of impulse arbiter.

1) In the extreme data detector of RSEPD, only one neighboring pixel $p_{i+1,j}$ is considered for edge preservation. As shown in Fig.13, the extreme data detector of RSEPD contains four EC units where two EC units are used to determine p and another two are used to determine b.

2) In the noise filter of RSEPD, three directional differences are considered. The mapping module used in SEPD becomes unnecessary and is removed. As shown in Fig. 14, we can apply three $|SUB|$units to calculate three differences, and then determine the smallest one with the minimum tree unit.

3) In RSEPD, both the extreme data detector and noise filter need three clock cycles to complete their functions. They work in parallel because there exists no data dependency between them. As shown in Fig. 10, the impulse arbiter requires one clock cycle to complete its job.

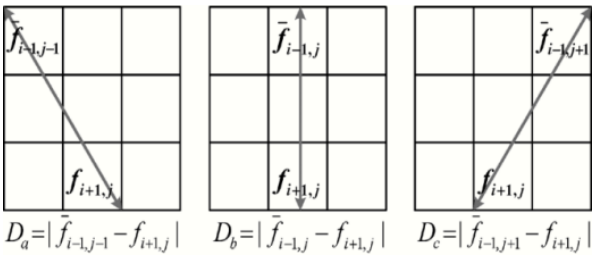

Fig. 11. Three directional differences in RSEPD.

```
for (i = 0; i < row; i = i + 1)      / * Input image size : row(height) × col(width) * /
{
   for (j = 0; j < col; j = j + 1)
   {

/ * Extreme Data Detector * /
      Get W, the 3 × 3 mask centered on (i, j);
      Find MINinW and MAXinW in W;
      / * the minimum and maximum values from the first W to current W * /
      φ = 0;                                  / * initial values * /
      if ((f_{i,j} = MINinW) or (f_{i,j} = MAXinW))
            φ = 1;            / * p_{i,j} is suspected to be a noisy pixel * /
      if (φ = 0)
            { f̄_{i,j} = f_{i,j}; break;}          / * p_{i,j} is a noise - free pixel * /
      b = 0;                                  / * initial values * /
      if ((f_{i+1,j} = MINinW) or (f_{i+1,j} = MAXinW))
            b = 1;            / * p_{i+1,j} is suspected to be a noisy pixel * /

/ * Edge - Oriented Noise Filter * /
      if (b = 1)
            f̂_{i,j} = (f̄_{i-1,j-1} + 2 × f̄_{i-1,j} + f̄_{i-1,j+1}) / 4;     / * no edge is considered * /
      else
            { Find D_min (the smallest directional difference among D_a, D_b and D_c);
            f̂_{i,j} = the mean of luminance values of the two pixels which own D_min ;}

/ * Impulse Arbiter * /
      if (| f_{i,j} − f̂_{i,j} | > T_S)
            f̄_{i,j} = f̂_{i,j};     / * p_{i,j} is judged as a noisy pixel * /
      else
            f̄_{i,j} = f_{i,j};     / * p_{i,j} is judged as a noise - free pixel * /
   }
}
```

Fig. 12. Pseudo code of RSEPD method.

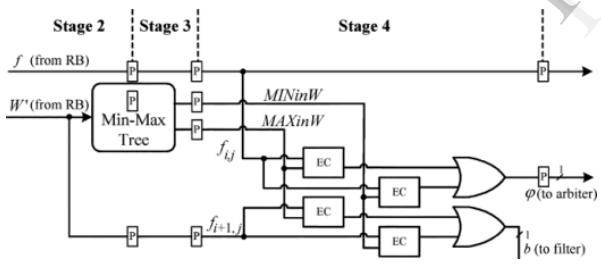

Fig. 13. Extreme data detector in RSEPD

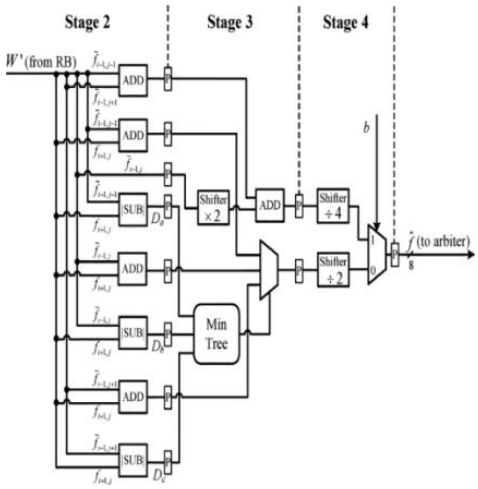

Fig. 14. Noise filter in RSEPD.

V. DECISION-TREE-BASED DENOISING METHOD (DTBDM)

The complexity in decision making process of above techniques is rectified by introducing a new technique called Decision-Tree-Based Denoising method.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 [6], [20], [16]-[21] 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.

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 neighboring 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.

VI. SIMULATION RESULTS

To verify the characteristics and performances of various denoising algorithms, a variety of simulations are carried out on a 255×255 camera man test image. In the simulations, images are corrupted by impulse noise

(salt-and-pepper noise). The proposed VLSI architectures of SEPD, RSEPD and DTBDM were implemented by using VHDL. MODELSIM was used for simulation.

(a) noisy image (b) SEPD (c) RSEPD (d) DTBDM

Fig. 15. Results of different methods in restoring corrupted image "Camera man".

VII. CONCLUSION

In this paper, an efficient VLSI technique for ramdom-valued impulse noise removal is presented. The extensive simulation results demonstrate that our design achieves excellent performance in terms of quantitative evaluation and visual quality. For real-time applications, a 7-stage pipeline architecture for SEPD and a 5-stage pipeline architecture for RSEPD are also developed and simulated. As the outcome demonstrated, RSEPD outperforms other chips [11]–[13], [19], [20] with the lowest hardware cost. The architectures work with monochromatic images, but they can be extended for working with RGB color images and videos. Here as a part of modification a new approach is also introduced. The approach uses the decision-tree-based detector to detect the noisy pixel and employs an effective design to locate the edge. It requires only low computational complexity and two line memory buffers. Therefore, it is very suitable to be applied to many real-time applications.

ACKNOWLEDGMENT

The authors would like to thank the management, and Faculty Members, of Department of Electronics and Communication Engineering, Younus College of Engineering and Technology, Kollam for many insightful discussions and the facilities extended to us for completing the task.

REFERENCES

[1] T. Nodes and N. Gallagher, "Median filters: Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-30, no. 5, pp. 739–746, Oct. 1982.
[2] 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, Sep. 1991.
[3] H. Hwang and R. Haddad, "Adaptive median filters: New algorithms and results," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 499–502, Apr. 1995.
[4] T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," *Pattern Recog. Lett.*, vol. 15, no. 4, pp. 341–347, Apr. 1994.
[5] S. Zhang and M. A. Karim, "A new impulse detector for switching median filter," *IEEE Signal Process. Lett.*, vol. 9, no. 11, pp. 360–363, Nov. 2002.
[6] 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.
[7] W. Luo, "Efficient removal of impulse noise from digital images," *IEEE Trans. Consum. Electron.*, vol. 52, no. 2, pp. 523–527, May 2006.
[8] W. Luo, "An efficient detail-preserving approach for removing impulse noise in images," *IEEE Signal Process. Lett.*, vol. 13, no. 7, pp. 413–416, Jul. 2006.
[9] K. S. Srinivasan and D. Ebenezer, "A new fast and efficient decisionbased algorithm for removal of high-density impulse noises," *IEEE Signal Process. Lett.*, vol. 14, no. 3, pp. 189–192, Mar. 2007.
[10] S.-C. Hsia, "Parallel VLSI design for a real-time video-impulse noise-reduction processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 4, pp. 651–658, Aug. 2003.
[11] I. Andreadis and G. Louverdis, "Real-time adaptive image impulse noise suppression," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 3, pp. 798–806, Jun. 2004.
[12] V. Fischer, R. Lukac, and K. Martin, "Cost-effective video filtering solution for real-time vision systems," *EURASIP J. Appl. Signal Process.*,vol. 2005, no. 13, pp. 2026–2042, Jan. 2005.
[13] Z. Wang and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 78–80, Jan. 1999.
[14] 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 Process.*, vol. 14, no. 10, pp. 1479–1485, Oct. 2005.
[15] P.-E. Ng and K.-K. Ma, "A switching median filter with boundary discriminative noise detection for extremely corrupted images," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1506–1516, Jun. 2006.
[16] 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, Jul. 2008.
[17] R. H. Chan, C. Hu, and M. Nikolova, "An iterative procedure for removing random-valued impulse noise," *IEEE Signal Process. Lett.*, vol. 11, no. 12, pp. 921–924, Dec. 2004.
[18] C.-Y. Lee, P.-W. Hsieh, and J.-M. Tsai, "High-speed median filter designs using shiftable content-addressable memory," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 6, pp. 544–549, Dec. 1994.
[19] C.-T. Chen, L.-G. Chen, and J.-H. Hsiao, "VLSI implementation of a selective median filter," *IEEE Trans. Consum. Electron.*, vol. 42, no. 1, pp. 33–42, Feb. 1996.
[20] 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 Process.*, vol. 14, no. 10, pp. 1479–1485, Oct. 2005.
[21] P.-Y. Chen and C.-Y. Lien, "An Efficient Edge-Preserving Algorithm for Removal of Salt-and-Pepper Noise," *IEEE Signal Process. Lett.*, vol. 15, pp. 833- 836, Dec. 2008.