International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 1 No: 3 March 2016

An Effective FPGA Implementation of Denoising of Impulse Noise in Images

A.Jeniffer[1], S.Haripasath[2], S.Chinthamani[3], G.Chitra[4], V.Karthiga[5]
[1,2,3,4,5]Department of Electronics and Communication Engineering
Saranathan College of Engineering, Trichy, India
[1]ajeniffer94@gmail.com, [2]hariprasadh@yahoo.com, [3]chintha93@gmail.com

Abstract: Noise filtering and image enhancement are two applications of great importance in the field of image processing. The final image is an essential part of any image processor, whether it is used for visual interpretation or automatic analysis. Images are degraded by various types of noise during image acquisition and transmission. One such noise affecting nearly all images is the noise of the impulse. In this work, an efficient detection and removal scheme is proposed for the presence of noise from impulses in images. Subsequently, the VLSI architecture is also proposed using a Virtex family of FPGAs. In the proposed work in 4 stages the noise detection and reconstruction of the pixel from noise is performed. Compared to the existing techniques, the amount of hardware calculations is reduced in the proposed work, which could therefore be used for real - time applications.

Keywords: Edge Preserving Filter, FPGA Implementation, Impulse noise, Image Denoising, Sorting.

1. INTRODUCTION

In the fields of Medicine, Forensics, Remote Sensing, Communication, Industrial Automation, Defense, Robotics, Traffic Control, etc.,. Image Processing is an assuring area of research. The acquired image should be de-blurred and noise free in order to have a very good visual display in different applications. Noise corrupts images during the acquisition process, transmission, storage and retrieval, so it is important to effectively suppress the noise without perverting the image's edges and fine details. Digital images are often corrupted due to transmission errors, malfunctioning pixel elements in camera sensors, defective memory locations and analog to digital conversion timing errors. An important feature of pulse noise is that only part of the pixels are corrupted and the rest are free of noise. Noise from impulses is fixed and randomly valued. In the noise of a fixed impulse, the noisy pixel values in gray - scale images are either maximum (white, 255) or minimum (black,0). The image therefore contains dark and white spots, so it is also called the noise of salt and pepper. For gray - scale images, the values of noisy pixels corrupted by randomly valued impulse noise are distributed uniformly in the range [0 to 255].

2. EXISTING METHODS

Different methods for identification of presenceof noise in the image under consideration werereported earlier [3],[4],[5],[6],[7]. In the works discussed in [3],[4] mean filters are used which cause blurring of the image. Hence, it is not suitable. In J. Ko's work[7] the weight values are assigned for individual pixels and using median filters impulse noise was removed but here both noisy and noise free pixels are affected. In another work reported by F. Ahmed [5], alpha Trimmed Mean Filter is used. The disadvantage is for lower values of d(a parameter used here) it resembles mean filter but for higher values of d it resembles median filter.In the work proposed by I. Aizenberg and C. Butakoff [6] rank order based method is used for noise detection and removal. But here the disadvantage is if a pixel is not corrupted but still has highest or lowest rank it will be identified as an impulse.
Hence an effective switching based median filter is used in this paper. This paper's content is organized as follows. The steps in our proposed method are described in Section III. Section IV provides the results of the proposed work for simulation and synthesis. Finally, the future work, conclusions and perspectives are provided in section V and VI.

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 1 No: 3 March 2016

3. PROPOSED METHOD

In our design, we have taken an input noisy image of size (256×256) 8- bit grey scale image and a mask of size (3×3). The (3×3) mask is shown is figure 1.

a	b	c
d	f (i,j)	e
f	g	h

Figure 1. 3X3 portion of the image under consideration

First the mask is applied to the top left corner of the image and only that portion of the image is extracted and processed accordingly. Then the mask is slided successively to next pixel location to cover the entire image and is processed. Let the pixels covered by the mask be labeled as fi,j for center pixel and a ,b ,c ,d ,e ,f,g ,h for neighboring pixels. The mask's adjacent pixels are divided into the top and bottom half. The top-half values consists of {a, b, c, d} and the bottom-half values consists of {e , f, g ,h } and is given by
Tophalf = {a, b, c, d}; Bottomhalf = {e , f, g ,h }
The decision whether or not the pixel is noisy is based on the correlation between the pixel fi, j and its adjacent pixel. This detection is based on whether the pixels are located on smooth region or not. A region is said to be a smooth region if the pixels values are slightly varying with each other, i.e the difference between the pixel values with their neighbors is small. This help to isolate the pixel value If the gray scale value difference between the pixel fi,j and the neighboring pixels is large it is an isolation point.

4. ISOLATION MODULE

In order to determine the isolated point from the smooth region, the maximum value and minimum value in the top half { a, b, c, d }, the maximum value and minimum value in the bottom half { e, f, g, h } are also determined. The difference between the maximum value and the minimum value in the top-half is calculated in order to obtain the top-half_diff. Similarly the difference between the maximum value and minimum value in bottom-half is calculated to obtain bottom-half_diff. This difference is calculated to determine whether the surrounding region belongs to a smooth region or not. A threshold value is now used to determine whether or not the pixels are in the smooth area. So the selection of the threshold value is an important factor. Determine the threshold value Th_IMa. If, either top-half_diff value or bottom-half_diff value is greater than the threshold value Th_IMa is considered noisy or noise - free. To decide whether the pixel is in a smooth region, the isolation module is used.

$$TopHalf\_diff = TopHalf\_max - TopHalf\_min$$

$$BottomHalf\_diff = BottomHalf\_max - BottomHalf\_min$$

$$DecisionI = \begin{cases} true, if (TopHalf\_diff \geq Th\_IMa) \\ \quad or (BottomHalf\_diff \geq Th\_IMa) \\ false, otherwise \end{cases}$$

DecisionI tells whether the surrounding region belongs to a smooth region. Next, we take centre pixel into consideration. The difference between fi, j and TopHalf max and the difference between fi,j and TopHalf_min is compared to another Th_IMb threshold.

$$IM\_TopHalf = \begin{cases} true, if (|fi,j - TopHalf\_max| \geq Th\_IMb) \\ \quad or (|fi,j - TopHalf\_min| \geq Th\_IMb) \\ false, otherwise \end{cases}$$

$$IM\_BottomHalf = \begin{cases} true, if (|fi,j - BottomHalf\_max| \geq Th\_IMb) \\ \quad or (|fi,j - BottomHalf\_min| \geq Th\_IMb) \\ false, otherwise \end{cases}$$

$$DecisionII = \begin{cases} true, if (IM\_TopHalf = true) \\ \quad or (IM\_BottomHalf = true) \\ false, otherwise \end{cases}$$

The combined result of decision I and decision II tells us whether the centre pixel is an isolation point or not.

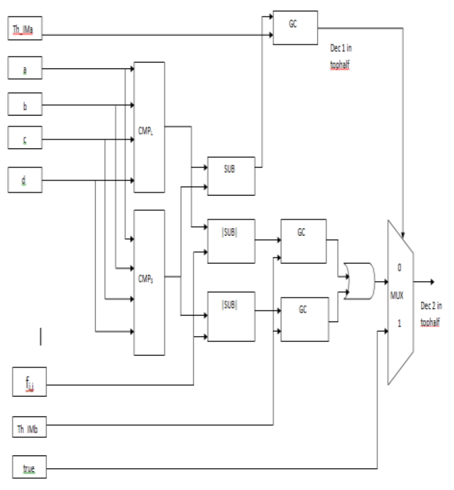


Figure 2. Architecture of Isolation module(TopHalf)

The architecture for the Isolation Module consists of the following components as shown in fig 2. Two comparators CMPl and CMPs are used.CMPl is used to output the larger value for top-half and bottom-half.

The CMPs comparator is used to display the smaller value for the top half and bottom half. The subtractor is used to determine the difference between the max value and the min value in the top and bottom half. The greater comparator is used for comparing these subtracted values with threshold valuleTh_IMa for both top-half and bottom-half. |SUB| is used to find the absolute value of difference with the outputs obtained from CMPl and CMPs with fi,j for top-half and bottom-half. Then the greater comparator is used for comparing the above two values with the threshold Th_IMb for both top-half and bottom-half. The output from these two greater comparator is OR ed to obtained the output for IM_Top Half and IM _Bottom Half. Then it is multiplexed to get the required output of whether the pixel is an isolation point or not.

5. FRINGE MODULE

The fringe module is used to determine whether there is an edge or not. If the center pixel value differs greatly from the adjacent pixel value, it is necessary to determine whether it is a noisy pixel or an edge pixel, so that the fringe module is used for this purpose. For the determination of the edge we consider four directions E1,E2,E3,E4 as shown .

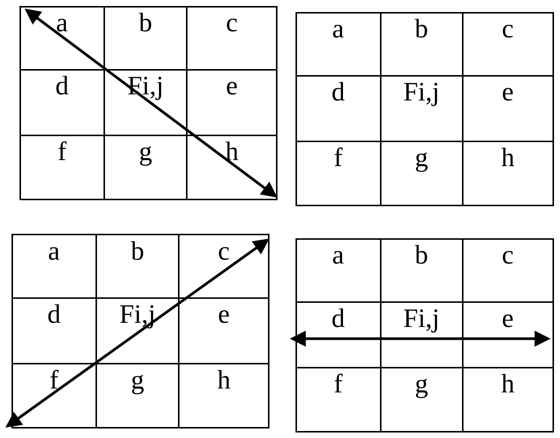


Figure 3. Four directions

We can determine the edge by finding the absolute difference between f (i,j) and the other two pixels in each direction. The equations are given below,

$$FM_{E1} = \left\{ \begin{array}{ll} noisy, & if(|a - fi,j| \geq Th_{FMa}) \\ & or\ (|h - fi,j| \geq Th_{FMa}) \\ & or(|a - h| \geq Th_{FMb}) \\ true, & otherwise \end{array} \right\}$$

$$FM_{E2} = \left\{ \begin{array}{ll} noisy, & if(|c - fi,j| \geq Th_{FMa}) \\ & or\ (|f - fi,j| \geq Th_{FMa}) \\ & or(|c - f| \geq Th_{FMb}) \\ true, & otherwise \end{array} \right\}$$

$$FM_{E3} = \left\{ \begin{array}{ll} noisy, & if(|b - fi,j| \geq Th_{FMa}) \\ & or\ (|g - fi,j| \geq Th_{FMa}) \\ & or(|b - g| \geq Th_{FMb}) \\ true, & otherwise \end{array} \right\}$$

$$FM_{E4} = \left\{ \begin{array}{ll} noisy, & if(|d - fi,j| \geq Th_{FMa}) \\ & or\ (|e - fi,j| \geq Th_{FMa}) \\ & or(|d - e| \geq Th_{FMb}) \\ true, & otherwise \end{array} \right\}$$

$$decision\ III = \left\{ \begin{array}{ll} noisefree, & if(FM_{E1})\ or\ (FM_{E2}) \\ & or(FM_{E3})or(FM_{E4}) \\ noisy, & otherwise \end{array} \right\}$$

The edge differences are calculated in each direction and it is compared with the thresholdsTh_FMa and Th_FMb. The result from all the directions are combined and a decision is made whether the pixel is in edge or not. If it is an edge pixel, then it is a noisefree pixel.

The architecture of fringe module is shown in fig.4.The four sub modules FM_1, FM_2, FM_3 and FM_4 each determining the directions E1, E2, E3 and E4 respectively are combined to form the Fringe Module output.

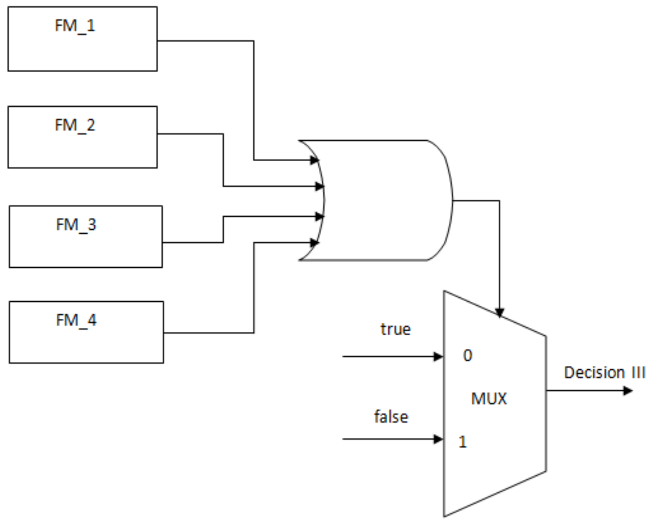


Figure 4. Architecture of Fringe Module

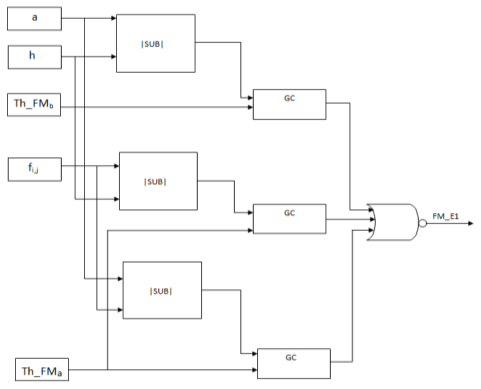


Figure 5. Architecture of FM_1 Module

The fig 5 shows the detailed architecture of FM_1 which corresponds to the direction E1. It consists of absolute subtractor, greatest comparator and a NOR gate. Since the FM 1 determines the E1, it consists of pixels a, h and fi,j. To determine the absolute difference between the three |SUB| units. If the upper input is greater than the lower input for all three |SUB| units, the larger comparator will output logic 1. The output from the greater comparator is NORed to produce the FM_E1. If the result is positive fi,j is on an edge. The same procedure is carried out for FM_2, FM_3 and FM_4 with each sub module including the pixels in that direction.

6. SIMILARITY MODULE

To confirm whether the the pixel is noisy or noisy free, the similarity module is used. This considers that the median value will be located in the center while the noise will be at the end of the variational series. In order to detect whether the pixel is noisy or not, we sort the values in the mask in ascending order to obtain the 4th, 5th and 6th values close to the median in the mask. The max and min value are found by considering the following equation.

$$Max_{i,j} = 6_{th} in\, W_{i,j} + Th\_SM_a$$

$$Min_{i,j} = 4_{th} in\, W_{i,j} - Th\_SM_a$$

A max and min limit is set so as to determine whether the value is noisy pixel or not. The equations are given below

$$Nmax = \begin{cases} Max_{i,j}, & if(Max_{i,j} \le MedianInW_{i,j} + Th\_SM_b) \\ MedianInW_{i,j} + Th\_SM_b, & otherwise \end{cases}$$

$$Nmin = \begin{cases} Min_{i,j}, & if(Min_{i,j} \ge MedianInW_{i,j} - Th\_SM_b) \\ MedianInW_{i,j} - Th\_SM_b, & otherwise \end{cases}$$

If the value of f(i,j) is not between Nmax and Nmin, it is considered to be a noisy pixel otherwise noise - free. The equation is presented by

$$DecisionIV = \begin{cases} true, if(f_{i,j} \ge N_{max}) or\ (f_{i,j} \le N_{min}) \\ false, otherwise \end{cases}$$

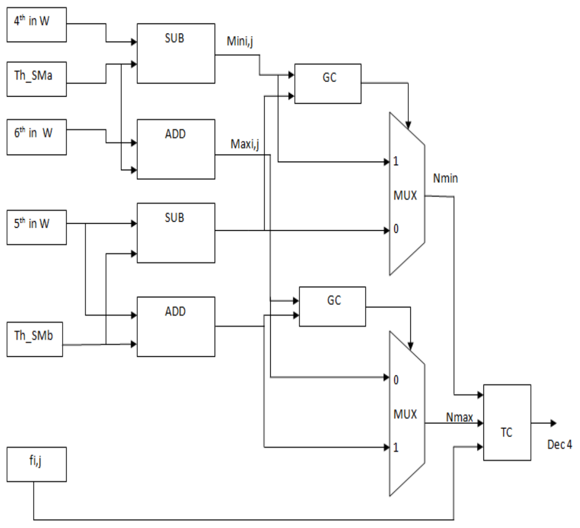


Figure 6. Architecture of Similarity Module

The architecture of similarity module is shown in fig. 6. It consists of adder and subtractors for calculating the maximum and minimum values and greatest comparators and mux for determining whether the pixel is within particular range or not.

7. EDGE PRESERVING FILTER

An edge preserving technique is used to recover the noisy pixel and at the same time preserve fine details like edges. Here first order derivative of edge preserving filter is used. For this we consider eight directional differences D_1 to D_8. This edge preserving filter first calculates these directional differences and find the minimum directional difference. The eight directional differences are given below

$$D1 = |d - h| + |a - e|$$
$$D2 = |a - g| + |b - h|$$
$$D3 = |b - g| * 2$$
$$D4 = |b - f| + |c - g|$$
$$D5 = |c - d| + |e - f|$$
$$D6 = |d - e| * 2$$
$$D7 = |a - h| * 2$$
$$D8 = |c - f| * 2$$

If the directional difference is small, there will be edge existing in that direction. So the mean luminance value in that direction is calculated. The equations for calculating mean luminance value is given by,

$$f'i,j = \begin{cases} \dfrac{a+d+e+h}{4}, & if\ Dmin = D1, \\ \dfrac{a+b+g+h}{4}, & if\ Dmin = D2, \\ \dfrac{b+g}{2}, & if\ Dmin = D3, \\ \dfrac{b+c+f+g}{4}, & if\ Dmin = D4, \\ \dfrac{c+d+e+f}{4}, & if\ Dmin = D5, \\ \dfrac{d+e}{2}, & if\ Dmin = D6, \\ \dfrac{a+h}{2}, & if\ Dmin = D7, \\ \dfrac{c+f}{2}, & if\ Dmin = D8. \end{cases}$$

Then we have used a standard median filter for reconstructing the pixel precisely and the median is for f'i,j and 4-neighborhood pixels as given in below equation,
Fi,j=median (f'i.j, b, d, c, g)
The architecture of Edge preserving filter consists of minED generator and average edge generator.

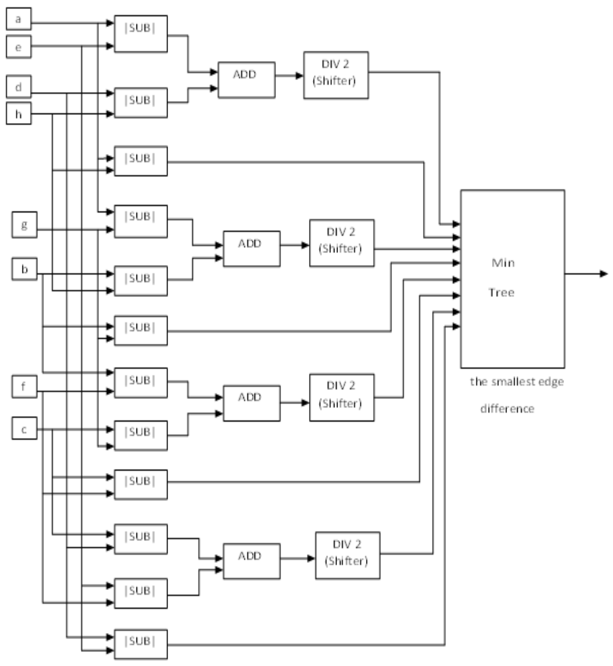


Figure 7. Architecture of minED generator

The minED generator consists of twelve |SUB|, four ADD four shifter units. The smallest directional difference is determined by the min tree unit which consists of a series of comparators. The output of this block is smallest directional difference.

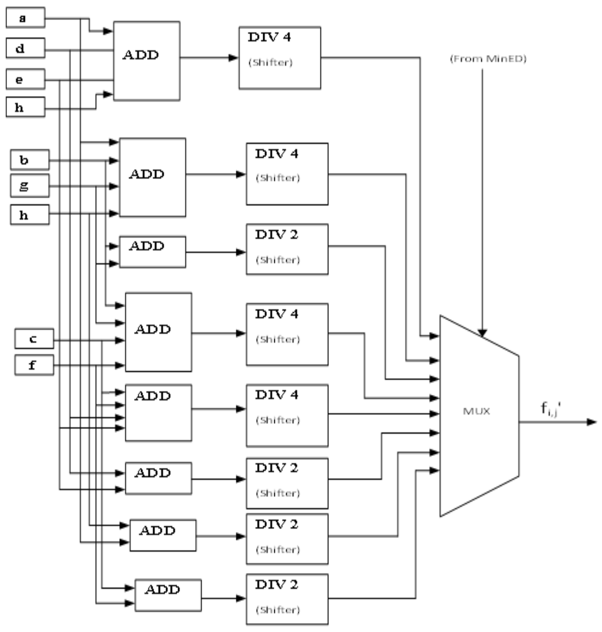


Figure 8. Architecture of average generator

The average output of the generator is the mean luminance values of the pixels that process the smallest directional difference (Dmin) and the average generator architecture is shown in fig 8. After average luminance is calculated, f'i, j, b, d, e and g are sorted to obtain the median value.

8. RESULTS

The simulation output is shown in fig9. Here there are 4 cases, each are explained separately below.

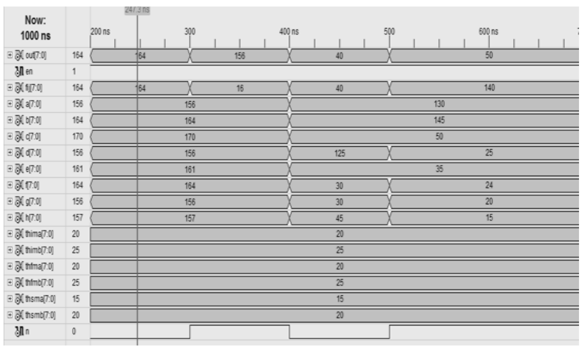


Figure 9. Simulation for (3x3) portion of the image

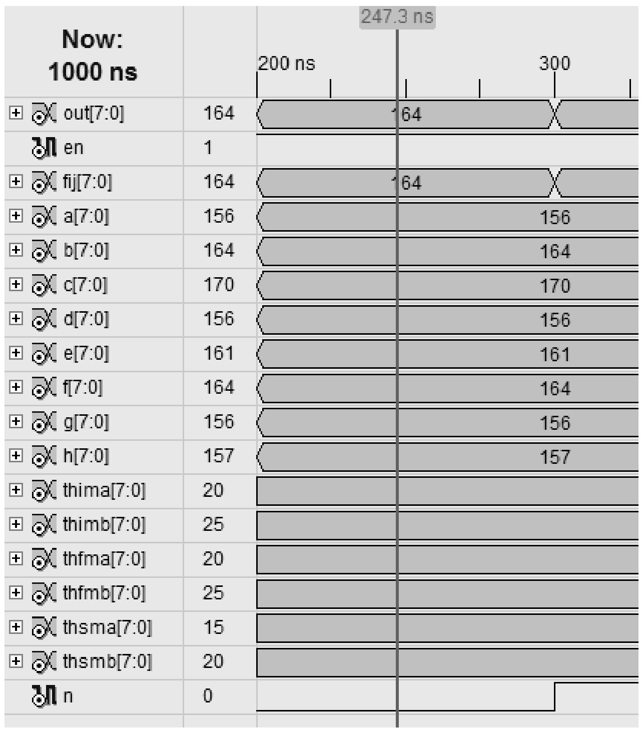
International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 1 No: 3 March 2016

CASE 1:
From 200 to 300 ns, the pixel is in smooth region. There is no noise in the centre pixel, so the detector output 'n' will be logic 0 and the same centre pixel (fi,j) is returned. For example the screen shot is shown at 247.3ns in fig10.

CASE 2:
From 300 to 400 ns, the pixel is in smooth region but here the centre pixel is noisy, so the detector output 'n' will be logic 1. The pixel is reconstructed using edge preserving filter and it is returned. For example the screen shot is shown at 360.7 ns in fig 11.

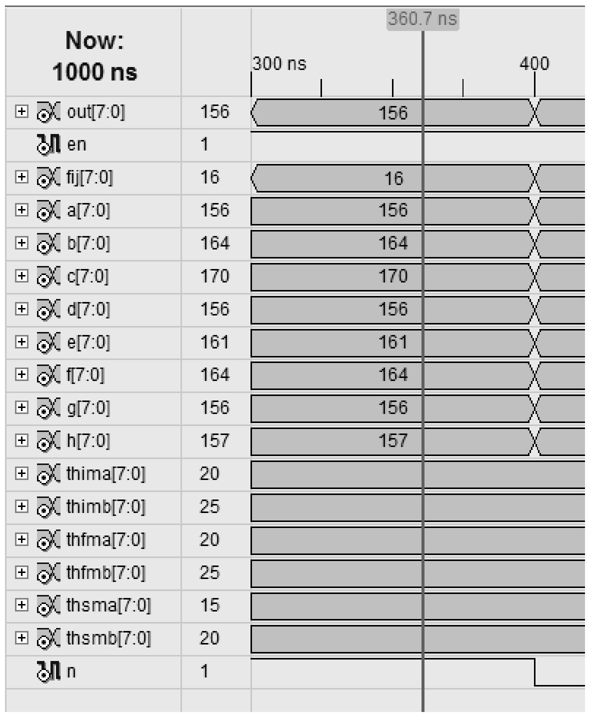
CASE 3:
From 400 to 500 ns, the pixel is in edge region. There is no noise in the centre pixel, so the detector output 'n' will be logic 0 and the same centre pixel (fi,j) is returned. For example the screen shot is shown at 441.3ns in fig 12.

CASE 4:
From 500 to 600 ns, the pixel is in edge region but here the centre pixel is noisy, so the detector output 'n' will be logic 1. The pixel is reconstructed using edge preserving filter and it is returned. For example the screen shot is shown at 558.1 ns in fig 13.



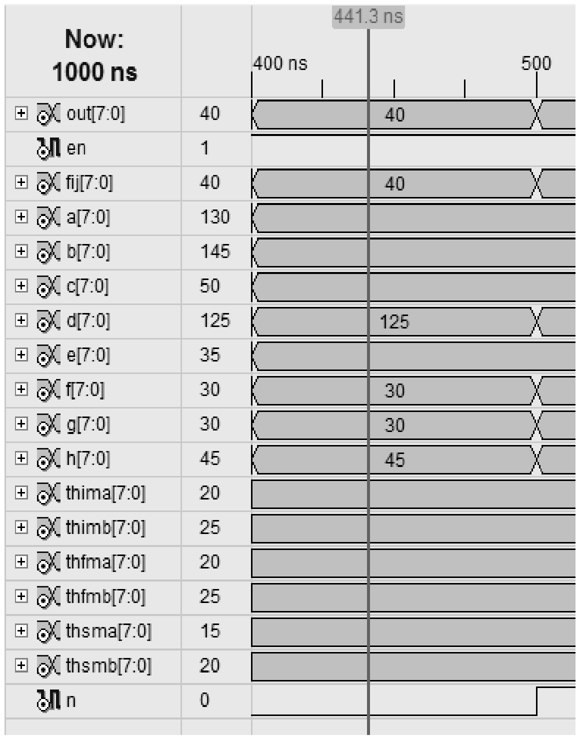
Now: 1000 ns	
out[7:0]	164
en	1
fij[7:0]	164
a[7:0]	156
b[7:0]	164
c[7:0]	170
d[7:0]	156
e[7:0]	161
f[7:0]	164
g[7:0]	156
h[7:0]	157
thima[7:0]	20
thimb[7:0]	25
thfma[7:0]	20
thfmb[7:0]	25
thsma[7:0]	15
thsmb[7:0]	20
n	0

Figure 10. Simulation for pixel in smooth region



Now: 1000 ns	
out[7:0]	156
en	1
fij[7:0]	16
a[7:0]	156
b[7:0]	164
c[7:0]	170
d[7:0]	156
e[7:0]	161
f[7:0]	164
g[7:0]	156
h[7:0]	157
thima[7:0]	20
thimb[7:0]	25
thfma[7:0]	20
thfmb[7:0]	25
thsma[7:0]	15
thsmb[7:0]	20
n	1

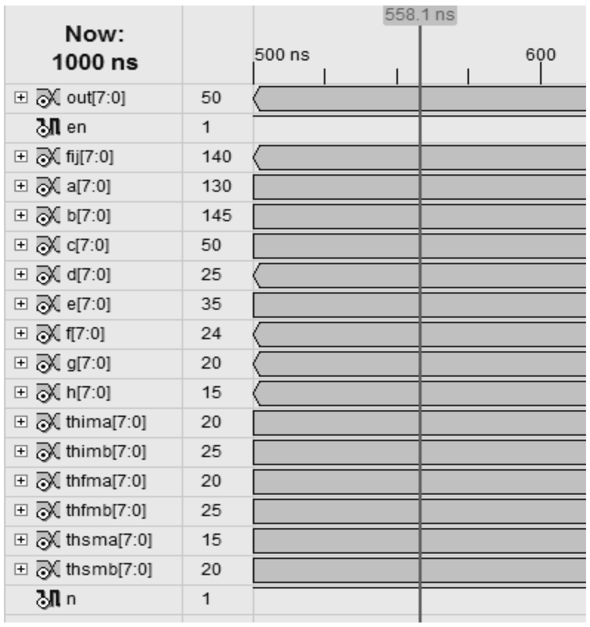
Figure 11. Simulation for noisy pixel in smooth region



Now: 1000 ns	
out[7:0]	40
en	1
fij[7:0]	40
a[7:0]	130
b[7:0]	145
c[7:0]	50
d[7:0]	125
e[7:0]	35
f[7:0]	30
g[7:0]	30
h[7:0]	45
thima[7:0]	20
thimb[7:0]	25
thfma[7:0]	20
thfmb[7:0]	25
thsma[7:0]	15
thsmb[7:0]	20
n	0

Figure 12. Simulation for pixel in edge region

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 1 No: 3 March 2016



```
Now:
1000 ns
out[7:0]     50
en           1
fij[7:0]     140
a[7:0]       130
b[7:0]       145
c[7:0]       50
d[7:0]       25
e[7:0]       35
f[7:0]       24
g[7:0]       20
h[7:0]       15
thima[7:0]   20
thimb[7:0]   25
thfma[7:0]   20
thfmb[7:0]   25
thsma[7:0]   15
thsmb[7:0]   20
n            1
```

Figure 13. Simulation for noisy pixel in edge region

The synthesis report is shown in table 1. Here we have used virtex 4 family and the selected device is 4vfx12sf363-12.

Table 1. Synthesis Report

Logic utilization	Used	Available	Utilization
Number of Slices	855	5472	15
Number of Slice Flip Flops	32	10944	0
Number of 4 input LUTs	1549	10944	14
Number of bonded IOBs	97	240	40
Number of GCLKs	1	32	3

9. FUTURE WORK

Till now we have implemented only for a (3X3) portion of an image. Our future work is to extend this to implement for (256X256) grey scale image.

10. CONCLUSION

In this proposed work, the detection and removal of impulse noise is carried out using various modules. As this method uses only fewer amounts of resources as shown in Table I it is more efficientthan previously reported works [3,4,5,6,7] and the quality of the reconstructed pixels are highly improved. Hence, theproposed method is suitablefor real time applications**.**

REFERENCES

[1] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Pearson Education, 2007.

[2] Samir Palnitkar, Verilog HDL (A guide to Digital Design and Synthesis), SunSoft Press, 1996.

[3] S. Deivalakshmi and P. Palanisamy, “Improved Tolerance Based Selective Arithmetic Mean Filter for Detection and Removal of Impulse Noise,” IEEE Aug. 2010.

[4] Sun Qiaoping, “A Geometric Mean Based Adaptive Local Noise Removal Algorithm,” IEEE Nov. 2005.

[5] F. Ahmed and S. Das, “Removal of High Density Salt and Pepper Noise in Images With an Iterative Adaptive Fuzzy Filter Using Alpha Trimmed Mean,” IEEE Oct. 2014.

[6] I. Aizenberg and C. Butakoff, “Effective Impulse Detector Based on Rank-Order Criteria,” IEEE Mar. 2004.

[7] S. J. Ko and Y. H. Lee, “Center Weighted Median Filters and Their Applications to Image Enhancement,” IEEE Sep. 1991.