A MODIFIED DECOMPOSITION FILTER FOR REMOVAL OF IMAGE NOISES AND ITS FPGA IMPLEMENTATION

Vasanth K[1], Karthik S[2]

[1]Sathyabama University, Chennai, Tamilnadu, India.
[2]Cognizant Technology solutions, Chennai, Tamilnadu, India
[1]vasanthece_k@yahoo.co.in
[2]skarthick76@gmail.com

ABSTRACT

The paper gives a new scheme for the restoration of images corrupted by Gaussian noise, impulse noise, speckle noise and mixed noise. The new algorithm works well on different noises and produce better image restoration than existing standard median filter (SMF), Centre weighted median filter (CWF) and threshold decomposition filter (TDF).The proposed algorithm (PA) is tested on different images corrupted by mixed noises and is found to produce better results in terms of the qualitative and quantitative measures of the image for noise densities up to 30% noise level for impulse noise, mean zero and 0.9% variance of Gaussian noise. The filter works well for speckle noise up to 0.8% variance. The proposed algorithm is targeted on Xc3e5000-5fg900 FPGA using Xilinx 7.1 compiler version which requires 3689 slices and a low power of 100mw and a optimum speed when compared to the other median finding architectures.

Keywords: Rank order filter, threshold decomposition, Modified decomposition filter, field programmable gate array.

I. INTRODUCTION

Images are often degraded by noises due to imperfections in image sensors or poor transmission medium. The different types of degradations that occur in images are additive random noise such as Gaussian white noise and salt-and-pepper impulse noise, signal-dependent noise such as speckle[4]. To restore the degraded images, a correct filter should be used. The characteristic of a good noise removal filter would exactly remove the noise distributions, thereby the original image is restored from the noisy image completely. In order to facilitate this, the chosen filtering algorithm must be stated to remove a particular noise distribution. In real time, Even the noise removal filter is designed to restore images well, there will be always some degree of variation in the restored pixel values from the original image. If the restoration of image pixels is deviated in excess when compared to original pixel then the restored pixel is not useful for processing. In the above stated condition the restored image might not be visually unacceptable if subjected to human inspection [5]. Due to the poor photo electronic detectors which results in thermal noise which is additive zero mean Gaussian noise is induced in the image [6]. A linear filter eliminates the Gaussian noise effectively. Impulse noise is caused by defect in pixels of camera sensors, fault in memory locations in hardware, or transmission in a noisy channel. Two common types of impulse noise are the salt-and-pepper noise and the random-valued noise. For images corrupted by salt-and pepper noise, the noisy pixels can take only the peak and the valley values while in the case of random-valued noise; they can take any random value in the dynamic range[6]. Speckle is a random, deterministic, interference pattern in an image formed with coherent radiation of a medium containing many sub-resolution scatterers. The local brightness of the speckle pattern, however, does reflect the local echogenicity of the underlying scatterers [5]. There are two basic approaches to image de-noising, spatial filtering methods and transform domain filtering methods [7]. A conventional method to remove noise from image data is to use a spatial filter. Spatial filters broadly classified into non-linear and linear filters. Many non-linear filters fall into the category of order statistic neighbor operators [2]. This means that the local neighbors are ordered in ascending order and this list is processed to give an estimate of the underlying image brightness. The simplest order statistic operator is the median [6], where the central value in the ordered list is used for the new value of the brightness. The median is good at reducing impulse noise

However, A mean or average filter is the optimal linear filter for Gaussian noise removal which tends to blur sharp edges, destroy lines and other fine image details, and perform poorly in the presence of signal-dependent noise. This paper is organized as follows. Section II describes noise model. Section III gives a overview of related work on Image De-noising using proposed algorithm and its hardware implementation. Section IV deals with Exhaustive Experimental Results and Discussions and finally Concluding Remarks are given in Section V.

II. NOISE MODEL

Let the true image x belong to a proper function space $S(\Omega)$ on $\Omega = [0\,;1]^2$, and the observed digital image y be a vector in Rmxm indexed by $A = 1, 2, \ldots m \times 1, 2, \ldots m$. The image degradation can be modeled as $y = N(Hx)$, where $H: S(\Omega) \rightarrow$ Rmxm is a linear operator representing blurring, and $N: R$ mxm Rmxm models the noise. Usually,

$y = Hx + \sigma n$ where $\sigma n \varepsilon$ Rmxm is an additive zero-mean Gaussian noise with standard deviation $\sigma >= 0$. outliers is modeled as impulse noise. [8].

$$y' = Hx + \sigma g \qquad \ldots\ldots\ldots (1)$$

$$y = N(y') \qquad \ldots\ldots\ldots (2)$$

where N represents the impulse noise. There are two common models for impulse noise: the salt-and-pepper noise and the random-valued noise. If [$d$min; $d$max] denote the dynamic range of y', i.e., $d$min $\leq y'_{ij}$ dmax for all (i,j), then they are denoted by Salt-and-pepper noise: the gray level of y at pixel location $(i\ j)$ is

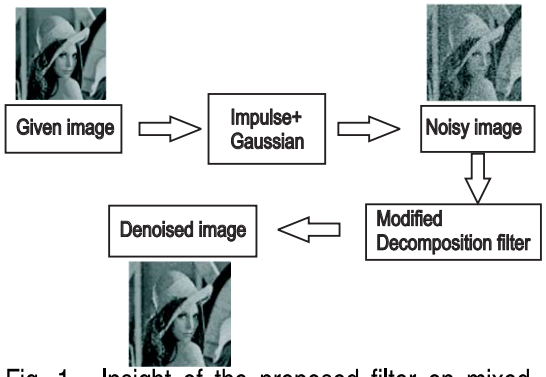


Fig. 1. Insight of the proposed filter on mixed noises

$$y_{ij} = \begin{cases} dmin\,; \text{ with probability } p\,; \\ dmax\,; \text{ with probaility } q\,; \\ y'_{ij}\,; \text{ with probability } 1-p-q\,; \end{cases}$$

Where $s = p + q$ denotes the salt-and-pepper noise level [8].

III. OVERVIEW OF PROPOSED ALGORITHM

The aim of the work is to remove the corrupted image (mixed noises i.e., zero mean Gaussian and impulse noise) using the proposed filter hardware. Figure 1 gives the insight of the proposed work. To overcome the problem in threshold decomposition algorithm, a new bit level decomposition algorithm is proposed. In the proposed scheme, the pixel intensity is decomposed into its equivalent string of 1s by keeping pixel intensities as threshold, thereby reducing the complexities that are encountered in the existing threshold decomposition method. The median is found easily by eliminating the process of finding median using the majority function which in turn eliminates the need for large comparisons. Proposed algorithm is given as follows:

Consider a given image of size MXN corrupted by mixed noise. Pad the edges of the image by 150.The value used for padding is obtained from extensive use of the value in more than 40 images of different type and stature and found to have better results. Hence the value is chosen to pad the edges of our algorithm.

STEP 1: A 2D window of size 3×3 is selected from the corrupted image to be processed. Assume the pixel to be processed is $p(x, y)$.

STEP 2: Every pixel of the window is decomposed into its number equivalent strings of 1's considering the pixel intensity itself as the threshold. Here the decomposition is done with the help of a counter ROW1, which eliminates the comparison involved in decomposition process of the conventional and existing threshold decomposition techniques. Simultaneously, the number of 1's in each column is counted with the help of a counter and its number equivalent is stored in COL1 simultaneously.

STEP 3: The values of COL1 counter are decomposed into its equivalent strings of 1's and the number of 1's at each column is recombined to obtain the pixel intensities of the window sorted in descending

order with the help of counter VAL. The fifth element of the VAL or the number equivalent of the fifth column counter gives the median of the window considered. After the computation of median, the centre pixel of the window is replaced by the evaluated median. Subsequently, the window moves towards the right for a new set of window values; this processing as well as the updating procedure are repeated until the end of the image element is reached. Fig 2 denotes the methodology of proposed algorithm [1]-[2].

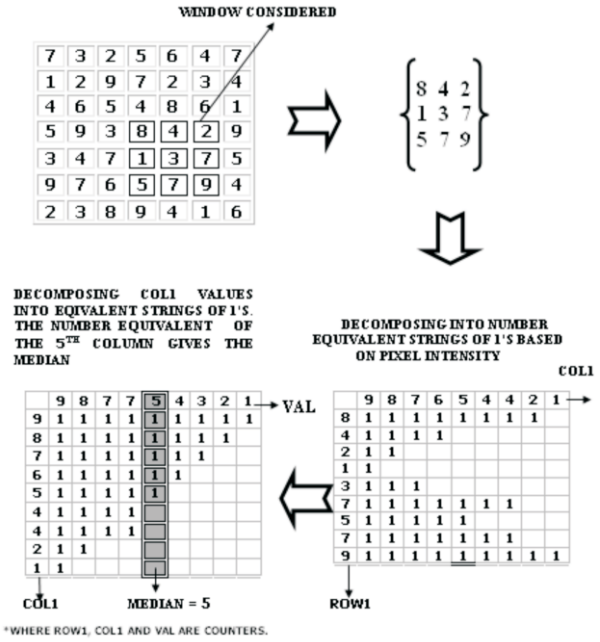


Fig. 2. Methodology of the proposed algorithm

A. Hardware implementation

The proposed algorithm works on two phase's decomposition and regrouping stage. Both these phases involve counters to decompose and to regroup the pixels. Hence a novel decomposition counter is proposed in this paper as shown in figure 3. The decomposition counter consists of a comparator, multiplexer, 2D buffer array x (i,j)

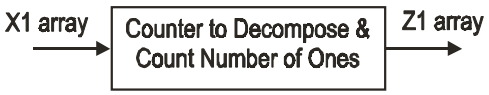


Fig. 3. Entity of Decomposition counter

A column counter array of size 255 named Z1 is initialized as zero. Initialize a 2D arrays named x1(i, j) and x2(i, j) where i (row index) ranges from 0 to 8 and j (column index) ranges from 0 to 255. In decomposition phase, decomposition is done by comparing the pixel intensity with column counter indices (j). If the index (j) value is greater than pixel value then assign zero to array x1, if the column index (j) is less than pixel intensity then assign one to array x1. Hence pixel values are decomposed. The number of 1s in each column is counted with the help of an adder adding ones in the array x1 and its number equivalent is stored z1 array which works as a counter which is illustrated in figure 4.

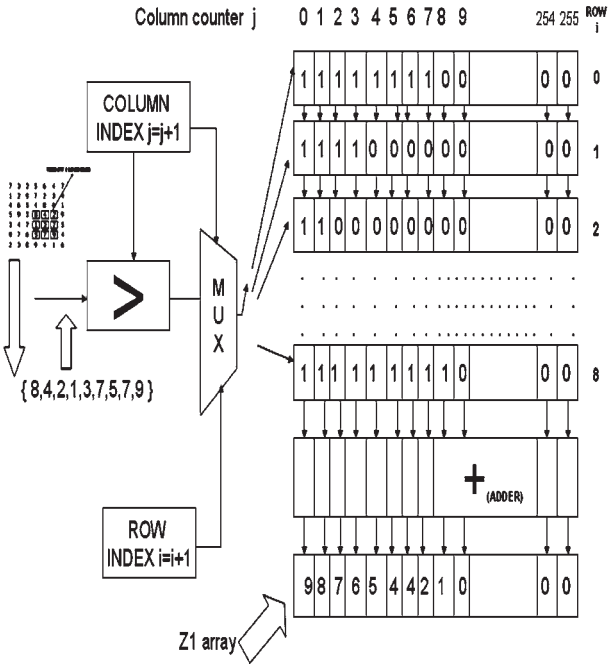


Fig.4 Illustration of Decomposition counter after decomposition phase

In recombination phase, the output array of first phase is considered as input to the next stage. The output of first array Z1 is decomposed into strings of ones comparing the pixel intensity with row counter indices (i). If the index (i) value is greater than pixel value then assign zero to array x1, if the column index (i) is less than pixel intensity then assign one to output array x2. Hence pixel values are decomposed. The number of 1s in each column is counted with the help of an adder adding ones in the array Z1 and its number equivalent is stored x2 array which works as a counter which is illustrated in figure 5. The resultant array is sorted in descending order. The process is repeated for the remaining 8 pixel values. Fifth element of the output array is the median [3].

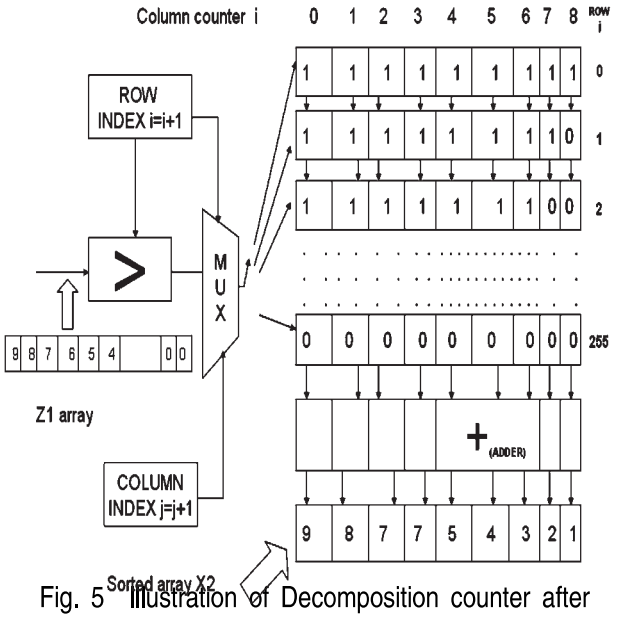


Fig. 5 Illustration of Decomposition counter after regrouping phase

IV. SIMULATION RESULTS

A. Proposed Algorithm Using MATLAB

This Section examines the performance of proposed image denoising algorithm with existing median filters, such as, standard median filter(inbuilt MATLAB function (3×3)) SMF, Center weighted median filter (CWF), Threshold decomposition filter (TDF), for noises such as salt and pepper noise, Speckle noise and zero mean Gaussian noise that are added on images such as Lena, Barbara, Baby, girl, Pepper and Cameraman image. It is proved experimentally that the proposed algorithm works optimally for better denoising of different noises. The quality of the restored image is found by computing Peak Signal to Noise Ratio (PSNR),Image enhancement factor(IEF) and time using (matlab inbuilt functions), which are the estimates of the quality of a filtered image compared with an original image. The PSNR is calculated using the formulae.

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{MSE}\right)$$

$$\mathrm{MSE} = \frac{\sum_{ij} (r_{ij} - x_{ij})\,2}{M\times N}$$

where r - Original image, MxN - size of image x - restored image. The Image enhancement factor is calculated using the formulae

$$\mathrm{IEF} = \frac{\left(\sum_{ij} n_{ij} - r_{ij}\right)^2}{\left(\sum_{ij} x_{ij} - r_{ij}\right)^2}$$

where n - corrupted image, r - original image, x - restored image [1].

The Quality measure such as PSNR, IEF, and CPU computation time in seconds for impulse noise, zero mean Gaussian noise and speckle for impulse are calculated for the PA and compared with SMF, CWF and TDF, in Tables 1 to 3 for pepper.bmp. The Proposed Algorithm uses a fixed 3X3 window for processing for increasing noise densities and thus leads to smaller computation time amongst the existing threshold decomposition filters or stack filters and centre weighted median filter. MATLAB 7.0(R14) on a PC equipped with 2-GHz CPU and 1GB of RAM memory has been employed for the evaluation of computation time of all algorithms. It was found from table1-3 that the proposed algorithm has better performance in removing impulse noise up to 30%.From table 5 and 6 it was observed that the proposed algorithm has capability to eliminate zero mean with 0.9% variance Gaussian noise and speckle noise up to 0.8% variance. Considering the discussions made before, Subsequent Tables 4 to 6 represents the performance of the SMF, CWF, TDF and PA for five different images by above said compositions of noises respectively. Table 7 and 8 shows the performance of the PA is better in terms of PSNR, IEF and optimum time when compared with SMF, CWF, and TDF for various types of images corrupted by all three types of noises in proportion. Fig 6-14 illustrates the performance of the PA over other filters for impulse noise, Gaussian noise and speckle noise. In fig 15-16 PA has higher PSNR, IEF when tested on different images which is corrupted by 30% impulse noise. In fig 18, 19 PA has slightly better PSNR, IEF over other filters that are used for denoising zero mean variance 0.9% variance Gaussian noise tested on various images. It was observed that for the images which have gray levels varying more (details of an image) such as cameraman.bmp, barbera.tif, girl.jpg the PA performance is average when compared with other filters. For the images whose gray levels is uniform (details of the image) such as baby.jpg, pepper.bmp the performance of the PA is good when compared with other filters. In fig 21 the PSNR performance of

the PA is in par with other filters for 0.8% variance speckle noise. From fig 22 we understand such that depending upon the variation in grey levels in an image the performance is good or average. IEF of the PA good on par with other filters if the grey level changes are more else the performance is average. Fig 24-25 gives the performance of PA over different images corrupted by mixed noises in some proportion has a good PSNR and IEF. Fig 50-53 shows pictorial representation obtained by employing various filters. Fig 8, 11, 14, 17, 20, 23, 26 denotes the optimum computational speed at which the PA works[2].

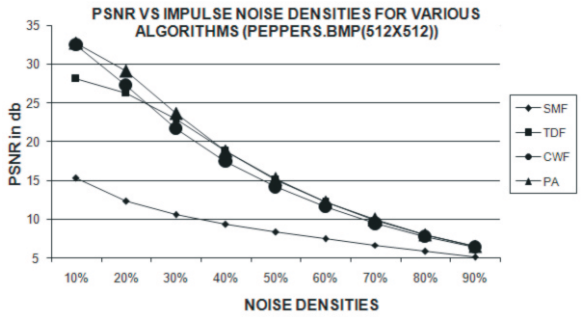


Fig. 6. Noise density versus PSNR for various filters for pepper image corrupted by impulse noise

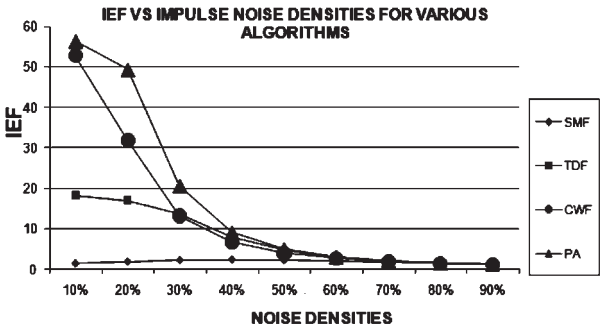


Fig. 7. Noise density versus IEF for various filters for pepper image corrupted by impulse noise

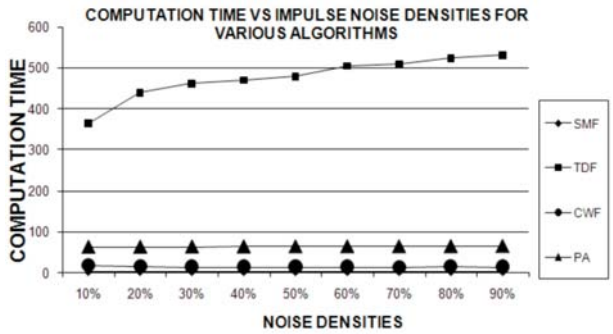


Fig. 8. Noise density versus TIME for various filters for pepper image corrupted by impulse noise

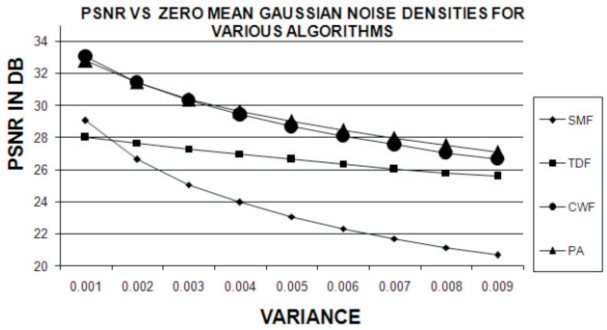


Fig. 9. Variance versus PSNR for various filters for pepper image corrupted by Gaussian noise

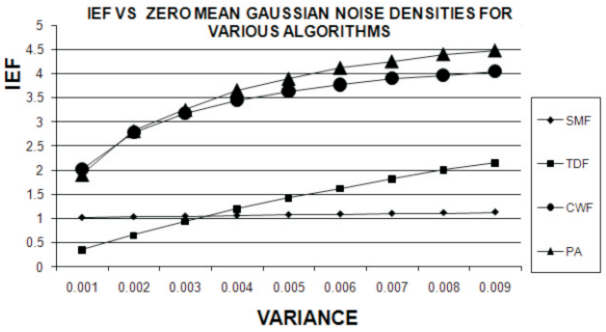


Fig. 10. Variance versus IEF for various filters for pepper image corrupted by Gaussian noise

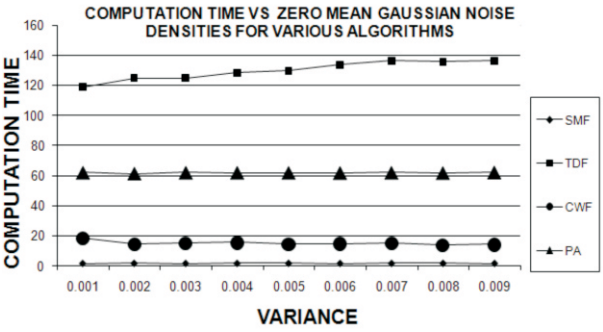


Fig. 11. Variance versus TIME for various filters for pepper image corrupted by Gaussian noise

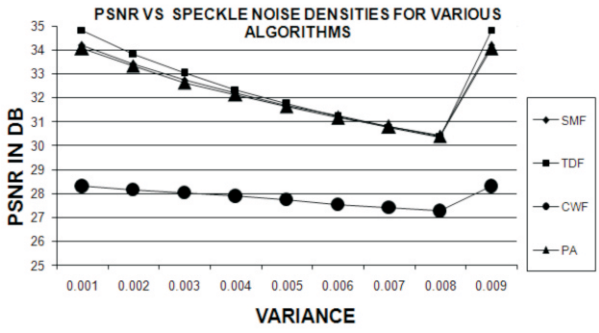


Fig. 12. Variance versus PSNR for various filters for pepper image corrupted by Speckle noise

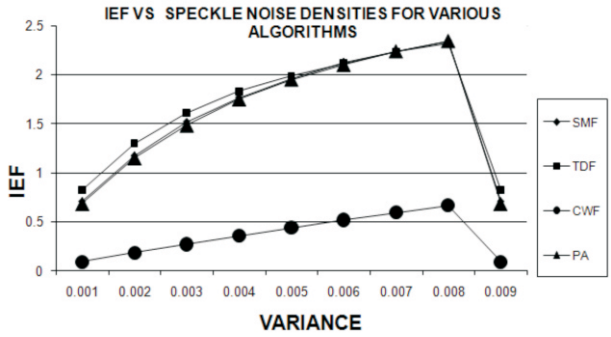


Fig. 13. Variance versus IEF for various filters for pepper image corrupted by Speckle noise

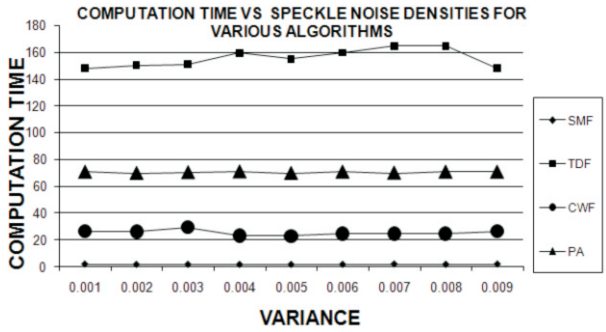


Fig. 14. Variance versus TIME for various filters for pepper image corrupted by Speckle noise

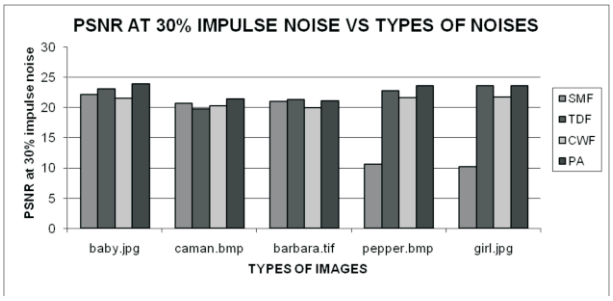


Fig. 15. PSNR for various filters applied over different images corrupted by 30% impulse noise

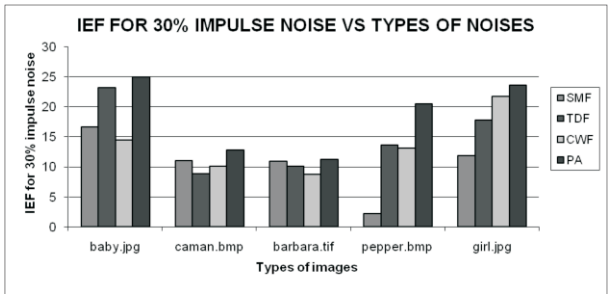


Fig. 16. IEF for various filters applied over different images corrupted by 30% impulse noise

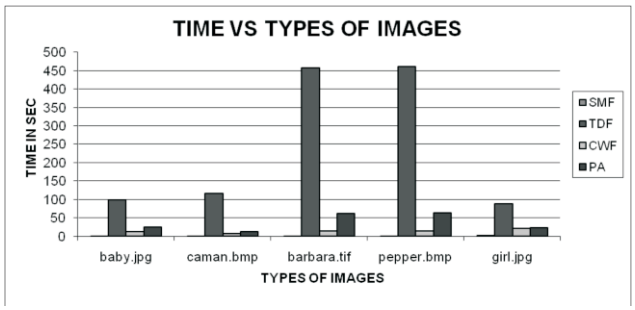


Fig. 17. TIME for various filters applied over different images corrupted by 30% impulse noise

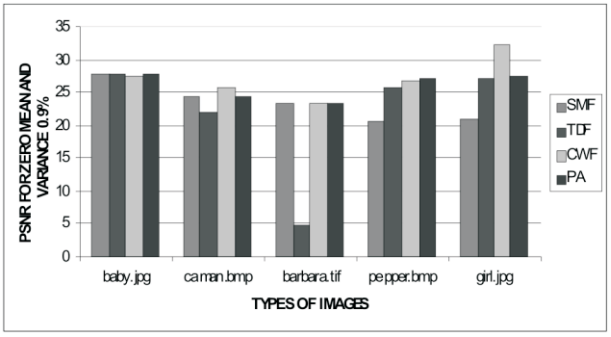


Fig. 18. PSNR for various filters applied over different images corrupted by zero mean and 0.9% variance Gaussian noise

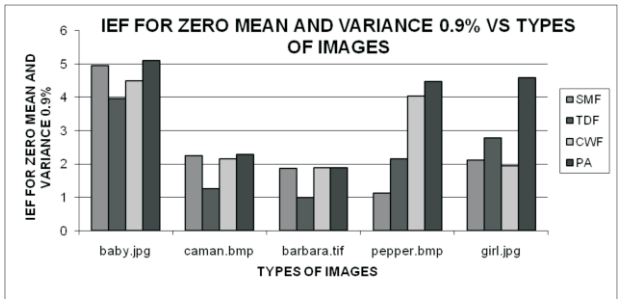


Fig. 19. IEF for various filters applied over different images corrupted by zero mean and 0.9% variance Gaussian noise

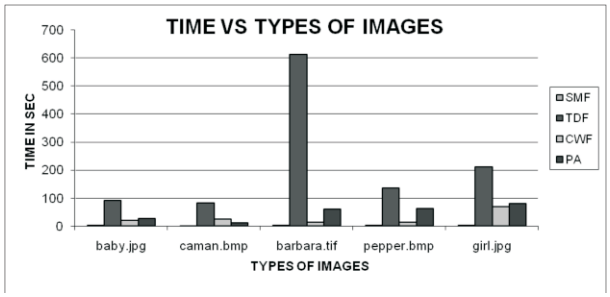


Fig. 20. TIME for various filters applied over different images corrupted by zero mean and 0.9% variance Gaussian noise

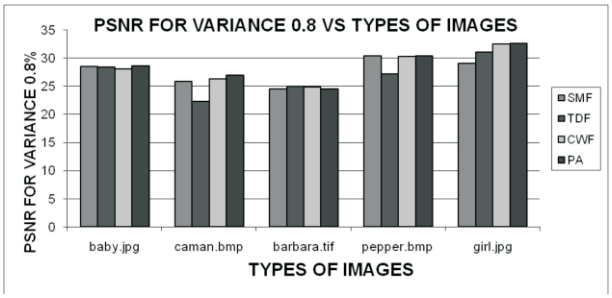


Fig. 21. PSNR for various filters applied over different images corrupted by 0.8% variance Speckle noise

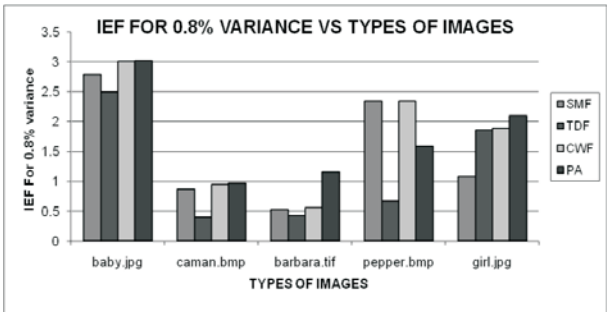


Fig. 22. IEF for various filters applied over different images corrupted by 0.8% variance Speckle noise

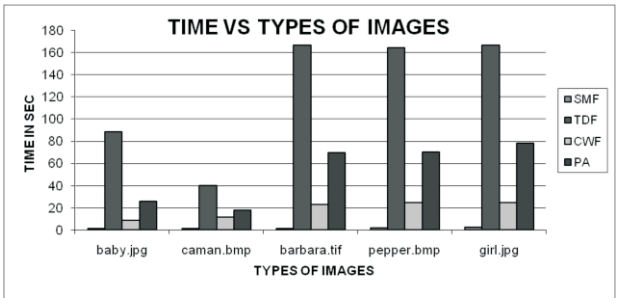


Fig. 23. TIME for various filters applied over different images corrupted by 0.8% variance Speckle noise

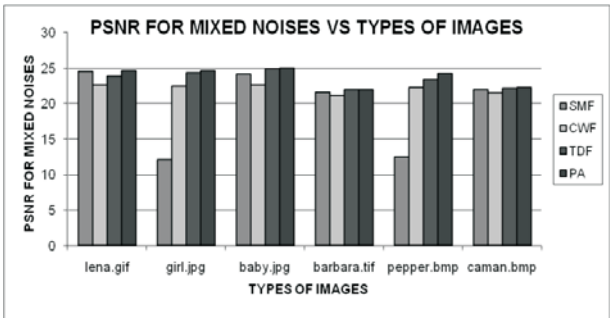


Fig. 24. PSNR for various filters applied over different images corrupted by 20% impulse noise, 0.9%variance Gaussian noise

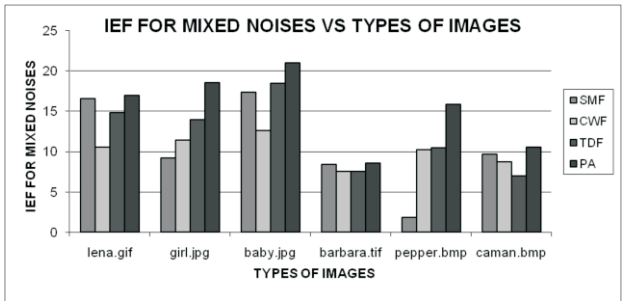


Fig. 25. IEF for various filters applied over different images corrupted by 20% impulse noise, 0.9%variance Gaussian noise

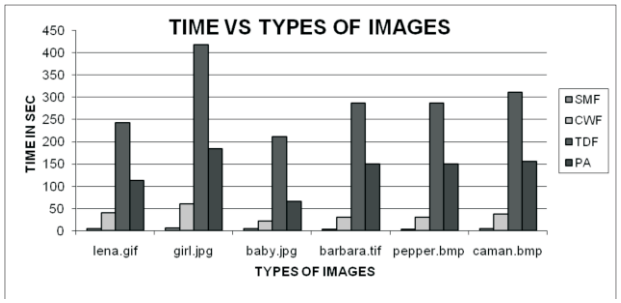


Fig. 26. TIME for various filters applied over different images corrupted by 20% impulse noise,0.9%variance Gaussian noise

B. Proposed Algorithm for FPGA

The proposed algorithm was targeted on Spartan 3e family XC3S5000-5fg900 FPGA. The code was developed using VHDL. The simulator tools used was a third party tool Modelsim 5.8i and synthesis tool XST was used as part of Xilinx 7.1i suit for CPLD & FPGA development. Table 9 gives the device utilization summary, timing specification and power report for the target FPGA for various median finding algorithms such as bubble sort, heap sort, insertion sort, Selection sort, Threshold decomposition Filter [3].

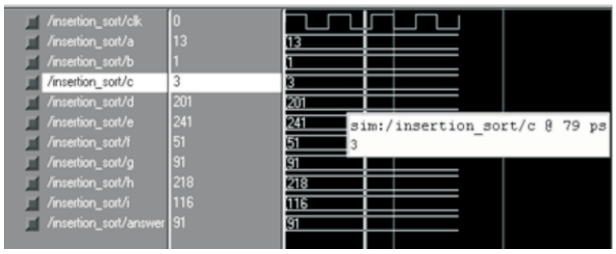


Fig. 27 Simulation result of Insertion sorting

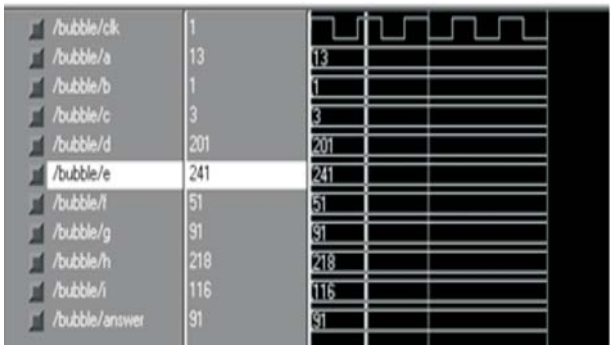


Fig. 28 Simulation result of bubble sorting

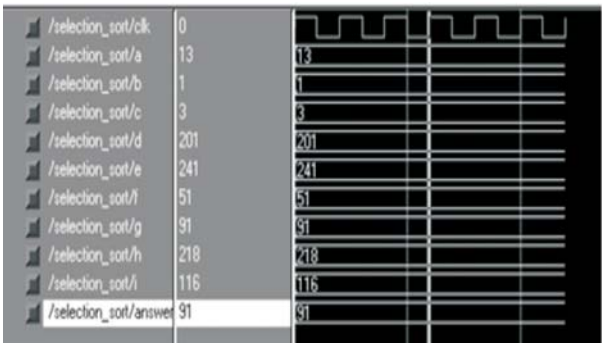


Fig. 29 Simulation result of Selection sorting

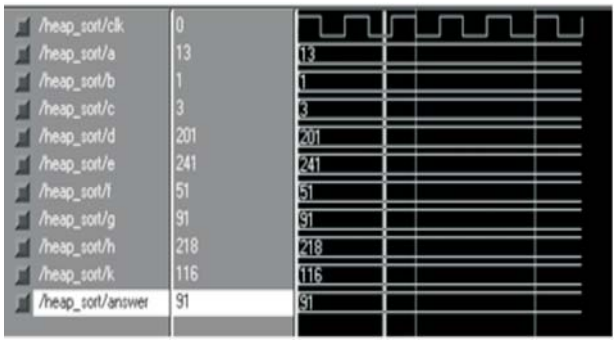


Fig. 30 Simulation result of Heap sorting

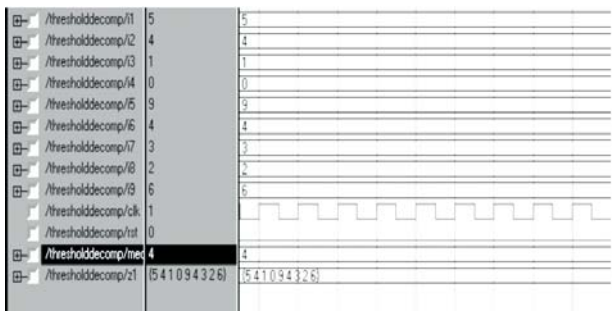


Fig. 31 Simulation result of TDF

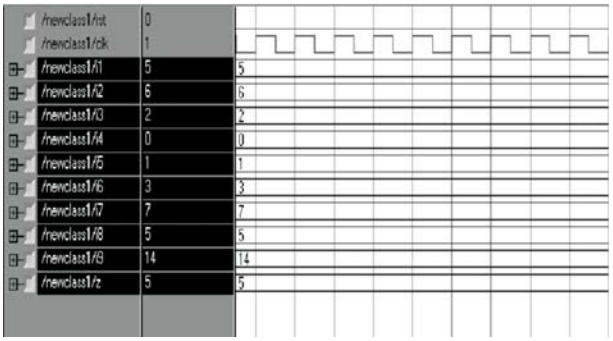


Fig. 32 Simulation result of PA

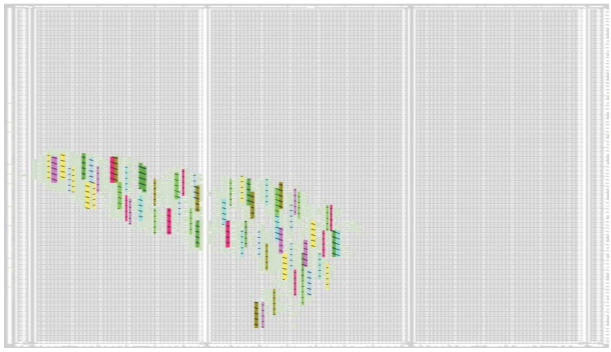


Fig. 33 Floor plan of Bubble Sorting

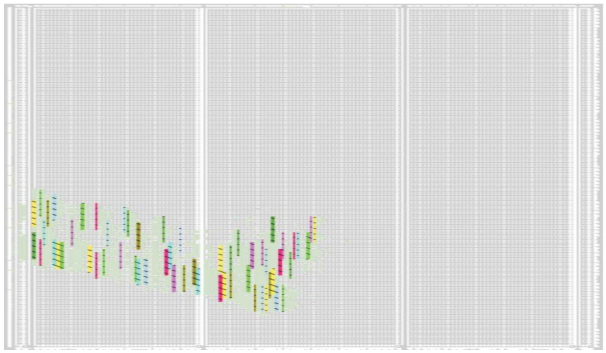


Fig. 34 Floor plan of Heap Sorting

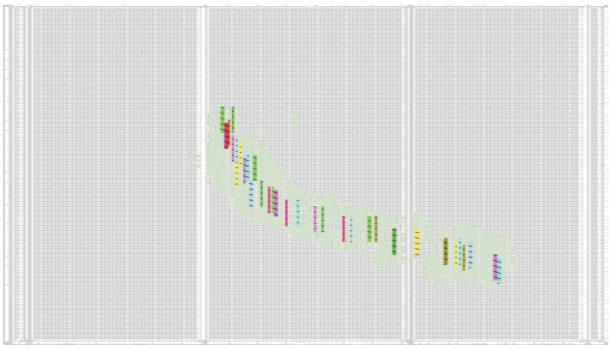


Fig. 35 Floor plan of insertion Sorting

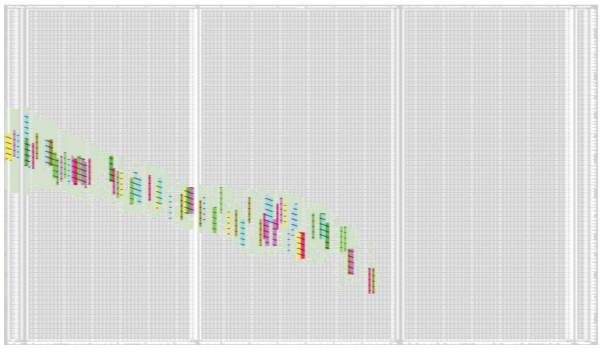


Fig. 36 Floor plan of Selection Sorting

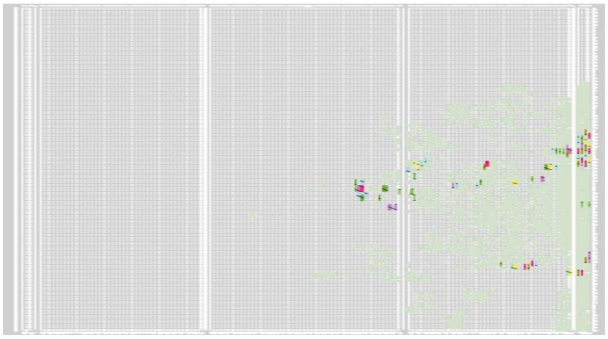


Fig. 37 Floor plan of TDF

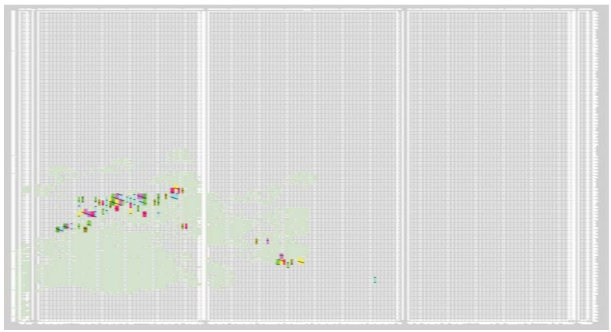


Fig. 38 Floor plan of PA

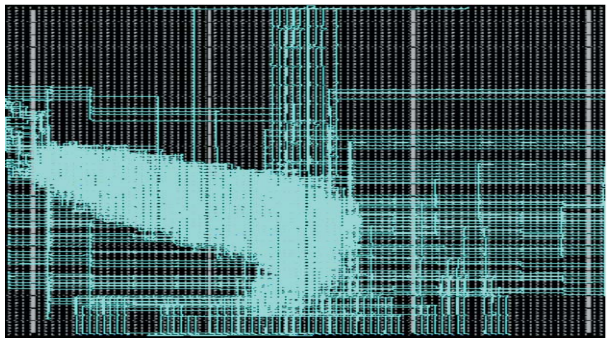


Fig. 39 Routed FPGA of Bubble Sorting

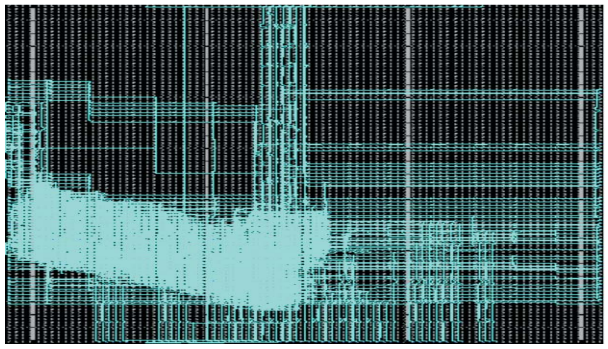


Fig. 40 Routed FPGA plan of Heap Sorting

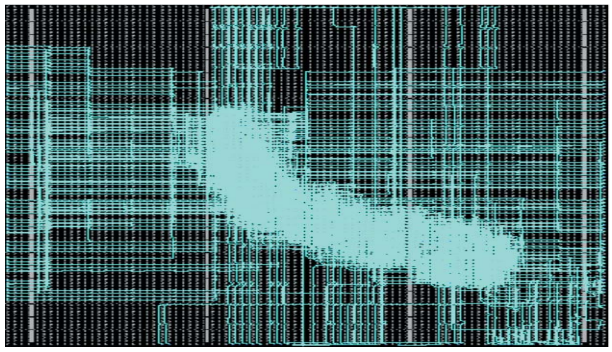


Fig. 41 Routed FPGA of insertion Sorting

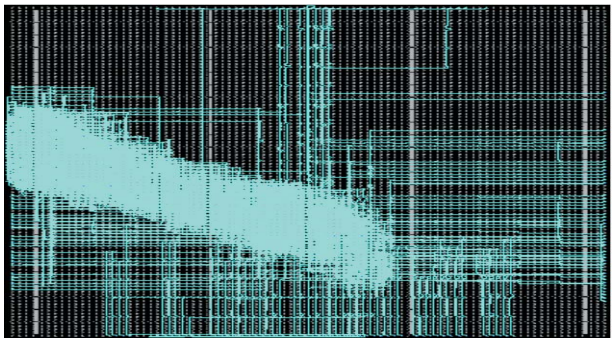


Fig. 42 Routed FPGA of Selection Sorting

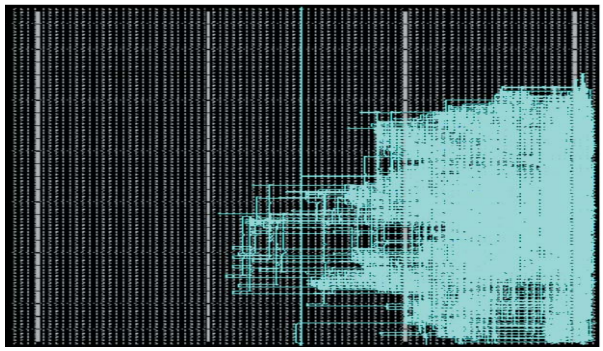


Fig. 43 Routed FPGA of TDF

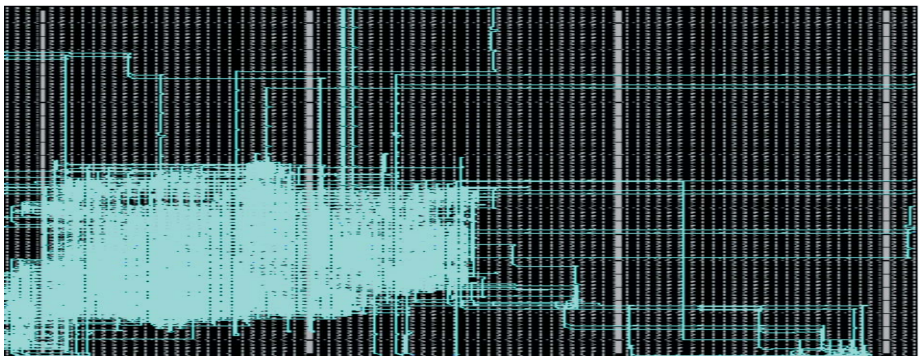


Fig. 44 Routed FPGA of TDF

Table 1 PSNR,IEF,TIME for PEPPER.BMP **(512 × 512)** IMAGE CORRUPTED BY IMPULSE NOISE AT DIFFERENT NOISE DENSITIES

	PSNR				IEF				TIME			
ND	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
10%	15.316	28.062	32.483	32.763	1.452	18.237	52.712	56.325	1.694	363.787	17.384	62.711
20%	12.324	26.279	27.267	29.112	1.891	17.049	31.779	49.28	1.759	439.819	15.085	63.22
30%	10.588	22.864	21.651	23.667	2.22	13.634	13.133	20.539	1.777	461.69	14.136	63.651
40%	9.341	18.741	17.456	18.789	2.363	7.978	6.659	9.039	1.777	470.404	14.286	63.933
50%	8.313	15.085	14.159	15.171	2.223	4.609	3.872	4.868	1.732	479.944	14.231	64.024
60%	7.446	12.181	11.6	12.224	1.936	2.871	2.56	2.967	1.777	505.359	14.258	64.127
70%	6.651	9.892	9.422	9.931	1.622	1.993	1.818	2.032	1.758	509.625	13.638	64.422
80%	5.867	8.024	7.762	8.013	1.348	1.493	1.413	1.494	1.722	524.815	15.723	64.574
90%	5.147	6.499	6.392	6.531	1.143	1.184	1.159	1.193	1.707	532.148	14.126	64.865

Table 2 PSNR,IEF,TIME for PEPPER.BMP **(512 × 512)** IMAGE CORRUPTED BY ZERO MEAN GAUSSIAN NOISE AT DIFFERENT NOISE DENSITIES

VAR	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
0.001	29.049	28.022	33.052	32.771	1.012	0.343	2.016	1.901	1.646	118.857	18.336	61.927
0.002	26.62	27.669	31.452	31.461	1.027	0.649	2.786	2.805	1.694	124.801	14.567	61.287
0.003	25.052	27.291	30.305	30.359	1.042	0.934	3.171	3.243	1.656	124.899	15.203	61.864
0.004	23.967	26.981	29.423	29.651	1.057	1.188	3.448	3.653	1.68	128.467	15.593	61.816
0.005	23.075	26.644	28.685	28.988	1.072	1.417	3.635	3.886	1.71	129.563	14.657	61.807
0.006	22.315	26.365	28.098	28.429	1.085	1.616	3.764	4.113	1.645	133.651	14.879	61.699
0.007	21.7	26.009	27.557	27.924	1.099	1.81	3.897	4.246	1.683	136.235	14.965	61.856
0.008	21.135	25.794	27.052	27.507	1.114	1.993	3.964	4.393	1.678	135.62	13.964	61.649
0.009	20.657	25.566	26.669	27.065	1.128	2.148	4.043	4.472	1.656	136.334	14.281	62.035

Table 3 PSNR,IEF,TIME for PEPPER.BMP **(512 × 512)** IMAGE CORRUPTED BY SPECKLE NOISE AT DIFFERENT NOISE DENSITIES

	PSNR				IEF				TIME +			
VAR	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
0.001	34.183	34.833	28.294	34.054	0.704	0.818	0.094	0.679	1.812	148	26.344	70.485
0.002	33.402	33.835	28.151	33.326	1.171	1.296	0.186	1.145	1.562	150.438	25.985	69.594
0.003	32.763	33.021	28.021	32.633	1.517	1.608	0.273	1.479	1.594	151.157	29.235	70.11
0.004	32.19	32.352	27.884	32.139	1.766	1.827	0.358	1.747	1.563	159.343	23.328	70.719
0.005	31.655	31.738	27.735	31.635	1.963	1.988	0.439	1.948	1.563	154.937	22.969	69.547
0.006	31.235	31.199	27.544	31.164	2.13	2.114	0.516	2.097	1.578	159.734	24.625	70.578
0.007	30.787	30.769	27.411	30.781	2.244	2.23	0.592	2.236	1.859	164.343	24.797	69.797
0.008	30.407	30.342	27.271	30.396	2.345	2.316	0.664	2.341	1.593	164.657	24.719	70.515
0.009	34.183	34.833	28.294	34.054	0.704	0.818	0.094	0.679	1.812	148	26.344	70.485

Table 4 PSNR,IEF,TIME for DIFFERENT IMAGES CORRUPTED BY IMPULSE NOISE AT 30% NOISE DENSITY

IMAGES	PSNR				IEF				TIME			
	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
BABY.JPG (292 × 425)	22.172	23.076	21.591	23.973	16.694	23.199	14.524	24.995	1.335	98.674	11.75	24.736
CAMERAMAN.BMP (256 × 256)	20.698	19.826	20.352	21.418	11.022	8.875	10.135	12.821	0.995	116.883	7.178	11.66
BARBERA.TIF (512 × 512)	21.038	21.327	20.041	21.147	10.917	10.075	8.722	11.239	1.38	457.713	14.24	62.22
PEPPER.BMP (512 × 512)	10.588	22.864	21.651	23.667	2.22	13.634	13.133	20.539	1.777	461.69	14.136	63.651
GIRL.JPG (600 × 900)	10.232	23.65	21.753	23.613	11.907	17.817	21.753	23.614	1.918	87.308	21.753	23.65

Table 5 PSNR,IEF,TIME for DIFFERENT IMAGES CORRUPTED BY ZERO MEAN GAUSSIAN NOISE FOR VARIANCE 0. 9

IMAGES	PSNR				IEF				TIME			
	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
BABY.JPG (292 × 425)	27.744	27.821	27.34	27.917	4.952	3.964	4.489	5.111	1.354	91.194	19.62	27.116
CAMERAMAN.BMP (256 × 256)	24.361	21.778	25.651	24.427	2.246	1.266	2.15	2.292	0.017	82.554	25.199	11.245
BARBERA.TIF (512 × 512)	23.246	4.642	23.316	23.287	1.875	0.988	1.892	1.892	1.472	614.141	14.045	60.599
PEPPER.BMP (512 × 512)	20.657	25.566	26.669	27.065	1.128	2.148	4.043	4.472	1.656	136.334	14.281	62.035
GIRL.JPG (600 × 900)	20.839	26.973	32.285	27.366	2.116	2.778	1.939	4.584	1.782	211.644	69.512	79.246

Table 6 PSNR,IEF,TIME for DIFFERENT IMAGES CORRUPTED BY SPECKLE NOISE FOR VARIANCE 0.8%

IMAGES	PSNR				IEF				TIME			
	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA	SMF	TDF	CWF	PA
BABY.JPG (292 × 425)	28.53	28.446	28.153	28.623	2.784	2.504	3.007	3.021	1.346	88.697	8.813	25.755
CAMERAMAN.BMP (256 × 256)	25.934	22.345	26.308	26.979	0.864	0.4	0.943	0.874	0.969	40.015	11.5	17.61
BARBERA.TIF (512 × 512)	24.523	24.969	24.842	24.563	0.52	0.42	0.558	0.522	1.157	166.657	23.062	69.656
PEPPER.BMP (512 × 512)	30.407	27.271	30.342	30.396	2.345	0.664	2.316	2.341	1.593	164.657	24.719	70.515
GIRL.JPG (600 × 900)	29.147	31.125	32.607	32.689	1.082	1.859	0.153	1.893	2.095	166.673	24.453	78.261

Table 7 PSNR, IEF, TIME for LENA.GIF, GIRL.JPG, and BABY.JPG IMAGES CORRUPTED BY 20% IMPULSE NOISE PLUS ZERO MEAN 0.9% VARIANCE GAUSSIAN NOISE

	LENA.GIF (512 × 512)			GIRL.JPG (600 × 900)			BABY.JPG (292 × 425)		
	PSNR	IEF	TIME	PSNR	IEF	TIME	PSNR	IEF	TIME
SMF	24.599	16.596	4.14	12.14	9.238	5.056	24.145	17.386	3.699
CWF	22.631	10.569	39.798	22.51	11.413	60.134	22.676	12.6072	21.484
TDF	23.946	14.854	242.255	24.42	13.971	419.09	24.92	18.48	211.781
PA	24.704	17.015	113.295	24.643	18.591	183.885	25.074	20.999	66.235

Table 8 PSNR, IEF, TIME for BARBARA.TIF,PEPPER.BMP, CAMERAMAN.BMP IMAGES CORRUPTED BY 20% IMPULSE NOISE PLUS ZERO MEAN 0.9% VARIANCE GAUSSIAN NOISE

	BARBARA.TIF (512 × 512)			PEPPER. BMP (512 × 512)			CAMERAMAN.BMP (256 × 256)		
	PSNR	IEF	TIME	PSNR	IEF	TIME	PSNR	IEF	TIME
SMF	21.593	8.399	3.378	12.515	1.812	3.378	22.023	9.693	4.169
CWF	21.134	7.509	29.632	22.289	10.213	29.632	21.514	8.754	36.738
TDF	21.962	7.547	286.573	23.46	10.437	286.573	22.173	7.005	311.423
PA	21.987	8.565	150.454	24.244	16.011	150.454	22.274	10.59	155.469

Table 9 Comparison of various median finding algorithms for XC3s5000-5FG900

No.	Parameters	IDF	PA	Bubble	Heap	Selection	Insertion
Device Utilization Factor After Synthesis							
1	Slices	4132	4021	4375	3810	4552	3399
2	4 i/p LUT	7066	6854	6080	5312	6368	5340
3	Bonded IOB	82	82	321	321	321	321
4	Gclk	2	1	1	1	1	1
Device Utilization Fator After place & Route							
1	Externai IOBs	82	82	321	321	321	321
2	Slices	3800	3687	3088	2783	3280	2810
Timing Specifications before Place & Route							
1	Minimum Input arrival before the clock ns	4.05	169.13	158.88	334.72	417.41	223.69
2	Maximum output required time after clock ns	6.21	6.21	7.165	7.165	7.165	7.165
Power consumption							
1	Power(mw)	298	100	298	298	298	298

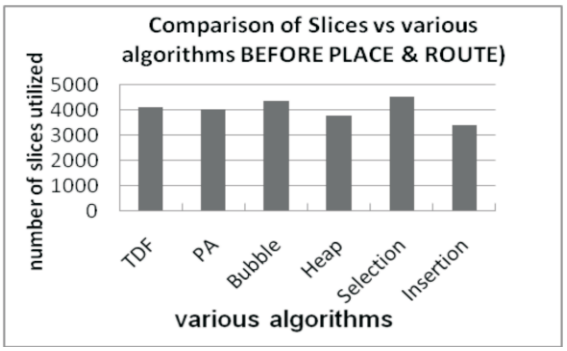


Fig. 45. Comparison of Slices utilized Vs Various algorithms

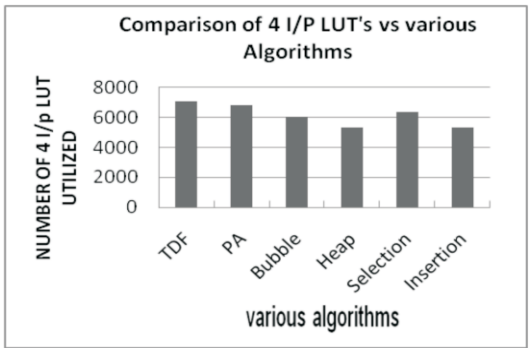


Fig. 46. Comparison of 4 i/p LUT's utilized Vs Various algorithms

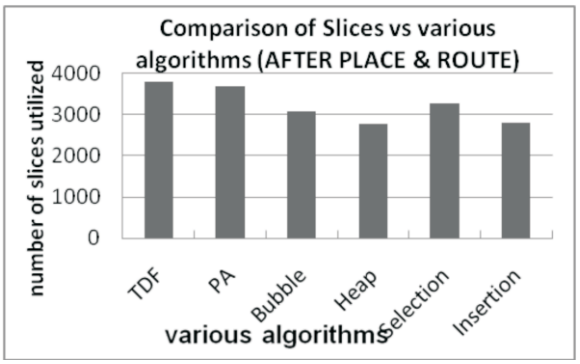


Fig. 47. Comparison of slices utilized Vs Various algorithms

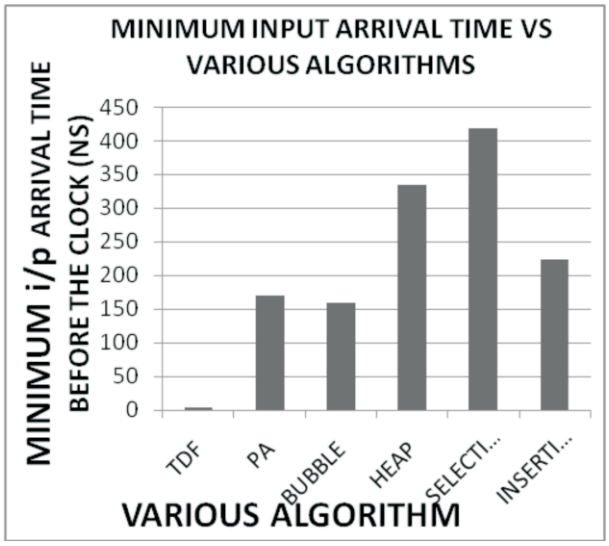


Fig. 48. Comparison of minimum input arrival Time Vs Various Algorithms

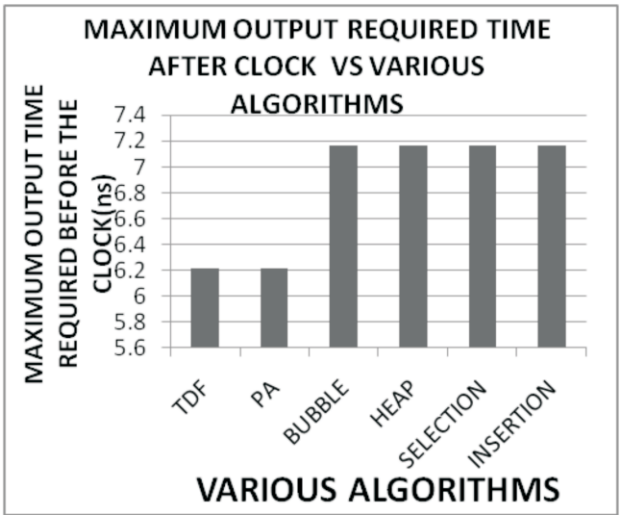


Fig. 49. Comparison of maximum output required Time after clock Vs Various Algorithms

(a) (b) (c) (d) (e) (f)

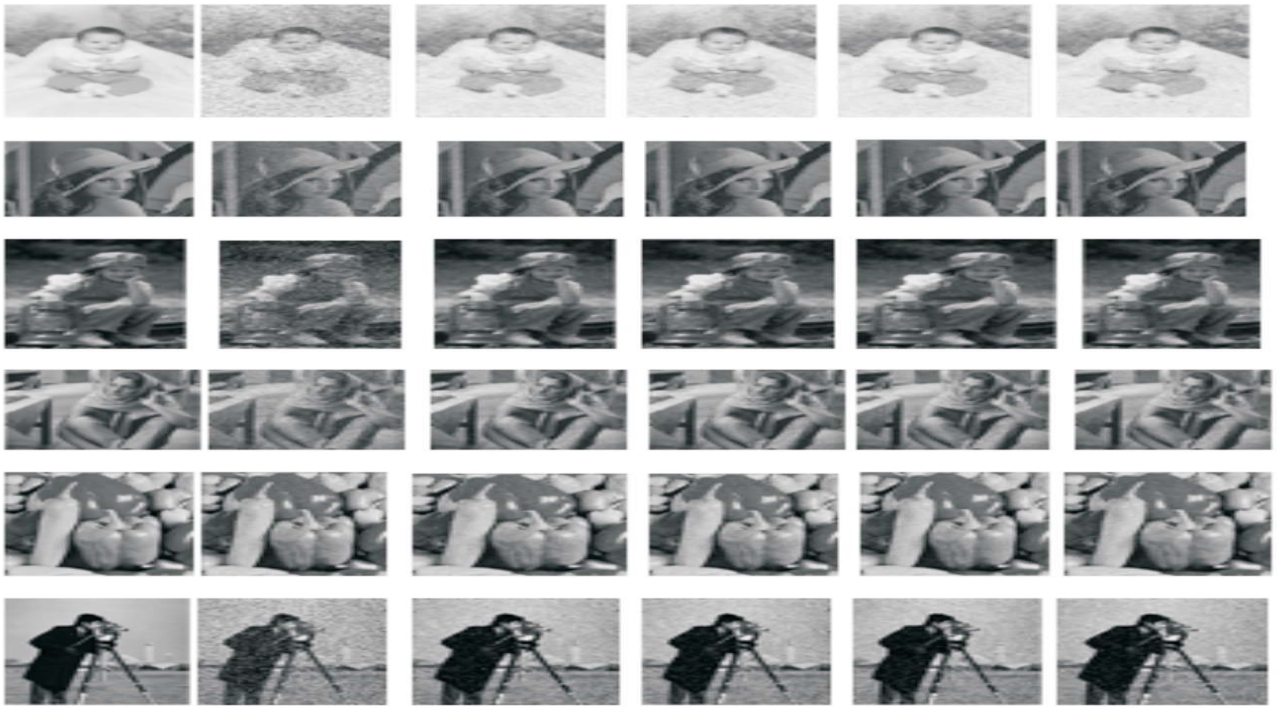
Fig. 50. Baby.jpg, pepper.bmp, girl.jpg (a) original image (b) impulse noise affected from by 30% (c) images restored by SMF (d) images restored from by TDF (e) images restored by CWF (f) images restored by proposed algorithm

(a) (b) (c) (d) (e) (f)

Fig. 51. Baby.jpg, pepper.bmp, girl.jpg (a) original image (b) Zero mean and 0.9% variance Gaussian noise (c) images restored by SMF (d) images restored from by TDF (e) images restored by CWF (f) images restored by proposed algorithm

(a) (b) (c) (d) (e) (f)

Fig. 52. Baby.jpg, pepper.bmp, girl.jpg (a) original image (b) 0.8% variance Speckle noise (c) images restored by SMF (d) images restored from by TDF (e) images restored by CWF (f) images restored by proposed algorithm



(a) (b) (c) (d) (e) (f)

Fig. 53. Barbara.tif, pepper.bmp, lena.gif, Cameraman.bmp, baby.jpg, girl.jpg (a) original image (b) Impulse noise 20% plus zero mean 0.9% variance Gaussian noise (c) images restored by SMF (d) images restored from by CWF (e) images restored by TDF (f) images restored by proposed algorithm

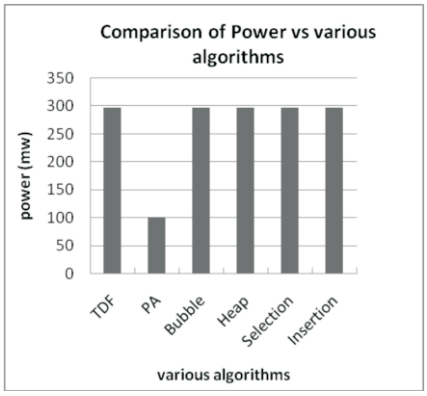


Fig. 54. Comparison of power Vs Various Algorithms

Figure 27-32 illustrates the simulation output of the existing median finding algorithm and proposed algorithm. Figure 33-38 eludes the floor plan for the existing algorithm and proposed algorithm. Figure 39-44 gives the routing of the logic with the pins for various existing median finding algorithm and proposed algorithm. Figure 45-49 and 54 gives the number of logic utilized before and after place and route, number of 4 input look up table utilized, Minimum input arrival time required before the clock and maximum output time required after the clock for various existing median filtering algorithm and proposed algorithm.

IV. CONCLUSION

The proposed algorithm and the existing median filters had been tested for different noise types for different images. It was assumed to keep a fixed 3x3 window for increasing noise densities for the proposed algorithm. From the extensive experiments, we conclude that, the highest PSNR (dB) and IEF is not obtained for PA for different images and for different noise type at higher noise densities. However, on an average sense, PA gives good performance for low density impulse noise up to 20%, zero mean 0.9% variance Gaussian noise removal. When compared to their class of decomposition filters such as TDF in specific, the PA exhibits better performance for Salt & Pepper noise removal up to 30% and reduces smaller proportion of zero mean 0.9% variance Gaussian noise. The proposed filter also exhibits good noise removal up to 0.8% speckle noise. In our method, time complexity of the existing methods is eliminated by using the pixel intensity itself as threshold. Hence, the proposed method shows optimum performance with fewer comparison complexities. The Proposed algorithm has good average computation time such that it's twice faster in comparison to TDF and exhibits optimum computation speed when compared with other filters. FPGA implementation part is concerned the simulation of each of the median finding algorithm along with the proposed algorithm is given in figure 27-32 The Proposed algorithm uses comparators, buffers, adders in comparison with large amount of complex comparators and adders used in existing bit level architecture such as TDF. It is evident visually that the Floor plan and routed FPGA that area occupied by MDF algorithm is optimum when compared with existing median finding algorithm. Table 9 gives the device utilization summary, timing and power specification for the target device XC3S5000-5fg900 required by the proposed algorithm in par with the existing median finding algorithm. Row 1 of the table 9 illustrates the slices required by the proposed logic after the synthesis. The proposed algorithm requires optimum number of slices when compared to the byte level median finding algorithms. But when compared with its class of bit level architecture it requires less number of slices. The second row of the table 9 denotes the number of 4 input look up table required by the proposed algorithm along with the existing algorithm. The proposed algorithm require optimum number of 4 input look up table when compared to existing bit level and byte level architectures which are discussed in this paper. The proposed logic requires less number of bonded IOB's. It was found both the bit level architecture require 89 bonded IOB's when compared to the other byte level architectures implemented and compared in this paper. The second half of the table deals with the device utilization of all the median finding algorithms after place and route. Once again it clear that the proposed logic requires optimum number of slices when compared with byte level algorithms such as bubble, insertion, selection, heap. It was found that the number of slices required is less when compared to the bit level architecture of TDF. The third part of the table deals with the minimum and maximum input and output required time before and after clock. It was found that the proposed logic requires optimum input and output arrival time. The last part of the table deals with power required by each median finding algorithm on the FPGA. It was found that the proposed logic requires very less power. It was also evident from the bar graphs in figure 45-49, 54 that the proposed logic requires optimum number of slices and optimum input and output arrival times when compared with byte level median finding algorithms and

less number of slices when compared with the existing algorithm. The proposed logic requires low power when compared with the existing median finding algorithm. The proposed algorithm exhibits very good results in restoration of images corrupted by non identical noise and an optimum area reduced, low power architecture is proposed for modified decomposition algorithm.

REFERENCES

[1] K.Vasanth,S.Karthik,"A New class of decomposition algorithm for the reduction of low density impulse noise", international conference on ARTCOM2009, kerala, India pages 203-207

[2] K.Vasanth, S.Karthik., "Performance Analysis of modified decomposition filter for non identical noises", ICTACT international journal on Image and video processing, 2010, Vol1, issue2, pages 105-115.

[3] K.Vasanth, S.Karthik., "FPGA implementation of modified decomposition filter", international conference on Signal and image processing, RMD Engg college, Tamilnadu, India,2010

[4] N. D. Sidiropoulos, J. S. Baras & C A Berenstein, 1994,"Optimal filtering of digital binary images corrupted by union/intersection noise", IEEE Trans. on Image Processing, Vol.3, No.4, July 1994, pp.382-403.

[5] G.R.Arce, N.C.Gallagher, and T.Nodes, 1986, "Median filters: Theory and applications," in Advances in Computer Vision and Image Processing, T.Huang, Ed.Greenwich, CT: JAI.

[6] I. Pitas and A.N.Venestanopoulos, "Nonlinear digital filters Principles and applications", *(Boston: kluwer academic Publishers, 1990).*

[7] Motwani M.C, Gadiya M.C, Motwani R.C. and Harris Jr. F.C., 2004, "Survey of Image Denoising Techniques", Proceedings of GSPx, Santa Clara, CA.

[8] A. Bovik, 2000, *Handbook of Image and Video Processing, Academic Press, 2000.*