Adaptive Reconfigurable Architecture for Image Denoising

Kartik V Hegde, Vadiraj Kulkarni, Harshavardhan R, Sumam David S
Department of Electronics & Communication Engineering,
NITK Surathkal, India
{hegdekartik7,vadiraj.13, vardhanharsha.r}@gmail.com, sumam@ieee.org

Abstract— **In this paper, we propose an adaptive reconfigurable architecture for image denoising. First part of this paper outlines an efficient noise detection hardware for Gaussian & impulse noise detection and suitable filters for denoising. With a robust noise detection method including a novel Gaussian noise detection method, we also explore the dynamic detection of noise in an image giving adaptability to the architecture for a better quality of denoising. Proposed architecture includes a decision making unit to find out the presence of noise as well as type of the noise, based on which a suitable filter is employed during run-time. An onboard microprocessor controls the reconfiguration and dataflow. Proposed architecture is tested on Xilinx Virtex-6 FPGA with localized noise and mixed noise conditions and it gives superior performance compared to the standard filters used. High quality denoising is achieved with simple filters on a reconfigurable region utilizing smaller area and lesser hardware resources.**

Keywords—dynamic reconfiguration; image processing; image denoising; adaptive architectures; FPGAs.

I. INTRODUCTION

Image processing applications are widely used across various domains including computer vision, artificial intelligence, pattern recognition, video processing, bio-medical applications etc. One of the most challenging and well explored area of image processing is image denoising, where the goal is to recover the original image by removing unwanted components or noise.

Noise in an image is an undesirable content which gets added during the image acquisition due to noise in electronic circuitry, due to poor illumination or other atmospheric conditions or due to sensor nonlinearities. Several methods have been explored for image denoising, both in frequency domain [1-3] and spatial domain [4-7]. Most common types of noise include Gaussian noise and impulse noise (salt-pepper noise). Gaussian noise generally appears due to the poor illumination [8] and is often removed using mean filter. Impulse noise usually appears due to errors in analog to digital conversion or bit errors in transmission [9] and is best removed using median filter. Often a part of the image does not contain any noise, thus needs no filtering. Hence, it is clear that applying a filter uniformly over the complete image is not always the best choice.

Several methods have been proposed for noise detection, mostly aimed at impulse noise detection [5,10]. However, detection of Gaussian noise is relatively a less explored area. Linear filters used for removing Gaussian noise are simpler to implement but may smoothen out the edges, whereas edge preserving bilateral filters are too complex to implement. In this paper, we propose a novel method to detect Gaussian noise which helps to retain the edges and sharpness of the image. This paper aims to address noise removal in images corrupted by more than one type of noise or localized noise.

Reconfigurable architectures are known for their combined advantage of both area & speed optimization of ASICs and flexibility of general purpose processors. Partial reconfiguration allows a part of the logic to be modified during run-time while rest of the logic continues to run intact, thus allowing to fit more logic in a given area. As improvements in high performance computing are largely driven by multi-core platforms and GPUs, they continue to suffer from disadvantages like fixed data path which forces them to use general purpose architectures. SIMD workloads like image processing are often well executed on a dedicated hardware rather than general purpose computing systems [11,12], where reconfigurable architectures can perform a better job.

Major issues to be addressed in the design of a reconfigurable system are why, how, when and what to reconfigure. We have already established the need of reconfiguration in image processing applications. We use Xilinx Partial Reconfiguration (PR) tool [13,14] for reconfiguration on FPGA. Microblaze processor running on the FPGA controls the reconfiguration using ICAP (Internal Configuration Access Port). We propose a control unit to take decision on what and when to reconfigure. The processor is interfaced with detector and reconfigures suitable filter in the filter module based on the code running on the processor.

Rest of the paper is organized as follows. Section II describes the methods used for image denoising while section III describes the algorithms used for detection of type of noise. Section IV details about the proposed architecture, Section V explains the experimental methodology, environment and results.

/15 $31.00 © 2015 IEEE
DOI 10.1109/IPDPSW.2015.126

II. Noise Detection Methods

We propose a novel three stage pipelined architecture for noise detection to detect both impulse noise and Gaussian noise. Both detectors are combined to generate a custom IP to interface it with the Micro-blaze processor with AXI protocol. This IP generates a two bit *noise_det* signal to indicate the type of noise. Each bit of *noise_det* signal represents output of each detector.

A. Gaussian Noise Detection

In Gaussian noise, pixel values vary slowly compared to their neighbors, where the distortion follows a normal distribution against the frequency of its occurrence. Detection of Gaussian noise accurately is relatively difficult compared to the impulse noise detection as the corrupted pixel values are not unusually high or low compared to their neighboring pixels.

The distortion added by the Gaussian noise to the original value results in increase of the variance of pixels values in a window. However, higher value of variance can indicate two things:

- Presence of noise
- Presence of an edge

Thus applying a smoothening filter like mean filter can potentially smoothen the edges leading to the deterioration of image quality. Here we propose a two stage detection of Gaussian noise which can differentiate between regions that contain no noise or an edge with regions that are corrupted by Gaussian noise.

Stage 1 differentiates the regions that contain noise with regions that may have an edge or corrupted by noise. Stage 2 further differentiates the regions that are corrupted by noise with regions that contain an edge.

Stage 1 calculates the variance of all the pixels in the 3x3 window which is compared with the pre-defined threshold value, Th_{var} as shown in (1),

$$\sigma_w^2 = \frac{\sum_{x=1}^{3}\sum_{y=1}^{3}(\, i(x,y) - \mu\,)^2}{9} \qquad (1)$$

where μ represents the mean of all pixels in the window and σ_w^2 represents the variance of the window.

$$noise\_edge = \begin{cases} 1, & \sigma_w^2 > Th_{var} \\ 0, & otherwise \end{cases} \qquad (2)$$

where *noise_edge* signal depicts the presence of noise or edge in the window. If *noise_edge* signal is high, 3x3 window is passed to stage 2, and if *noise_edge* signal is low, it represents the absence of noise.

Stage 2 differentiates between noise and edge using difference vectors. For a small window size, edges become piecewise linear as shown in Fig. 1.a, thus making the difference vectors as shown in the Fig. 2 to contain huge spikes. But noise, which is uniformly distributed as shown in Fig. 1.b, will not result in any spike in the magnitude of difference vectors along any direction.

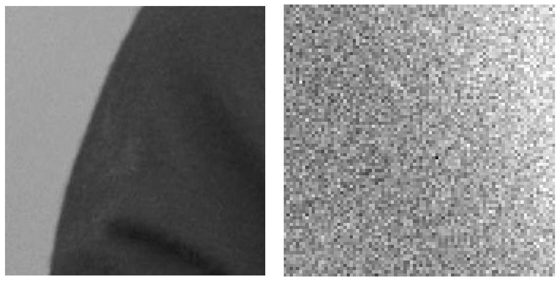

Fig. 1. Edge(a) and Noise(b) represented in a sub-window

$$\begin{aligned} D_1 &= |i(x-1,y+1) - i(x+1,y-1)| \\ D_2 &= |i(x,y-1) - i(x,y+1)| \\ D_3 &= |i(x+1,y-1) - i(x-1,y+1)| \\ D_4 &= |i(x-1,y) - i(x+1,y)| \end{aligned} \qquad (3)$$

where D_1-D_4 represents the difference vectors in four directions and i(x,y) represents the pixel value at co-ordinates (x,y) as depicted in Fig. 2.

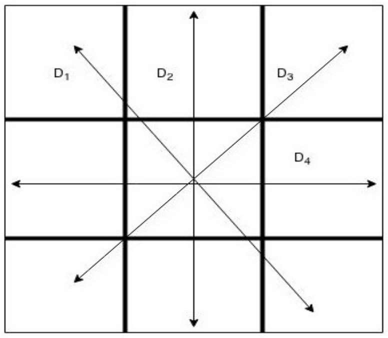

Fig. 2. Difference Vectors

Magnitudes of difference vectors in all directions are compared to obtain the maximum difference, $\max(D)$. Based on (4), detector finalizes the result as,

$$noise\_det = \begin{cases} 1, & \max(D) < Th_{diff} \\ 0, & otherwise \end{cases} \qquad (4)$$

where $\max(D)$ represents the maximum difference value and Th_{diff} represents the maximum threshold of differences for edge.

Threshold values, Th_{var} and Th_{diff}, to give the best detection performance were heuristically obtained after extensive experiments and training on several 8 bit grayscale images.

A three stage pipelined hardware was developed for Gaussian noise detection for the proposed method as depicted in Fig. 3.

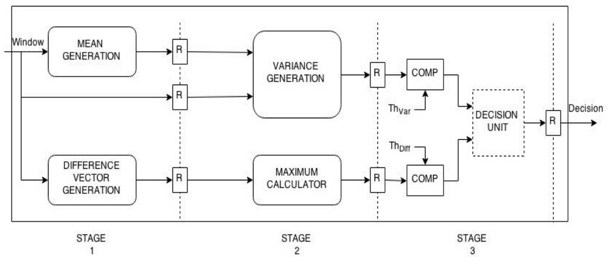

Fig. 3. Gaussian Noise Detector architecture

B. Impulse Noise Detection

Impulse noise, often referred to as 'salt and pepper' noise contains black pixels in white region or white pixels in black region. Corrupted pixels contain unusually high or low values compared to their neighbours. Impulse noise detection is a well explored area. We use a part of the design proposed by Pei-Yin Chen et al [10] for impulse noise detection.

Let i(x,y) represent the central pixel value of the 3x3 window. Then thresholds Th_{max} and Th_{min} are defined as,

$$Th_{max} = \begin{cases} 2\,I_{max}, & (2\,I_{max}) < 255 \\ 255, & otherwise \end{cases} \tag{5}$$

$$Th_{min} = \begin{cases} 2\,I_{min} - 255, & (2I_{min}) > 255 \\ 0, & otherwise \end{cases} \tag{6}$$

where I_{max} and I_{min} represent minimum and maximum pixel values in the window respectively. To classify the pixel as corrupt or not,

$$noise\_det = \begin{cases} 1, & i(x,y) \geq Th_{max} \; or \; i(x,y) \leq Th_{min} \\ 0, & x \geq 0 \end{cases} \tag{7}$$

Based on the above criterion, a three stage pipelined architecture is proposed, as depicted in Fig. 4.

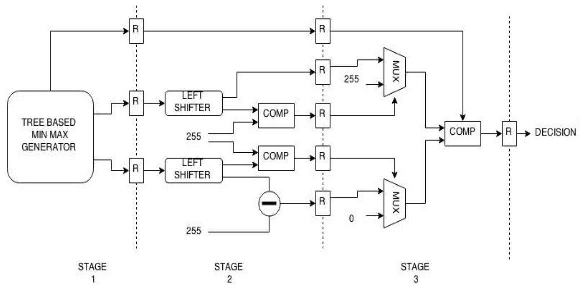

Fig. 4. Impulse Noise Detector architecture

C. Proposed Implementation

Based on Impulse and Gaussian noise detectors described, we design the noise detection system as shown in Fig. 5.

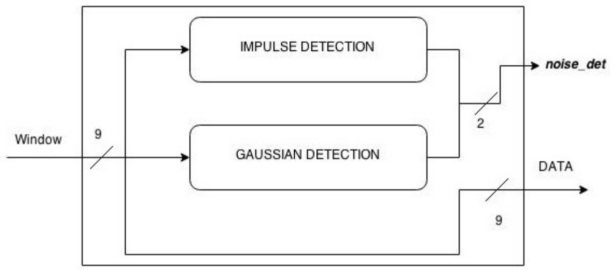

Fig. 5. Complete Noise Detection architecture

III. Image Denoising

We consider a 3x3 moving window to apply the filters for corrupted pixels on an 8 bit grayscale image.

A. Median filter

Median filter is a widely used non-linear filter known for its edge preserving quality and simple implementation. Here every corrupted pixel is replaced with the median of its neighbors. For a window, it can be performed by sorting its columns, then its rows and finally the median is obtained by sorting the diagonal.

We employ a three stage architecture as depicted in Fig. 6.

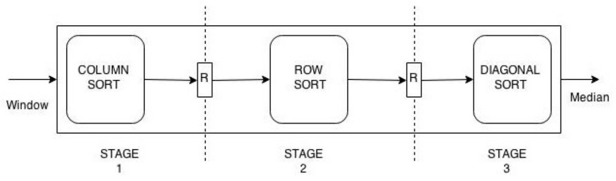

Fig. 6. Median Filtering Scheme

B. Gaussian filter

Linear smoothening filters like mean filter and Gaussian filter can effectively denoise the images affected by Gaussian noise, but they smoothen out the edges thereby reducing the image quality.

Many methods have been explored [19-21] to denoise the image while preserving the details, but are too complex for a hardware implementation. We employ a filter similar to the method used in [18] and implement it on hardware as shown in Fig.8. This method implements a low pass filter where the cut-off is based on the variance of the window. Also, this method makes use of variance, which is available from the noise detection stage.

In the window corrupted by Gaussian noise, absolute difference is calculated between central pixel and other pixels. Mean of all such pixels which satisfies (8) is calculated to replace the central pixel.

$$|i(x,y) - p|^2 < Th_{LPF}\sigma_w^2 \tag{8}$$

where i(x,y) represents pixel values, p represents central pixel value and Th_{LPF} represents a threshold value for the low-pass filter which can be heuristically estimated for the best performance.

Fig.7. describes an efficient three stage pipelined implementation of the filter. Stage 1 generates the threshold value for the low-pass filter and the differences of all pixels with central pixel calculated in parallel. Stage 2 squares the difference values and compares it with threshold in parallel. Based on the results of stage 2, mean is generated to replace the central pixel.

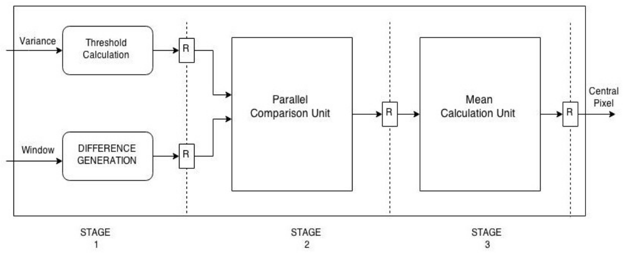

Fig. 7. Filter for Gaussian Noise removal

IV. System Architecture

The architectures described in previous sections are combined to perform image denoising on a reconfigurable platform. Noise Filter is the reconfigurable region where suitable filter is instantiated based on the result of noise detection module during run-time. Processor can communicate with these two modules using AXI interconnect block and can perform reconfiguration using AXI hardware ICAP. This section describes the details of the architecture.

A. *MicroBlaze Processor & AXI Interconnect*

We use a MicroBlaze processor with AXI interconnect along with other peripherals as shown in fig. 9. We use AXI Hardware ICAP for reading and writing to the FPGA configuration memory. The choice of the partial bit file to be reconfigured is based on the input to the processor from noise detectors in real time. Along with the peripherals, we interface two custom IPs to AXI interconnect block.

The custom IPs include *noise_detector_v1_00* for noise detection and *noise_filter_v1_00* for filtering which implement logics that are described in section III and IV. Reconfigurable partition is present within *noise_filter_v1_00* IP and suitable partial bit files which are designed externally with similar inputs and outputs are loaded onto the logic dynamically based on the choice made by the processor.

B. *Pixel Processor*

We propose a novel Pixel Processor (PP) as depicted in system diagram as shown in Fig. 9. Pixel processor performs the noise detection and filtering operations on a sub-window, which here is a 3x3 window. A pixel processor for denoising consists of two major blocks. The pixel processor receives a 3x3 window and outputs one pixel value which will be stored at a location based on input index as a part of the denoised image.

Noise detector block takes a window of the image and performs both impulse and Gaussian noise detection and gives a two bit result. Noise filter block takes the window of the image and performs the filter operation to give the filtered output. This block is the block that performs reconfiguration and is reconfigured dynamically.

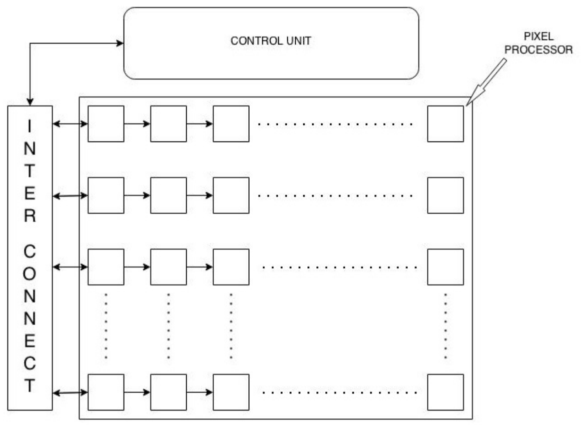

Fig. 8. SIMD processor made up of Proposed Pixel Processors

Since all the operations performed on the window are independent, these pixel processors can be replicated to form an SIMD processor exploiting data parallelism yielding high throughput as shown in Fig. 8. However, this paper describes the complete architecture of one pixel processor optimized for the best performance.

Since we use a moving window which accesses consecutive memory locations, such SIMD processor can make best use of burst access of the AXI interconnect to reduce the memory overhead.

C. *Dynamic Reconfiguration Details*

Power and performance efficient architectures for the detectors and filters are designed using Verilog and integrated into two custom IPs, namely *noise_detector_v1_00* and *noise_filter_v1_00*. Based on architectures for both the filters, we generate the bitfiles to be stored as partial bitfiles for reconfiguration.

For the complete system, netlist is generated which represents the static region in the system. Suitable software for the processor to perform reconfiguration is designed. Using Xilinx PlanAhead tool, floorplanning is performed. Based on the resource requirements, reconfigurable partition sizes are defined.

Fig. 9. Represents the complete system overview with a single pixel processor connected to the AXI interconnect.

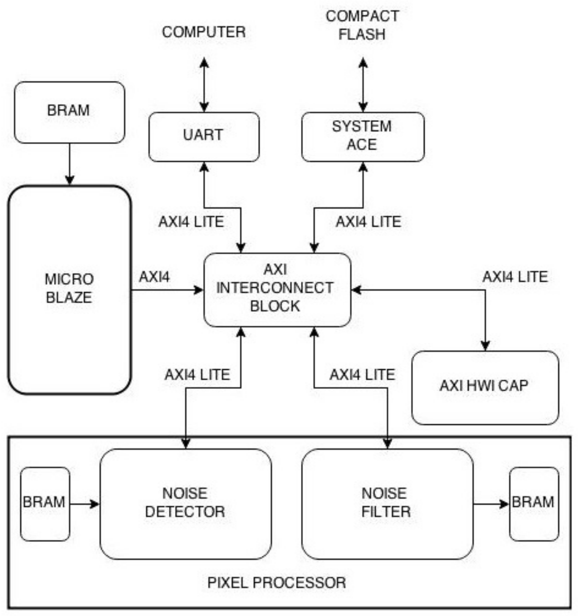

Fig. 9. System Overview

V. Experimental Results

A. *Experimental environment*

The proposed architecture was implemented on Xilinx Virtex-6 ML605 evaluation board with Xilinx ISE v14.5 and Partial Reconfiguration (PR) tool for ISE. Test image used was Lena 256x256, 8 bit grayscale image.

B. Experimental methodology

We added following types of noises:

1. Gaussian noise
2. Impulse (Salt – Pepper) noise
3. Gaussian and Impulse mixed noise where variance and density of the noise is increased simultaneously.

We tabulated following parameters of the image to check the effectiveness of the implementation:

1. Peak Signal to Noise Ratio (PSNR), calculated as,

$$PSNR = 10\ \log_{10}(\ 255^2\ /\ MSE) \tag{9}$$

where Mean Square Error (MSE), calculated as,

$$MSE = \left(\frac{1}{m_1 m_2}\right)\sum_{x=1}^{m_1}\sum_{y=1}^{m_2}\big(I(x,y) - D(x,y)\big)^2 \tag{10}$$

2. Image Enhancement factor (IEF),

$$IEF = \frac{\sum_{x=1}^{m_1}\sum_{y=1}^{m_2}\big(I(x,y) - N(x,y)\big)^2}{\sum_{x=1}^{m_1}\sum_{y=1}^{m_2}\big(D(x,y) - N(x,y)\big)^2} \tag{11}$$

where m_1 and m_2 represent the number of rows and columns, I represents original image, D represents denoised image and N represents the noisy image.

C. Results

Fig. 10-15 show the experimental results obtained for the test image with noises added.

Clearly, proposed method outperforms the standard filters significantly. This is possible due to the two important advantages the proposed architecture has:

1. Filtering is performed only when noise is found.
2. Used filter is best suited for the type of noise found.

The advantage reflects more in the case of mixed noise, where image is corrupted by both impulse and Gaussian noise. This is possible due to the adaptive nature provided by the dynamic reconfiguration.

Although it is difficult to accurately measure the reconfiguration time[22], it is possible to calculate the reconfiguration overhead based on hardware specifications. ICAP used in Virtex-6 platform runs at 100 MHz with a bandwidth of 3.2 Gbps[13] and the biggest partial bitstream is of the size of 355 kilobytes. Thus, the worst case reconfiguration time is approxilately 0.88 ms. This overhead can be masked using the proposed SIMD processor made up of multiple pixel processors as shown in Fig.8.

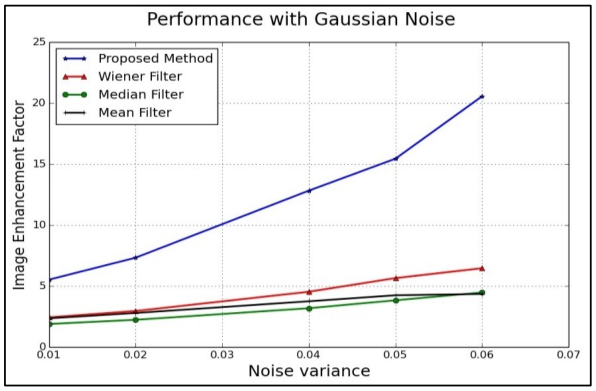

Fig. 10. IEF with Gaussian noise

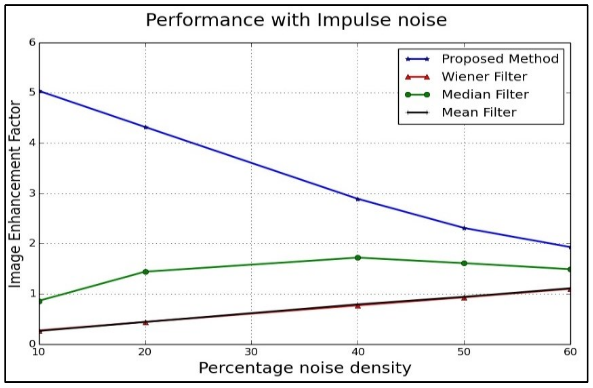

Fig.12. IEF with Gaussian noise

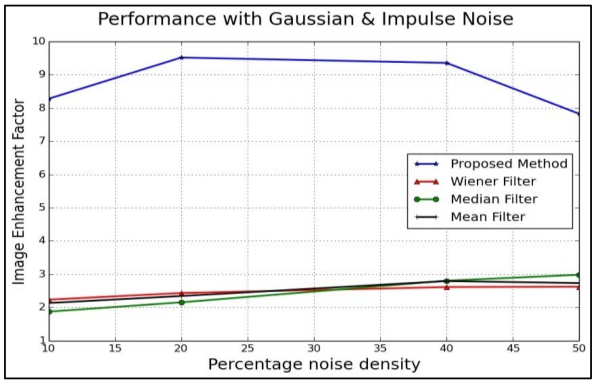

Fig.14. IEF with overlapping Gaussian & Impulse noise

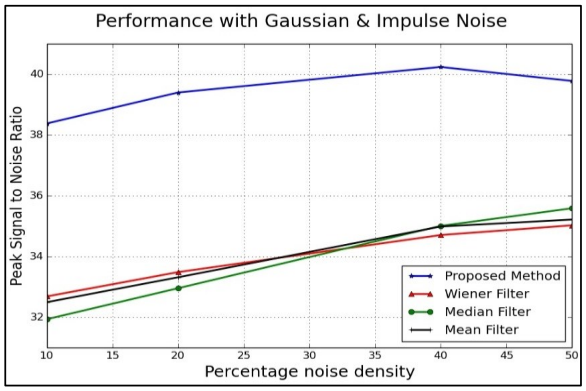

Fig.15. PSNR with overlapping Gaussian & impulse noise

TABLE V

Synthesis Report for XC6VLX240T FF1156 device

Particulars	Impulse Noise Filter	Gaussian Noise Filter	Noise Detector
Slice Registers	8	95	69
Slice LUTs	315	595	322
IOs	82	82	84
Bonded IOBs	82	81	75
DSP48E1s	0	8	8
Logical delay	3.718 ns	5.82 ns	8.612 ns

Table V shows the synthesis report for filters and noise detector. Reconfigurability of the filter module allows us to implement both impulse and Gaussian filters at the hardware usage of only Gaussian filter, thereby saving around 50% of the hardware resources per pixel processor. This saving is very significant when multiple pixel processors are instantiated as depicted in Fig. 8, thus masking the area overhead due to ICAP module added for reconfiguration.

VI. CONCLUSION

We have proposed a novel algorithm for Gaussian noise detection and proposed a noise detection unit. We have shown that the proposed implementation performs better denoising compared to other standard filters under various noise conditions. The main reason for the improvement can be attributed to the adaptive nature of the denoising scheme during run-time provided by dynamic reconfiguration.

Robust noise detection scheme not only helps to apply the right filter for the identified type of noise, but also helps to apply the filter only when noise is present, thereby preserving the image details. Improved denoising is achieved using simple filters, utilizing minimum hardware and power.

Reconfiguration allows to pack both the filters in one region, further reducing the utilization of area and hardware resources. Thus, the proposed implementation exploits the advantages of both ASICs and general purpose processors, proving the suitability of reconfigurable computing in image processing applications. Given the advantages offered by reconfiguration, multiple functionalities of image processing can be included in the same pixel processor. Thus, very significant savings in area, power and increase in throughput can be achieved by adding more functionalities needed for image processing.

Since operation performed by every pixel processor is independent, an SIMD processor as depicted in Fig.8 can be realized to exploit inherent data parallelism of image processing applications. Implemented Microblaze processor can handle the data flow and reconfiguration acting as control unit. Latency introduced by reconfiguration can be masked by implementing such a high throughput system.

ACKLOWLEDGEMENT

Authors would like to thank Mr Lakshmikant BR for his inputs regarding Reconfiguration time.

REFERENCES

[1] S. Grace Chang, Bin Yu, and Martin Vetterli, “Spatially Adaptive Wavelet Thresholding with Context Modeling for Image Denoising”, IEEE Transactions on Image Processing, Vol. 9, no. 9, September 2000.

[2] Jean-Luc Starck, Emmanuel J. Candès, and David L. Donoho, “The Curvelet Transform for Image Denoising”, IEEE Transactions On Image Processing, Vol. 11, no. 6, June 2002.

[3] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli, “Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain ”, IEEE Transactions On Image Processing, Vol. 12, No. 11, November 2003

[4] S. Esakkirajan, T. Veerakumar, Adabala N. Subramanyam, and C. H. PremChand, “Removal of High Density Salt and Pepper Noise Through Modified Decision Based Unsymmetric Trimmed Median Filter“, IEEE Signal Processing Letters, Vol. 18, No. 5, May 2011.

[5] S.Deivalakshmi, S.Sarath, P.Palanisamy, “Detection and Removal of Salt and Pepper noise in images by Improved Median Filter”, Recent Advances in Intelligent Computational Systems (RAICS), 2011.

[6] Raymond H. Chan, Chung-Wa Ho, and Mila Nikolova, “Salt-andPepper Noise Removal by Median-Type Noise Detectors and Detail-Preserving Regularization”, IEEE Transactions on Image Processing, Vol. 14, No. 10, pp. 1479-1485, Oct. 2005.

[7] Hong-yan Li Guang-long Ren Bao-jin Xiao ,Image Denoising Algorithm Based on Independent Component Analysis,WRI World Congress on Software Engineering, 2009.

[8] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice-Hall, 2001.

[9] Charles Boncelet . "Image Noise Models". In Alan C. Bovik. *Handbook of Image and Video Processing*. Academic Press, 2005.

[10] Pei-Yin Chen, Chih-Yuan Lien, and Yi-Ming Lin, A Real-time Image Denoising Chip. IEEE International Symposium on Circuits and Systems, ISCAS 2008

[11] Takashi Komuro, Shingo Kagami, and Masatoshi Ishikawa, “A Dynamically Reconfigurable SIMD Processor for a Vision Chip”, IEEE Journal of Solid-State Circuits, Vol. 39, No. 1, January 2004

[12] Kazimierz Wiatr, Specialised Architecture of Dedicated Hardware Processors for Real-Time Image Data Pre-Processing, Ninth Euromicro Workshop on Real-Time Systems, 1997.

[13] Xilinx Partial Reconfiguration, http://www.xilinx.com/tools/partial-reconfiguration.html

[14] Xilinx User Manual for Partial Reconfiguration, UG702(v14.5), Xilinx

[15] H.Hwang and R.A.Hadded,“Adaptive median filter: New algorithms and results,”IEEE Trans. Image Process., vol. 4, no. 4, pp. 499–502, Apr. 1995.

[16] K. S. Srinivasan and D. Ebenezer, “Anew fast and efficient decision based algorithm for removal of highdensity impulse noise,”IEEE Signal Process. Lett., vol. 14, no. 3, pp. 189–192, Mar. 2007

[17] Zhou Wang and David Zhang, Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images, IEEE Transactions on Circuits and Systems—Ii: Analog and Digital Signal Processing, Vol. 46, No. 1, January 1999

[18] V.R.Vijaykumar, P.T.Vanathi, P.Kanagasabapathy, “Fast and Efficient Algorithm to Remove Gaussian Noise in Digital Images”, Intl. Journal of Computer Science, vol. 37, no. 1, pp. 78-84, Feb. 2010.

[19] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” Proc. IEEE Int. Conf. Computer Vision, pp.839-846, 1998

[20] Keigo Hirakawa and Thomas W. Parks,“Image Denoising Using Total Least Squares,” IEEE Trans. on Image Processing, vol.15, no.9, Sept.2006.

[21] R. Garnett , Timothy Huegerich and Charles Chui, ‘A Universal Noise Removal Algorithm with an Impulse Detector’ IEEE Trans. on Image Processing, Vol. 14, No.11, pp.1747-1754

[22] Papadimitriou, K., Dollas, A., Hauck, S.: ‘Performance of partial reconfiguration in FPGA systems: a survey and a cost model’. ACM Transactions on Reconfigurable Technology and Systems, 2010