A Novel FPGA Implementation of Adaptive Rank Order Filter for Image Noise Removal

M. C Hanumantharaju, M. Ravishankar, D. R Rameshbabu, and S. B Satish

***Abstract*—In this paper, a Field Programmable Gate Array (FPGA) implementation of Adaptive Rank Order Filter (AROF) is proposed. AROF is a powerful technique for denoising an image corrupted by salt and pepper noise. This method provides better filtering properties then it is possible with Adaptive Median Filter (AMF). The expansion of the window size in an AMF is based on whether the median is noisy or not. However, this criterion is not an appropriate when the noise density is moderate or high. Further the pixels processed by the AMF are reused in the AMF filtering process. The restored image using this scheme generally degrades the visual quality of an image. The proposed method implements AROF in order to filter images with higher noise densities. The AROF uses median pixel or median computed from noise free pixels in order to replace noisy center pixel. The AROF adapts the window size itself when all pixels within the current window are noisy or when median itself is noisy. The AROF is implemented on Xilinx Vertex XC2VP50-7ff1152 FPGA device. The pipelining and parallel processing techniques have been adopted in order to speed up the filtering process. The experimental results show that the proposed FPGA implementation of AROF has better performance than the AMF, when the noise density is moderate or high. The performance of the proposed algorithm is verified by applying Peak Signal to Noise Ratio (PSNR) and Image Enhance Factor (IEF).**

***Index Terms*—Adaptive rank order filter, adaptive median filter, impulse noise, pipelining; verilog; FPGA.**

I. Introduction

Nowadays, digital images are usually transmitted through satellites, wired and wireless networks. However, it is quite often to introduce impulse noise in digital images during image acquisition and transmission. During transmission, images are corrupted due to interference in the channel such as atmospheric disturbance. In order to process images for better visual quality or for further use in digital image processing system, the noise has to be removed. The noise reduction algorithms remove noise without degrading image information. One approach to remove an impulsive noise is linear filters. But, linear filters tend to blur an image; therefore, these are not commonly used. Non-linear filters [1] such as median filter, order statistics filter cause little blurring and provide more satisfactory results in comparison to linear filters.

A prominent scheme to deal with salt and pepper noise is the median filter and its derivatives such as Weighted Median (WM) filter, Switched Median (SM) filter, Iterative Median (IM) filter and Center Weighted Median (CWM) filter. Median filter based approaches have low computational complexity, edge preserving capability and provide satisfactory results for low noise density. However, median filter removes very fine details and sometime changes the visual quality of an image. The main reason is that the median filter uses the rank order information of the input data within the filter window and discards its original temporal-order information. In addition, filter performance deteriorates as the noise density increases. The hardware implementation of median filter is straightforward and requires few resources such as sliding window unit, sorting network and median computation unit.

In order to overcome the drawbacks of median filter, Hwang et al. proposed an AMF [2]. The median filter algorithm uses fixed window of 3×3, 5×5 etc. However, in AMF the window size adapts according to noise density. AMF starts with an initial window size of 3×3 with the following steps: (1) Test whether the center pixel within the window is noisy or not. If the center pixel is noisy than sort pixels within window. Otherwise slide the window to next pixel and repeat this step. (2) Test whether the median is noisy or not. If the median is noisy then expand the window size and sort pixels within the window until non-noisy median pixel is found. (3) Replace the center pixel with the median value computed in step (2). As the noise density increases, AMF employs a larger window to clean up the noise. However the quality of the restored image degrades when larger window is employed. The maximum allowed window size depends on the application, but a reasonable starting value can be estimated by experimenting with various sizes of the standard median filter. This will establish a visual baseline regarding expectations on the performance of the AMF. The hardware implementation of AMF [3] with filtering window 7×7 exhibits a very good performance per cost in comparison to standard median filters. However, this filter occupies approximately 30% of the chip area and is able to remove noise level up to 60%.

Yan Lu et al. [3] proposed an optimization of sorting algorithm for median filter and its FPGA implementation. In this paper, real time median filtering using pipelining technology has been presented. This method optimizes only sorting module without concern with noisy pixels. In addition, this method may not be efficient for highly corrupted images. Zdenek Vasicek et al. [4] proposed a new FPGA

Manuscript received April 16, 2012; revised June 2, 2012.

M. C Hanumantharaju, M. Ravishankar, and D. R Rameshbabu are with the Department of Information Science and Engineering, Dayananda Sagar College of Engineering, Bangalore, India, (e-mail: mchanumantharaju@gmail.com, ravishankarmcn@gmail.com, bobrammysore@gmail.com).

S. B Satish, is with the Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bangalore, India, Pin-560078 (e-mail: satishbhairannawar@gmail.com).

implementation for AMF. In this scheme, AMF was optimized for better throughput allowing 300M pixels filtered per second at the cost of hardware complexity. The complexity of hardware increases as the noise density increases. Therefore, this approach may not be efficient at higher noise densities.

An efficient hardware implementation of median and weighted median filter using cumulative histogram to compute median have been proposed by Fahmy et al. [5]. This scheme is independent of window size and the area is being determined by bit width. The architecture presented is efficient in terms of area for larger window size; however, the system is slow for smaller window size due to hardware complexity compared with other methods. Ioannis et al. [6] proposed a new intelligent hardware module suitable for the computation of AMF. The filter process avoids image blurring and integrity of edge is preserved along with detailed information. The digital hardware used in this scheme processes an image with an 8-bit resolution, fully pipelined and processed parallel to reduce computation time. However, the window size of the system adaptively expands as noise density increases. This process increases hardware complexity and slows down the system for highly corrupted images.

In this work, FPGA implementation of AROF is proposed. This filter removes salt and pepper noise present in the images. The AROF provides better filtering properties then it is possible with AMF. The proposed method not only removes salt and pepper noise but also improves the visual quality of an image even at higher noise densities. The improved performance of the proposed method is measured using Peak Signal to Noise Ratio (PSNR) and Image Enhancement Factor (IEF).

This paper is organized as follows: Section 2 describes the proposed AROF. Section 3 gives brief details of hardware implementation. Section 4 provides experimental results and discussions. Finally conclusion is presented in Section 5.

II. Proposed Method

This paper proposes a FPGA implementation of AROF for image noise removal in order to overcome the drawback of the AMF. The AMF is an iterative algorithm, since window size becomes large for higher noise densities. Therefore, the reconstructed images using this scheme are generally not satisfactory. For a fixed window size, there may be some pixels which are not noisy, when the median is noisy. In such case, if the center pixel is replaced by some non-median pixel. The filter becomes a kind of adaptive rank orders. The reconstructed image using this method provides better visual quality than that possible with AMF. In this work, window adapts itself for two cases: (i) if all pixels within the current window are noisy. (ii) In order to replace center pixel with non-median pixel if the median is noisy. The proposed algorithm for adaptive rank order filter is as follows:

1) Read the color/gray image.
2) Select a window of size 3×3 at top left corner of the image.
3) Test whether the center pixel within the window is noisy. If yes, then go to step 4. Otherwise, slide the window to next pixel and repeat step 2.
4) Sort all pixels within the window in an ascending order and find the minimum p_{min}, median p_{med} and max. p_{max} pixel.
5) Determine if p_{med} is noisy by $p_{min} < p_{med} < p_{max}$. If it holds, p_{med} is not a noisy pixel and go to step 6. Otherwise, p_{med} is noisy pixel and go to step 7.
6) Replace the corresponding center pixel in output image with p_{med} and go to step 9.
7) By using condition $p_{min} < p_{med} < p_{max}$, check if all other pixels are noisy. If yes, then expand window and go back to step 4. Otherwise, go to step 8.
8) Replace corresponding center pixel in output image with the noise free pixel which is the closest one to the median.
9) Reset window size and center of window to next pixel.
10) Repeat the steps until all pixels are processed.

The flowchart of the AROF is as shown in Fig. 1. The visual quality of the restored images is far better than AMF. In order to apply AROF for color image, the image is separated into red (R), green (G) and blue (B) color components. The above algorithm is repeated for each of R, G and B color components.

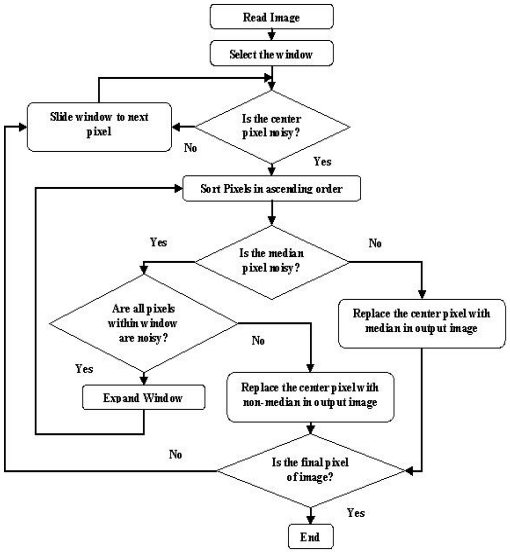

Fig. 1. Flowchart for adaptive rank order filter

III. Hardware Implementation

The proposed hardware architecture is based on pipeline stages in order to reduce computation time. In addition, parallel processing has been employed to accelerate the process. In this work, a maximum window size is chosen as 9×9 from computation point of view. The top level hardware structure of the AROF is as shown in Fig. 2. It comprises of five basic functional units, the sliding window unit, the noise detection unit, the sorting network, the median computation unit and the output selection unit. The image input data is placed in Random Access Memory (RAM). For every clock cycle, a pixel is read from RAM and placed into the sliding window module.

A. Detailed Architecture for Sliding Window Module

The pixel values of the input image (Pixel in) of width 8-bits are imported serially into the sliding window module. The Figure 3 shows the hardware architecture of 3×3 sliding window function that uses row buffers. In this work, one image pixel is read from memory in one clock cycle.

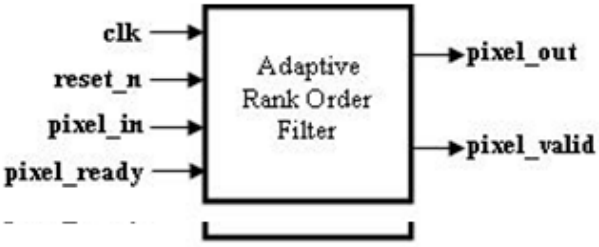

Fig. 2. Top level AROF module

The pixels are read row by row in a raster scan order. For a 3×3 sliding window, two First-In-First-Out (FIFO) buffers are used. The FIFO buffers are used to reduce the memory access to one pixel per clock cycle. The depth of the FIFO buffer is chosen as (w-3) where w is the width of the image. In order to access all values of the window for every clock cycle, the two FIFO buffers must be full.

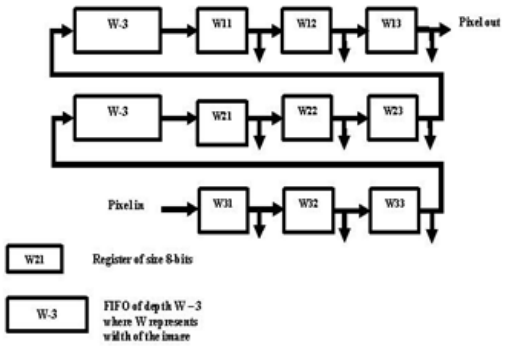

Fig. 3. Detailed architecture for sliding window 3×3 module

The contents of the window are shifted right, with the right most pixels being added to the tail of the FIFO. The top right pixel (Pixel out) is disposed after the computation on the pixel is completed, since it is not used further. In addition, 3×3 sliding window requires 9 registers to store pixels before it is placed into the noise detection unit. In this work, hardware design of sliding window has taken the advantage of certain features of FPGA. In order to achieve, read and write operations of the RAM in the same clock cycle the block RAM feature of the Xilinx Vertex FPGA have been used. However, the same effect is also achieved using Look-Up Table based (LUT) RAM implementation on FPGA. But the use of block RAM is more efficient then that possible with LUT-FPGA implementation, since logic associated with read and write operation is less. Similarly for 5×5 sliding window implementation, 4 FIFO buffers are used. The depth of FIFO is (w-5), where w represents width of the image. In this work, 7×7 and 9×9 sliding window are used for hardware implementation of AROF.

B. The Noise Detection Unit

The noise detection unit checks the pixels within window for noisy. The pixels from the sliding window are placed into the noise detection unit. If the pixels within the window are 0 or 255, then it is considered as salt and pepper noise. In this work, the noise detection unit checks if the pixels are 0 or 255. If the pixel value is equal to 0 or 255 then the noise detection unit generates output as zero. The output of the noise detection unit is fed into the sorting network.

C. The Sorting Network

A sorting network [8] is defined as a network of elementary operations (compare and swap) that sorts all input sequences. The fundamental building block of the sorting network is the compare and swap (sometimes called comparator). In sorting network, sequence of comparisons are fixed hence it is suitable for parallel processing and pipelined hardware implementation [9]. The complexity of a sorting network is measured in two ways: cost and depth. Cost represents the total number of comparators required to implement the sorting network, while depth represents the number of parallel operations required to sort all inputs. Chosen sorting algorithm influences the number of required comparators. The sorting network includes additional registers for pipelined processing. There are many ways to construct a sorting network. Batcher [10] describes two methods for constructing sorting networks: bitonic sorter and odd-even merge method. A bitonic sorter is a sorting network that accepts a bitonic sequence of numbers as input and generates a sorted sequence of numbers as output. A 0-1 is called as a bitonic sequence if it contains at most two changes between 0 and 1. In bitonic sorting network an input sequence is recursively divided into several parts. In each part bitonic sequences are created and merged in order to create another larger bitonic or sorted sequence. There is a problem with bitonic sorting, however, resulting in it failing to be a sorting network. The output of each smaller bitonic sorter is sorted, and two sorted sequences concatenated will not always be bitonic. In order to overcome this problem, the bitonic sorter [11] can be modified to flip (or reverse) the second half of the input sequence. Since both halves of the input sequence are sorted, the input sequence is not bitonic, but the effect of flipping the second half of the input sequence will result in the input sequence becoming a bitonic sequence A Batcher's odd-even merge sorting network is based on sorting two halves of the input sequence. This is followed by merging two sorted sequence into one larger sorted sequence. The Fig. 4 shows the optimal hardware structure sorting 9 elements. Each vertical line represents one compares and swaps operation.

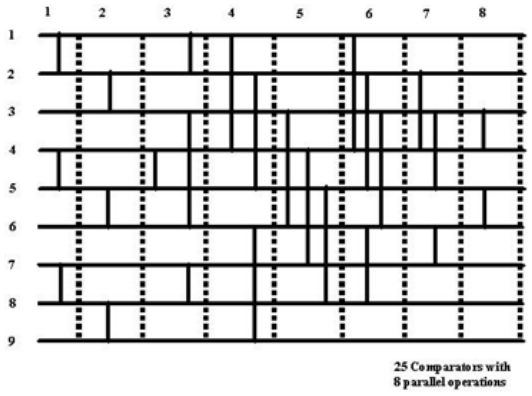

Fig. 4. Optimal sorting network

D. The Median Computation Unit

The task of the median computation unit is to compute the median value from noise free pixels. The noise detection unit discussed above converts the pixels into 0 if its value is equal to 255. However the pixels with 0 value remains as zero. The median computation unit operates after the sorting unit, therefore the noisy pixels or pixels with 0 values are ignored by this module and computes median from the noise free pixels.

E. The Output Selection Unit

The output selection unit works on priority basis. The comparator output of the first row is checked for noisy. If the first row comparator output is noisy then control jumps to second row comparator output. This process is followed up to 4th row. Apart from the above described hardware structures, AROF also uses additional logic.

IV. Experimental Results and Discussions

The hardware implementation of AROF were described in Verilog, simulated using ModelSim Version SE 6.4 and synthesized using Xilinx ISE tools Version 9.2i to Vertex XC2VP50-7ff1152 FPGA device.

Fig. 5. Adaptive rank order filter results for lena, house and tiger images

Fig. 6. AROF results of color images: lena, plane, baboon

TABLE I: Synthesis Results of Sorting Networks

	Target device : XC2VP50-7ff1152 Available Slices : 23616				
	Number of slices used				
No. of inputs	**Bubble sort**	**Bitonic sort**	**Batcher's odd-even merge sort**	**Optimal sorting**	**Max. freq. (MHz)**
9	553	449	437	446	305
25	5215	2662	2461	2568	303
49	14457	9764	7342	8468	298
81	21528	15627	15320	15590	263

First column: Noisy Lena image with salt and pepper noise levels of 50%, 70% and 90% respectively, Second Column: Restored Lena images, Third Column: Noisy Tiger image with salt and pepper noise levels of 50%, 70% and 90% respectively, Fourth Column: Restored Tiger image.

First column: Restored images with 50% noise density; Second column: Restored images with 70% noise density; Third column: Restored images with 90% noise density;

TABLE II: PSNR for AROF and AMF with Various Noise Density

Peak Signal to Noise Ratio (PSNR)						
	Lena		House		Tiger	
Noise Density	**AROF**	**AMF**	**AROF**	**AMF**	**AROF**	**AMF**
10%	41.39	41.45	41.46	41.95	40.67	41.21
20%	37.12	37.69	36.56	36.79	35.99	36.66
30%	34.47	34.81	34.05	34.67	33.40	34.32
40%	32.46	32.67	31.76	31.89	31.30	31.23
50%	30.71	29.93	30.31	30.25	29.77	29.55
60%	29.56	28.64	29.04	28.63	28.44	27.70
70%	28.46	27.81	27.90	26.32	27.26	26.42
80%	27.21	25.42	26.42	24.12	25.93	23.15
90%	25.37	21.53	24.47	20.54	23.89	19.30

TABLE III: IEF for AROF and AMF with Various Noise Densities

	Image Enhancement Factor (IEF)					
	Lena		House		Tiger	
Noise Density	**AROF**	**AMF**	**AROF**	**AMF**	**AROF**	**AMF**
10%	389.3	389.7	438.3	438.9	372.6	373.1
20%	289.4	291.5	281.9	281.6	254.7	254.3
30%	236.1	236.9	236.9	236.2	212.9	212.5
40%	200.5	200.1	187.4	187.1	174.4	174.3
50%	166.8	166.3	167.6	167.2	153.1	152.4
60%	153.4	152.2	149.5	148.9	135.7	133.7
70%	139.2	138.3	134.6	133.8	120.0	116.1
80%	119.1	116.5	109.2	97.27	101.0	97.22
90%	87.96	56.12	78.57	49.54	71.21	41.32

In order to evaluate the performance of the proposed method, PSNR [12] and IEF [13] is used: Eqns. (1) to (3).

$$PSNR = 10\log_{10}\left[\frac{255^2}{MSE}\right] \tag{1}$$

The Mean Square Error (MSE) is given by Eqn. (2)

$$MSE = \sum_{x=1}^{p}\sum_{y=1}^{q}\frac{\left(E(x,y)-I(x,y)\right)^2}{pq} \tag{2}$$

where E(x, y) is the enhanced gray pixel at position (x, y), I(x, y) is the original gray pixel at position (x, y) and, p and q denote the size of the gray image. The Image Enhancement Factor (IEF) is expressed as:

$$IEF = \frac{\sum_{x=1}^{p}\sum_{y=1}^{q}\left(n(x,y)-I(x,y)\right)^2}{\sum_{x=1}^{p}\sum_{y=1}^{q}\left(f(x,y)-I(x,y)\right)^2} \tag{3}$$

where n(x, y) is the noisy image, I(x, y) is the original image and f(x, y) is the filtered image.

The implementation costs are expressed in terms of number of slices. Table I provides synthesis results of various sorting network architectures.

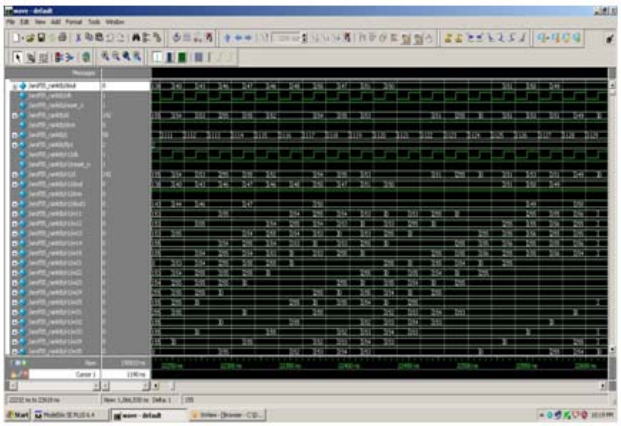

Fig. 7. AROF simulation results with 5×5 window

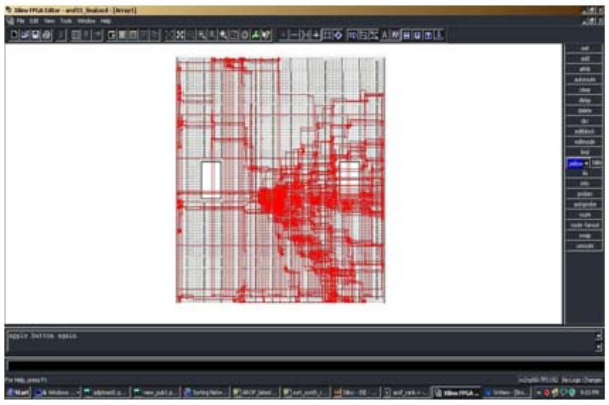

Fig. 8. AROF implementation on xilinx FPGA XC2VP50-7ff1152 device

The sorting network implemented using odd-even merge sort requires fewer slices then the bitonic sorting network. However, AROF are implemented as pipeline circuits with the maximal degree of parallelism.

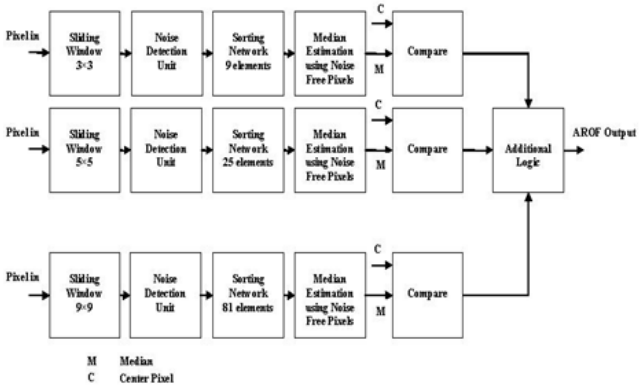

Fig. 9. Detailed architecture of proposed AROF

V. Conclusion

In this paper, FPGA implementation of AROF for salt and pepper noise removal is presented. Adaptive rank order filter provides better filtering properties than AMF. The AMF uses median value in order to replace noisy center pixel. However the AROF uses median pixel or median computed from noise free pixels for the replacement of noisy center pixel. In AMF, window expansion is based on the criterion of whether the median is noisy or not while the AROF checks if all pixels within the window are noisy. The AMF reuses the pixels which are replaced by median while the AROF avoids it. The experimental results show that the restored images with higher noise densities based on AROF are better in visual quality than that is achieved using AMF. The performance of the proposed method is verified by applying peak signal to noise ratio and image enhancement factor. Currently research work is under progress for filtering aerial images and medical images.

References

[1] J. Astola and P. Kuosmanen, "Fundamentals of Nonlinear Digital Filtering," *CRC Press*, Boca Raton, 1997.

[2] H. Hwang and R. A Haddad, "New algorithms for Adaptive Median Filters," Proceedings of Visual Communication and Image Processing, vol. 1606, pp. 400-407, Nov 1991.

[3] L. Y. Dai and M. J. Shi, "Sort Optimization Algorithm of Median Filtering Based on FPGA," *International Conference Machine Vision and Human-Machine Interface (MVHI)*, pp. 250-253, 24-25 April, 2010.

[4] Z. Vasicek and L. Sekanina, "Novel Hardware Implementation of Adaptive Median Filters," in *proceedings of 11th IEEE workshop on Design and Diagnostics of Electronic Circuits and Systems*, pp. 1-6, 16-18 April, 2008.

[5] S. A Fahmy, P. Y. K Cheung, and W. Luk, "Novel FPGA-based Implementation of Median and Weighted Median Filters for Image Processing," *International Conference on Field Programmable Logic and Applications*, pp. 142-147, 2005.

[6] I. Andreadis and G. Louverdis, "Real-Time Adaptive Image Impulse Noise Suppression," *IEEE transactions on Instrumentation and Measurement*, vol. 53, no. 3, pp. 798-806, June 2004.

[7] G. Louverdis, I. Andreadis, and N. Papamarkos, "An intelligent Hardware Structure for Impulse Noise Suppression," *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, 2003.

[8] S. M. Meena and K. Linganagouda, "Rank Based Merge Sorting Network Architecture for 2D Median and Morphological Filters," *IEEE International Conference on Advance Computing*, pp. 473-479, 6-7 March, 2009.

[9] C. Chakrabarti, "Novel Sorting Based Architectures for Rank Order Median Filters," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 2, no. 4, pp. 502-507, 1994.

[10] Z. Vasicek and L. Sekanina, "An Evolvable Hardware System in Xilinx Virtex II Pro FPGA," *International Journal Innovative Computing and Applications*, vol. 1, pp. 63-73, 2007.

[11] R. Maheshwari, S. S. S. P. Rao, and E. G. Poonach, "FPGA Implementation of Median filter," VLSI Design, *IEEE Computer Society*, pp. 523-524, 1997.

[12] K. Wiatr, "Median and Morphological Specialized Processors for a Real –Time Image Data Processing," *EURASIP Journal on Applied Signal Processing*, 2002

[13] L. Breveglieri and V. Piuri, "Digital Median Filters," *Journal of VLSI Signal Processing Systems for Signal*, Image and Video Technology, vol. 31, no. 3, pp.191-206, 2002.

[14] D. Richards, "VLSI Median Filters," *IEEE Transactions on Acoustics, Speech and Signal processing*, vol. 38, no. 1, pp.145-53, 1990.

M. C Hanumantharaju obtained his B.E degree from Bangalore University with Electronics and Communication Engineering in the year 2001, M.Tech degree from Visvesvaraya Technological University with Digital Communication and Network Engineering in 2004, Belgaum, India. He is currently a research scholar at the Department of Information Science & Engineering, Dayananda Sagar College of Engineering, Bangalore, India. His research interests include VLSI Architecture Development for Signal & Image Processing Applications, Synthesis & Optimization of ICs, DSP with FPGAs etc.