FPGA Implementation and Simulation of Guided Image Filtering

Rajapriya, S, Rijuvana Begum, A, Kannan, G
Department of ECE
Prist University
Thanjavur, India.
{rajapriyaselvaraj, arbbegum}@gmail.com, contact@prist.ac.in

ABSTRACT: *Field Programmable Gate Array (FPGA) technology has become a viable target for the implementation of real time algorithms suitable to video image processing applications. The exclusive architecture of the FPGA has allowed the technology to be used in many applications encompassing all aspects of video image processing. In this paper, we propose a novel type of explicit image filter- guided filter. Resulting from a local linear model, the guided filter produces the filtering amount fashioned by considering the content of a guidance image, which can be the input image itself. The guided filter has a fast and non-approximate linear-time algorithm, whose computational intricacy is independent of the filtering kernel size. The guided filter is together effective and efficient in a great variety of computer vision and computer graphics applications including noise reduction, detail smoothing/enhancement, image matting/feathering. In this work, an implementation guided image filtering using a FPGA Xilinx Spartan 3E, with educational purposes, is obtainable. The system is connected to a USB port of a personal computer, in that way it forms a authoritative and low-cost intend station.The FPGA-based system is accessed through a Matlab graphical user interface, which handles the communication arrangement. A similarity between results obtained from MATLAB simulations and the described FPGA-based implementation is obtained.*

Keywords: Image Processing, FPGA, Filtering, Edge-preserving Filtering

Received: 11 October 2013, Revised 19 November 2013, Accepted 27 November 2013

1. Introduction

In the recent times, Field Programmable Gate Arrays have become a viable target for the implementation of various algorithms related to video image processing applications. The unique architecture of FPGA supports many applications concerning high speed processing. With increase in image sizes, the use of software for image processing are being replaced by real time systems required for high speed image and video processing. Specialized image processing programs cannot adequately process very large amounts of streaming data on PCs since they have been designed for general purpose applications. In order to accomplish further optimization, hardware devices need to be used. Increase in FPGA speeds and capacities have followed or exceeded Moore's law for the last several years.

FPGAs do not have inbuilt cache mechanisms. They are usually connected to a number of local memories so as to maximize throughput. However, the available space in FPGA imposes a restriction on the size of a block of code which is to be executed in a single clock cycle. As long as the code fits on the chip, the performance of the FPGA is largely based on Memory Traffic and

Clock Frequency. Results have shown that complex tasks have resulted in very large speedups as high as 800 by utilizing the parallelism within FPGAs.Our goal is to familiarize application programmers with the state of the art in compiling programs using hardware description language to FPGA and to show how FPGAs implement a Guided Image Filtering Techniques.

A digital image is very much essential for daily life applications such as satellite television, medical imaging (magnetic resonance imaging, ultrasound imaging, x-ray imaging), computer tomography etc. It is also essential for the researches in the areas of Science and Technology such as geographical information systems and radio astronomy. The images collected by different type of sensors are generally contaminated by different sources of noise. Noise may be generated due to imperfect instruments used in image processing, problems with the data acquisition process, and interference, all of which can degrade the data of interest. Furthermore, noise can be introduced by transmission errors and compression also. So image enhancement is a necessary task in image processing.

Image enhancement improves the quality (clarity) of images for human viewing and computer vision. It basically improves the interpretability or perception of information in images and providing 'better' input for other automated image processing techniques. The principal objective of image enhancement is to modify attributes of an image to make it more suitable for a given task and a specific observer.The image enhancement methods can broadly be divided in to the following two categories:

1. Spatial Domain Methods
2. Frequency Domain Methods

In spatial domain techniques, we directly deal with the image pixels. The pixel values are manipulated to achieve desired enhancement. In frequency domain methods, the image is first transferred in to frequency domain. It means that, the Fourier Transform of the image is computed first. All the enhancement operations are performed on the Fourier transform of the image and then the Inverse Fourier transform is performed to get the resultant image. These enhancement operations are performed in order to modify the image brightness, contrast or the distribution of the grey levels. As a consequence the pixel value (intensities) of the output image will be modified according to the transformation function applied on the input values.

Guided Image filtering is one of the spatial Domain Enhancement technique in which the filtering output is locally a linear transform of the guidance image. A Guidance image is an input image itself or another image (flashed input image) but not entirely different image. The guided filter provides more structured output than the input image. Guided filter performs variety of applications such as Image Smoothing, Enhancement, HDR Compression, Flash/No Flash Imaging, Matting /Feathering, Dehazing and Joint Up sampling. Guided Image Filter is implemented using Box Filter. BOX filter can be computed by two methods like Moving Sum Method and Integral Image method. In This paper Guided Image Filtering algorithms are simulated in both MATLAB and VHDL.

This paper is organized as follows. In section II FPGA methodology for Guided image filter is discussed .Section III introduces the algorithms for Guided filter and its applications. Section IV presents conclusion and future work.

2. Methodology

The FPGA based designing approach is divided into two methods–VHDL and Verilog. The limitation of HDL programming is the difficulty in implementing algorithmic expressions that takes place in applications programming. The Cameron project's SA-C (Single Assignment C language) and Celoxica's Handel C are two popularly used cross-compilers which map high-level algorithmic language on reconfigurable hardware. However, VHDL was chosen in this paper due the flexibility and controllability it offers, over timing constraints and hardware.

This paper uses the methodology to simulate a link between an Image Acquisition System and an FPGA device namely Direct Software Link. MATLAB was chosen as the image end software due to its capability to work directly with matrices.The MATLAB environment is much suited for PC based image processing applications.MATLAB allows the designer to treat a color image as a 3-dimensional matrix of red, green and blue information of 8 bits each and develop optimized matrix operations. In this paper, MATLAB is used for rasterization of color image into pixels.These pixels are then transported to required HDL simulation software via the direct software link.For viewing the input images and Processed output images are displayed using Visual Basic Software.

3. Guided Filter

In this paper, we consume a novel explicit image filter called guided filter.Resulting from a local linear model, the guided filter produces the filtering amount produced by considering the content of a guidance image, which can be the input image itself or an additional different image. The filtering output is locally a linear transform of the guidance image. On one hand, the guided filter has superior edge-preserving smoothing properties like the bilateral filter, but it does not endure from the gradient setback artifacts. On the other hand, the guided filter can be used beyond smoothing.With the help of the guidance image, it can make the filtering output more prearranged and less smoothed than the input. We make obvious that the guided filter performs very well in a great variety of applications together with image smoothing/improvement, HDRcompression, flash/no-flash imaging, matting/feathering, dehazing and combined upsampling.

Moreover, the guided filter naturally has an $O(N)$ time (in the number of pixels N) non approximate algorithm for both gray-scale and high dimensional images, regardless of the kernel size and the intensity range. Classically, our CPU implementation achieves 40 ms per mega-pixel performing gray-scale filtering.To the best of our understanding, this is one of the fastest edge-preserving filters. The guided filter enables a high-quality real-time $O(N)$ stereo matching algorithm. The guided filter has also been applied in optical flow estimation, interactive image segmentation, saliency detection and illumination rendering. We believe that the guided filter has great potential in computer vision and graphics, given its unfussiness, efficiency, and premium.

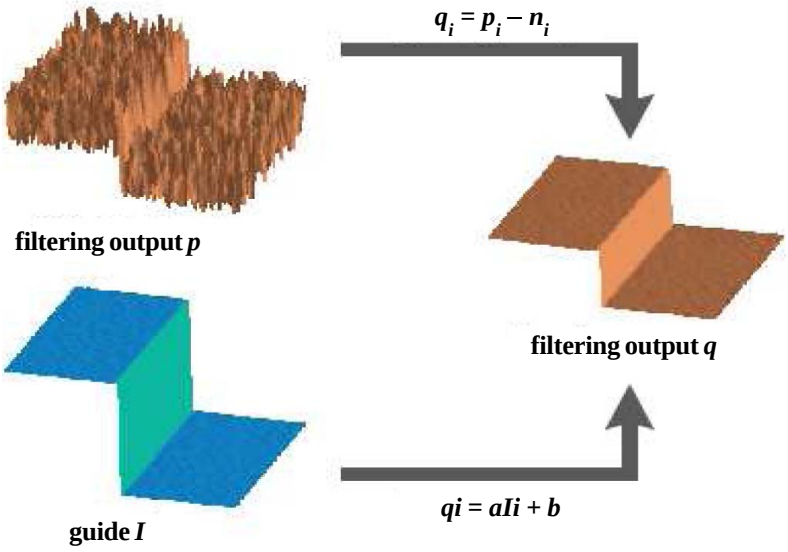

Figure 1. Guided Image Filtering Process

3.1 Definition

Now we define the guided filter. The key assumption of the guided filter is a local linear model between the guidance I and the filtering output q. We assume that q is a linear transform of I in a window centered at the pixel k:

$$q_i = a_k I_k + b_k, \forall_i \in \omega_k \tag{1}$$

where (a_k, b_k) are some linear coefficients assumed to be constant in ω_k. We use a square window of a radius r. This local linear model ensures that q has an edge only if I has an edge, because $\nabla q = a \nabla I$. This model has been proven useful in image super-resolution, image matting and dehazing.

To determine the linear coefficients (a_k, b_k), we need constraints from the filtering input p.We model the output q as the input p subtracting some unwanted components n like noise/textures:

$$q_i = p_i - n_i \tag{2}$$

We seek a solution that minimizes the difference between q and p while maintaining the linear model (1). Specifically, we minimize the following cost function in the window ω_k:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_k + b_k - p_i)^2 + \varepsilon a_k^2) \quad (3)$$

Here, ε is a regularization parameter penalizing large a_k. We will investigate its intuitive meaning in Section 3.2.

Equation (3) is he linear ridge regression model and its solution is given by,

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i - \mu_k p_k}{\sigma_k^2 + \varepsilon} \quad (4)$$

$$b_k = p_k - a_k \mu_k \quad (5)$$

Here μ_k, σ_k^2 are mean and variance of guided image in ω_k. $|\omega|$ is the number of pixels in ω_k and $\bar{p}_k = \frac{1}{|\omega|}\sum_{i \in \omega_k} p_i$ is the mean of p in ω_k. Having obtained the linear coefficients (a_k, b_k), we can compute the Filtering output q_i by (1). Figure 1 shows an illustration of the guided filtering process.

However, a pixel i is involved in all the overlapping windows ω_k that covers i, so the value of q_i in (1) is not identical when it is computed in different windows. A simple strategy is to average all the possible values of q_i. So after computing (a_k, b_k) for all windows ω_K in the image, we compute the filtering output by

$$q_i = \frac{1}{|\omega|}\sum_{i \in \omega_k} (\bar{a}_k I_K + \bar{b}_k) \quad (6)$$

Noticing that $\sum_{(k \mid i \in \omega_k)} a_k = \sum_{k \in i} a_k$ due to symmetry of the box window (6) by

$$q_i = (\bar{a}_i I_i + \bar{b}_i) \quad (7)$$

Where $\bar{a}_i = \frac{1}{|\omega|}\sum_{i \in \omega_i} a_k$ and $\bar{b}_i = \sum_{i \in \omega_i} b_k$ are the average coefficients of all windows overlapping i. The averaging strategy of overlapping windows is popular in image denoising and is a building block of the very successful BM3D algorithm

With the modification in (7), ∇q is no longer scaling of ∇q because the linear coefficients ($\bar{a}_\iota, \bar{b}_\iota$) vary spatially. But as ($\bar{a}_\iota, \bar{b}_\iota$) are the output of a mean filter, their gradients can be expected to be much smaller than that of I near strong edges. In this situation we can still have $\nabla q \approx a \nabla I$, meaning that abrupt intensitychanges in I can be mostly preserved in q.

Equation (4), (5), (7) are the definition of the guided filter. A pseudocode is in Algorithm1. In this algorithm,mean is a mean filter with a window radius r. The abbreviations of variance (var), and covariance (cov) indicate the intuitive meaning of these variables.

Algorithm 1. Guided Filter

Input : Filtering input image p, guidance image I, radius r, regularization parameter .

Output : Filtering output q.

1. Read the image say I, it acts as a guidance image.
2. Make $p = I$, where p acts as our filtering image.
3. Enter the values assumed for r and eps (ε), where r is the local window radius and eps is the regularization parameter.
4. Compute the mean of I, p, $I * p$.
   mean_I = BF (I, p). / N

mean_p = BF (p, r). / N

mean_Ip = BF (I .* p, r). / N

BF is the Box Filter for calculating the mean.

5. The compute the covariance of (I, p) using the formula:-

cov_Ip = mean_Ip – mean_I .* mean_p;

6. Then compute the mean of (I * I) and use it to compute the variance using the formula:

var_I = mean_II – mean_I .* mean_I

7. Then compute the value of a, b. where a, b are the linear coefficents.

8. Then compute mean of both a and b.

9. Finally obtain the filtered output image q by using the mean of a and b in the formula

q = mean_a .* I + mean_b;

10. Display the output along with the input image.

mean value is calculated using 1-Dimensional Box Filter by Algorithm 2. The 1-Dimensional Box Filter is implemented using the Integral Image Technique.We can also utilize the moving Sum method for Box Filter to calculate the mean.But the integral image representation is a remarkable idea that permits to evaluate the sum of image values over rectangular regions of the image, regardless of the size of the region.

Algorithm 2. 1D Box Filter via integral Image Technique

Input: input signal p, radius r

Output: output signal s

1. The cumulative sum of the array of pixels over an interval is calculated using

$S[i] = \sum_{i=0}^{r} p[i]$

2. Set the Initial Value as $S[0] = p[0]$ for quick Calculation.

3. for $i = 1$ to end **do**

4. $S[i] = S[i-1] + p[i]$

5. end for

The 2-Dimensional Box filter is shown in Figure 2. we take as input an image I. We will compute an image S, called the integral image:

$$S[x, y] = \sum_{x' \le x, y' \le y} I[x', y']$$

we can compute this quickly, by doing each row in one pass and each column in a second pass. We will have an intermediate image C [x, y], and we will compute:

$C[0, y] = I[0, y]$

$C[x, y] = C[x-1, y] + I[x, y]$

$S[x, 0] = C[x, 0]$

$S[x, y] = S[x, y-1] + C[x, y]$

Suppose that we are given our rectangle as two points ($x1$, $y1$), ($x2$, $y2$), where ($x1$, $y1$) is the upper left corner of the rectangle, and ($x2$, $y2$) is the lower right corner of the rectangle. Then the sum over the rectangle is Calculated using inclusion and exclusion.

$$R(x1, y1, x2, y2) = S[x2, y2] - S[x1-1, y2] - S[x2, y1-1] + S[x1-1, y1-1].$$

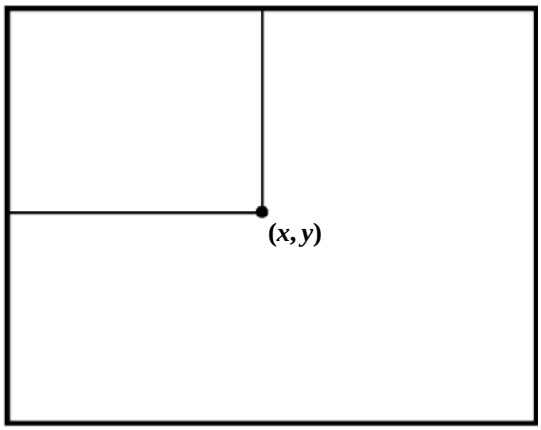

Figure 2. The integral image at (*x*, *y*) has the sum over the box with corners (0, 0) and (*x*, *y*)

Based on Algorithm 1 and 2 the following applications Edge Preserving, Detail Enhancement, Flash / No Flash Denoising, Guided Feathering are simulated using MATLAB and VHDL and the results are compared.

3.2 Edge – Preserving Filtering

Given the definition of the guided filter, we initial study the edge-preserving filtering property. Here we investigate the special case where the guide *I* is identical to the filtering input *p*. We can see that the guided filter behaves as an edge-preserving smoothing operator. The edge-preserving filtering property of the guided filter can be explained as following. Consider the case $I \equiv p$. In this case, $a_k = \sigma_k^2 / (\sigma_k^2 + \varepsilon)\, b_k = (1 - a_k)\, \mu_k$. It is clear that if $\varepsilon = 0$, then $a_k = 1$ and $b_k = 0$. If $\varepsilon > 0$, we consider two cases.

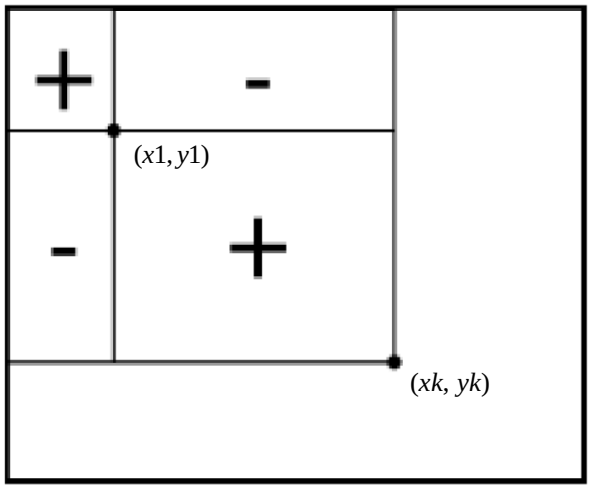

Figure 3. Using Inclusion-Exclusion

Case 1: "*High Variance*". If the image changes a lot within ω_k.Then $\sigma_k^2 >> \varepsilon$. So $a_k \approx 0$ and $b_k \approx 0$.

Case 2: "*Flat patch*". If the image I is almost constant in ω_k. Then $\sigma_k^2 >> \varepsilon$. So $a_k \approx 0$ and $b_k \approx \mu_k$.

More specifically, the criterion of a "*flat patch*" or a "*high variance*" one is given by the parameter .The patches with variance (σ^2) much smaller than are smoothed, whereas those with variance to a great extent larger than ε are preserved.

3.3 Detail Enhancement

Detail Enhancement is an advanced non linear image processing method that preserves details in high dynamic range imagery. This meticulous image is enhanced so that it matches the total dynamic range of the original image, thus making the details

perceptible to the operator even in scenes with extreme temperature dynamics.

Though the guided filter is an edge-preserving smoothing operator like the bilateral filter, it avoids the gradient setback artifacts that may appear in detail enhancement and HDR compression. Given the input signal p, its edge-preserving smoothed output is used as a base layer q. The difference between the input signal and the base layer is the detail layer : $d = p - q$. It is magnified to boost the details. The enhanced signal is the combination of the boosted detail layer and the base layer.

Input Image Guidance Image Output Image

Figure 4. Results from HDL simulation of Edge-Preserving filtering

Input Image Guidance Image Output Image

Figure 5. Results from MATLAB simulation of Edge-Preserving filtering

The guided filter performs better on avoiding gradient setback. In fact, if we use the patch-wise model (1), it is guaranteed to not have gradient reversal. In the case of self-guided filtering ($I \equiv p$). In this case, (4) gives $a_k = \sigma_k^2 / (\sigma_k^2 + \varepsilon) < 1$ and b_k is a constant So we have $\partial_x q = a_k \partial_x p$ and the detail layer gradient $\partial_x d = (\partial_x p - \partial_x q) = (1 - a_k)\, \partial_x p$, meaning that $\partial_x d$, $\partial_x p$ are always in same direction.When we use overlapping model (6) instead of, we have $\partial_x q = \bar{a} \partial_x p + p \partial_x \bar{a} + \partial_x \bar{b}$.

Input Image Guidance Image Output Image

Figure 6. Results from HDL simulation of Detail Enhancement

3.4 Guided Feathering

The guided filter is not simply a smoothing filter. Due to the local linear model of $q = aI + b$, the output q is locally a scaling (plus an offset) of the guidance I. This makes it possible to transfer arrangement from the guidance I to the output q, even if the filtering input p is smooth.

The structure-transferring filtering, introduce an application of guided feathering. Feathering is a technique used in computer graphics software to smooth or blur the edges of a feature. Matting refers to the dilemma of exact foreground estimation in images and video. It is one of the key techniques in many image bowdlerization and film production applications. In this method greatly reduces the obscurity for extracting an exact matte.

Input Image Guidance Image Output Image

Figure 7. Results from MATLAB simulation of Detail Enhancement

Input Image Guidance Image Output Image

Figure 8. Results from HDL simulation of Guided Feathering

Input Image Guidance Image Output Image

Figure 9. Results from MATLAB simulation of Guided Feathering

A binary mask is refined to appear an alpha matternear the object boundaries . The binary mask can be obtained from graph-cut or other segmentation methods, and is used as the filter input preserve that the guided filter faithfully recovers the hair, even though the filtering input p is binary and very rough. The guided filter does not lose thin structures because the guided filter has a patch-wise model.The structure-transferring filtering is an important property of the guided filter. It enables new filtering-based applications, including feathering/matting and dehazing.

3.5 Flash / No Flash Denoising

Digital photography has made it possible to quickly and easily take a pair of images of low-light environments.one with flash to capture detail and one without flash to capture ambient illumination. Guided image Filter analyze and combine the strengths of such flash/noflash image pairs. It include denoising and detail transfer (to merge the ambient qualities of the no-flash image with the high-frequency flash detail), white-balancing (to change the color tone of the ambient image), continuous flash (to interactively adjust flash intensity), and red-eye removal (to repair artifacts in the flash image). We demonstrate how these applications can synthesize new images that are of higher quality than either of the originals.

Input Image Guidance Image Output Image

Figure 8. Results from HDL simulation of Flash / No Flash Denoising

Input Image Guidance Image Output Image

Figure 9. Results from HDL simulation of Flash / No Flash Denoising

4. Conclusion and Future Work

This paper has demonstrated the simulation of Guided Image Filtering and its applications on FPGA. A comparison of results obtained from VHDL and MATLAB has been made. Future work would be to extend the capability of this system to process the Mean Square Error and Peak Signal to Noise Ratio. Also an improved hardware implementation of the above simulated routines would be the next course of action.

References

[1] Tomasi, C., Manduchi, R. (1998). Bilateral Filtering for Gray and Color Images, *In*: Proc. IEEE Int'l Computer Vision Conf.

[2] Yang, Q., Tan, K.-H., Ahuja, N. (2009). Real-Time O (1) Bilateral Filtering, *In*: Proc. IEEE Conf. Computer Vision and Pattern Recognition, p. 557-564.

[3] Adams, N. Gelfand, Dolson, J., Levoy, M. (2009). Gaussian KDTrees for Fast High-Dimensional Filtering, *In*: Proc. ACM Siggraph, p. 21:1-21:12.

[4] He, K., Sun, J., Tang, X. (2010). Guided Image Filtering, *In*: Proc.European Conf. Computer Vision, p. 1-14.

[5] Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M. (2011). Fast Cost-Volume Filtering for Visual Correspondence and Beyond, *In*: Proc. IEEE Conf. Computer Vision and Pattern Recognition, p. 3017-3024.

[6] Ding, Y., Xiao, J., Yu, J. (2011). Importance Filtering for Image Retargeting, *In*: Proc. IEEE Conf. Computer Vision and Pattern Recognition, p. 89-96.

[7] He, K., Rhemann, C., Rother, C., Tang, X., Sun, J. (2011). A Global Sampling Method for Alpha Matting, *In*: Proc. IEEE Conf. Computer Vision and Pattern Recognition, p. 2049-2056.

[8] He, K., Sun, J., Tang, X. (2010). Fast Matting Using Large Kernel Matting Laplacian Matrices, *In*: Proc. IEEE Conf. Computer Vision and Pattern Recognition, p. 2165-2172.

[9] Katkovnik, V., Foi, A., Egiazarian, K., Astola, J. (2010). From LocalKernel to Nonlocal Multiple-Model Image Denoising, *Int'lJ. Computer Vision*, 86 (1) 1-32.

[10] Dabov, K., Foi, R., Katkovnik, V., Egiazarian, K. (2007). Image Denoising by Sparse 3D Transform-Domain Collaborative Filtering, *IEEE Trans. Image Processing*, 6 (8) 2080- 2095, Aug.

[11] Allin Christe, S., Vignesh, M., Kandaswamy, A. (2011). An Efficient FPGA Implementation of MRI Image Filtering and Tumour Characterization using Xilinx System Generator, *International Journal of VLSI design & Communication Systems* (VLSICS) 2 (4), December.

[12] Alba M. Sánchez, G., Ricardo Alvarez, G., Sully Sánchez, G. (2007). Architecture for filtering Ximages using Xilinx System Generator, *International Journal of Mathematical Models and Methods in Applied Sciences*, V. 1.

[13] Daggu Venkateshwar Rao , Shruti Patil, Naveen Anne Babu ,V Muthukumar. (2006). Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture usingCbased Hardware Descriptive Languages, *International Journal of Theoretical and Applied Computer Sciences*, 1 (1) 9–34.

[14] Ye Li, Qingming Yao, Bin Tian,Wencong Xu. (2011). Fast doubleparallel image processing based on FPGA, IEEE International Conference on Vehicular Electronics and Safety (ICVES), 16, p. 97 - 102, Mar.

[15] Sami Hasan, Alex Yakovlev, Said Boussakta. (2010). Performance Efficient FPGA Implementation of Parallel 2-D MRI Image Filtering Algorithms using Xilinx System Generator, CSNDSP IEEE.

[16] Satish K.Shah, Rakesh K.Soni, Brijesh Shah. (2011). FPGA Implementation of Image Compression using bottom- up approach of Quad tree technique, *IETE Journal of Research*, 57 (2), Mar-Apr.

[17] Youn-Hong Kim, Kyong-il Jun, Kang-Hyeon Rhee. (1999). FPGA Implementation of Subband Image Encoder Using Discrete Wavelet Transform, IEEE TENCON.

[18] Mohd Fauzi Bin Othman, Norarmalina Abdullah, Nur Aizudin Bin Ahmad Rusli. (2010). An Overview of MRI Brain Classification using FPGA Implementation, IEEE Symposium on Industrial electronics & Applications (ISIEA) Oct, Malaysia.

[19] Implementation of Image Processing Algorithms on FPGA hardware. (2000). by Anthony Edward Nelson, MS thesis, May.

[20] Elamaran, V., Rajkumar, G. (2012). FPGA Implementation of Point Processes using Xilinx System Generator, *Journal of Theoretical and Applied Information Technology*. 41 (2).

[21] Anthony Edward Nelson. (2000). Implementation of Image Processing Algorithms on FPGA hardware, May,VanderbiltUniversity, Nashville, TN.

[22] Hammes, J., Bohm, A. P. W., Ross, C., Chawathe, M., Drapper, B., Najjar, W. (2004). High Performance Image Processing on FPGAs, Feb, Computer Science Department, Colorado State University, Ft. Collins, CO, USA.

[23] Bruce A. Draper, Ross Beveridge, J., Willem Bohm, A. P., Charles Ross, Monica Chawathe. (2003). Accelerated Image Processing on FPGAs, *IEEE Trans. Image Processing*, 12, p. 1543-1551, Dec.