Image Smoothing Based On FPGA

Dr. Maha A. R. Hasso **Farah Saad Al-Mukhtar**
Department of Computer Science
College of Computer Sciences and Mathematics
University of Mosul

Received **Accepted**
04 / 06 / 2013 **12 / 09 / 2013**

الملخص

يظهر المنطق القابل للبرمجة كحل فعال للعديد من تطبيقات معالجة الصور، ومع الزيادة في حجم الصور فان البرمجيات (software) أصبحت أقل فائدة في معالجة الصور لذلك اصبحت تقنية مصفوفة البوابات الحقلية القابلة للبرمجة FPGA الهدف الفعال لتطبيق الخوارزميات التي تناسب تطبيقات معالجة الصور واستخدمت هذه التقنية في العديد من تطبيقات معالجة الصور.

تنعيم الصور هي واحدة من هذه التطبيقات والتي تستخدم لتقليل تأثير ضوضاء نقطة في الصورة.

تم في هذا البحث استخدام لغة توصيف العتاد للدوائر المتكاملة ذات السرعات المرتفعة جدا VHDL وهي اللغة التي تصف هيكلية التصميم في FPGA لتمثيل اثنين من المرشحات المستخدمة في تنعيم الصور وهما (مرشح الوسيط median filter ومرشح المعدل average filtering) وتم أيضاً في هذا البحث تطبيق هذه المرشحات على FPGA وقد أثبتت النتائج سرعة عالية في اداء الخوارزميات التي تعتمد على الاجهزة والبرمجيات مقارنة بالبرمجيات لوحدها اذ ظهر فرق كبير جدا في سرعة التنفيذ بالاعتماد على الاجهزة المادية التي أظهرت سرعة تنفيذ بمقياس 10^{-9} من الثانية بينما بتطبيق الخوارزميات تقاس بالثواني.

تم تطبيق البرمجيات في هذا البحث باستخدام لغة MATLAB 2010 البرمجية وكذلك لغة VHDL للتعامل مع جهاز FPGA المستخدم والذي كان من نوع Xilinx XC3S500E Spartan-3E.

Abstract

Programmable logic is emerging as an attractive solution for many digital image processing applications. As image sizes and bit depths grow

larger, software has become less useful in the image processing, Field Programmable Gate Array (FPGA) technology has become a viable target for the implementation of algorithms suited to image processing applications, the unique architecture of the FPGA has allowed the technology to be used in many such applications encompassing all aspects of image processing.

Image smoothing is one of image processing applications, it often done to reduce the effect of pixel noise in images. This paper presents VHDL architectures (that allow description of the structure design of FPGA) to implement two of image smoothing filters:

a) averaging filters
b) median filters

This research is also applying the filters on FPGA. The results proves high-speed performance of the algorithms that rely on hardware and software compared to software alone, as it appeared very big difference in the speed of execution, depending on the hardware devices that showed the speed of the implementation of the scale of nanosecond, while the software application of algorithms is measured in seconds.

The software was implemented in this research using MATLAB 2010 language code as well as the VHDL language to deal with use of FPGA device, which was of a kind (Xilinx XC3S500E Spartan-3E).

1. Introduction

Image processing is considered to be one of the most rapidly evolving areas of information technology, with growing applications in all fields of knowledge. It constitutes a core area of research within the computer science and engineering disciplines given the interest of potential applications ranging from image enhancing, to automatic image understanding, robotics and computer vision. The performance requirements of image processing applications have continuously increased the demands on computing power, especially when there are real time constraints. Image processing applications may consist of several low level algorithms applied in a processing chain to a stream of input images. In order to accelerate image processing, there are different alternatives ranging from parallel computers to specialized Application Specific Integrated Circuits (ASIC) architectures. The computing paradigm using reconfigurable architectures based on Field Programmable Gate Arrays (FPGAs) promises an intermediate trade-off between flexibility and performance [1].

Various techniques have been developed in Image Processing during the last four to five decades. Most of the techniques are developed for enhancing images obtained from unmanned spacecrafts, space probes and military reconnaissance flights. Image Processing systems are

becoming popular due to easy availability of powerful personnel computers, large size memory devices, graphics software etc. [2].

Filtering is fundamental operation in image processing, it can be used for image enhancement, noise reduction, edge detection, and sharpening, the concept of filtering has been applied in the frequency domain, where it rejects some frequency components while accepting others. In the spatial domain, filtering is a pixel neighborhood operation. Commonly used spatial filtering techniques include: (median filtering, average filtering, Gaussian filtering, etc.) [3]. The filtering function sometimes is called filter mask, or filter kernel. They can be broadly classified into two different categories: linear filtering and order-statistic filters. The common elements of a filter are the neighborhood including the pixel itself. Typically the neighborhood is a rectangular of different size, for example 3×3, 5×5… and smaller than the image itself [4][5].

Neighborhood of pixels is also called windowing operators that are use a window to calculate their output. For example, windowing operator may perform an operation like finding the average of all pixels in the neighborhood of a pixel. The pixel around which the window is found is called the origin. Figure 1, shows a 3×3 pixel window and the corresponding origin.

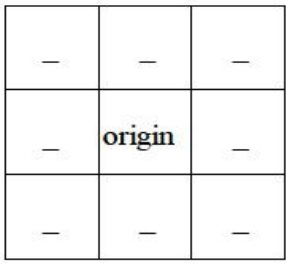


Figure 1; 3×3 Pixel window and origin [6]

The work in this paper is based on the usage of gray scale image smoothing using these pixel windows to calculate their output. Although a pixel window may be of any size and shape, a square 3x3 size was chosen for this application because it is large enough to work properly and small enough to implement efficiently on hardware.

2. Related work

Image smoothing algorithms are particularly suitable for implementation on FPGA, due to the parallelisms that may be exploited. Due to use of microcontroller or microprocessor instruction level parallelism is achieved. Research has been conducted to improve speed by designing system block by block.

Suhaib A. Fahmy suggested An hardware implementation of a median filter and use FIFO buffer to sort the samples for the window over

which the median must be found [7], Anthony Edward Nelson also use FIFO buffer to sort the samples for the window over which the median must be found [6].

In this work pointer is using to reach the positions in RAM instead of using the first in first out implementation (FIFO) which is reduce the complexity of the algorithms implementation, also it reduce the size of the algorithms.

3. The Implementation of the Filters in the Proposed Work:

3.1 Smoothing/Averaging Filters

To smooth an image might do a N×N pixel moving window average of image - e.g. with the 3×3 filter below. Place centre pixel of window over given pixel; multiply pixels of image with pixels in window, sum results and copy as value of output pixel. Then shift window one place to right or down and repeat; the operation is called convolution [8].

The average filtering is also called mean filtering, where the output pixel value is the mean of its neighborhood. Thus, the filtering mask is as show in equation (1) [4]:

$$H(i,j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \ldots\ldots(1)$$

The design of the averaging filter algorithm in VHDL was difficult problem. This was because the averaging filter algorithm uses adders, and dividers to calculate its output. On FPGAs, use of mathematics tends to slow down performance.

Addition is instantiated using simple (+) signs in the VHDL code. The VHDL synthesis tool provides mapping to efficient hardware mathematics designs for it.

Hardware dividers on FPGAs are quite large and slow, we use the bit shifting method of division. Since this is only possible with powers of two, a divide by 8 was implemented instead of a divide by 9, as was planned in the algorithm's design; Figure 2 shows a sketch representation of the mathematics of the hardware averaging filter; wile figure 3 shows the schematic design for the average filtering (the implementation of array, registers, adder and divider); figure 4 shows the simulation time result of average filtering in VHDL, and figure 5 shows the image after applying the average filtering in MATLAB and in VHDL. Finally, figure 6 shows the histogram for image after average filtering in MATLAB, VHDL and the histogram difference between the two images.

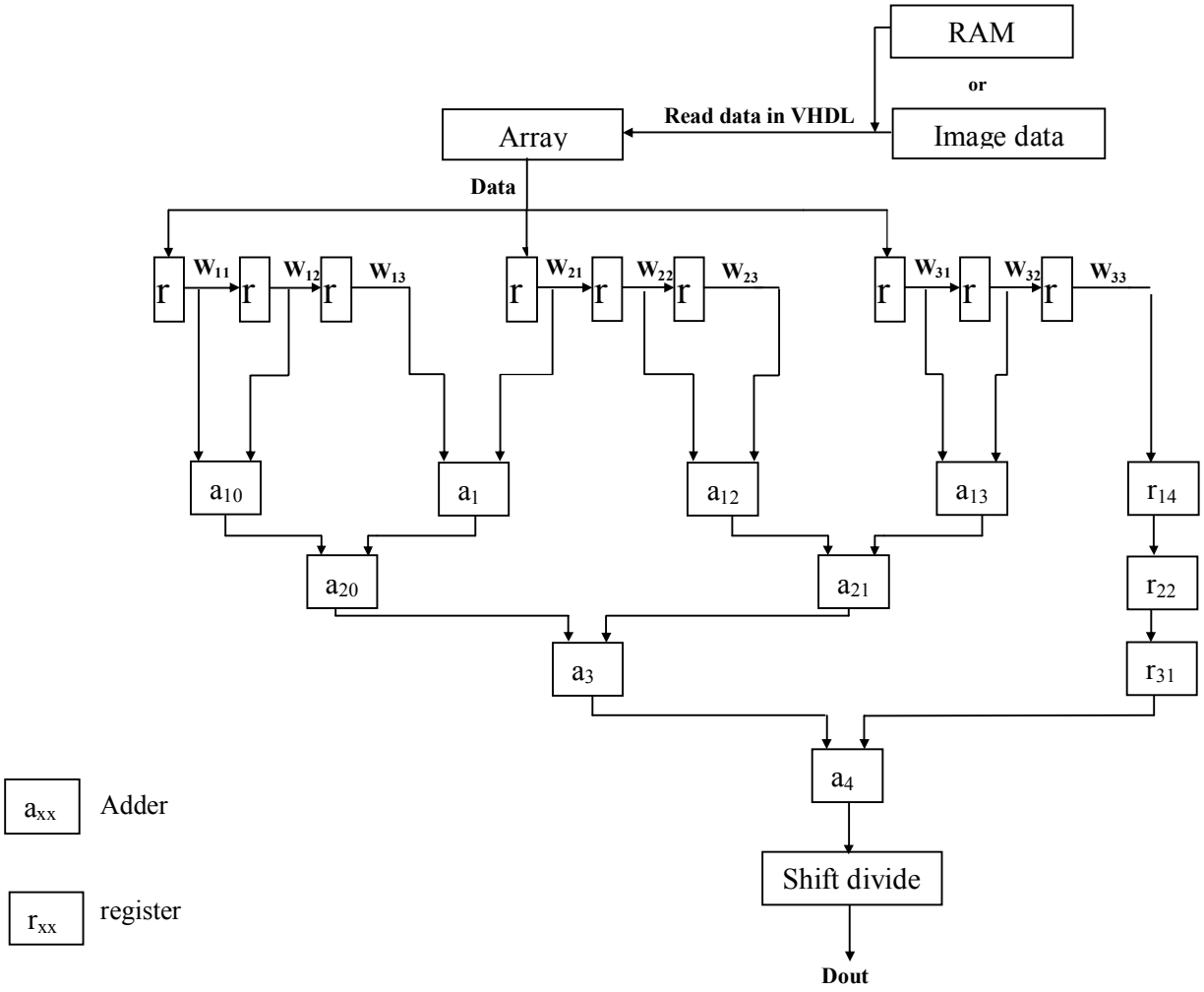


Figure 2: Graphic representation of the averaging filter in VHDL

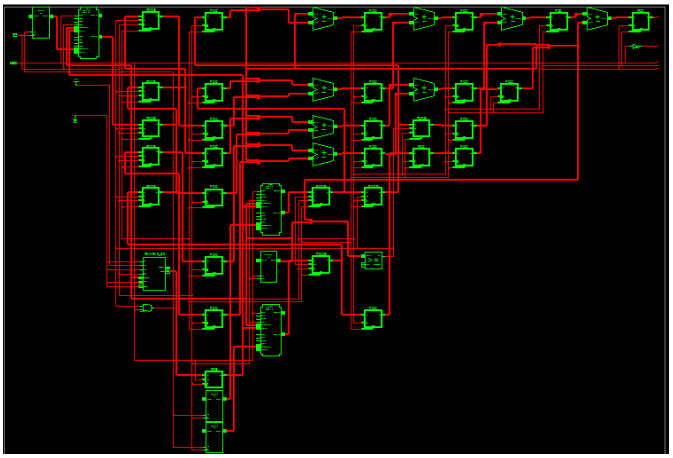


Figure3; The schematic design for the average filtering

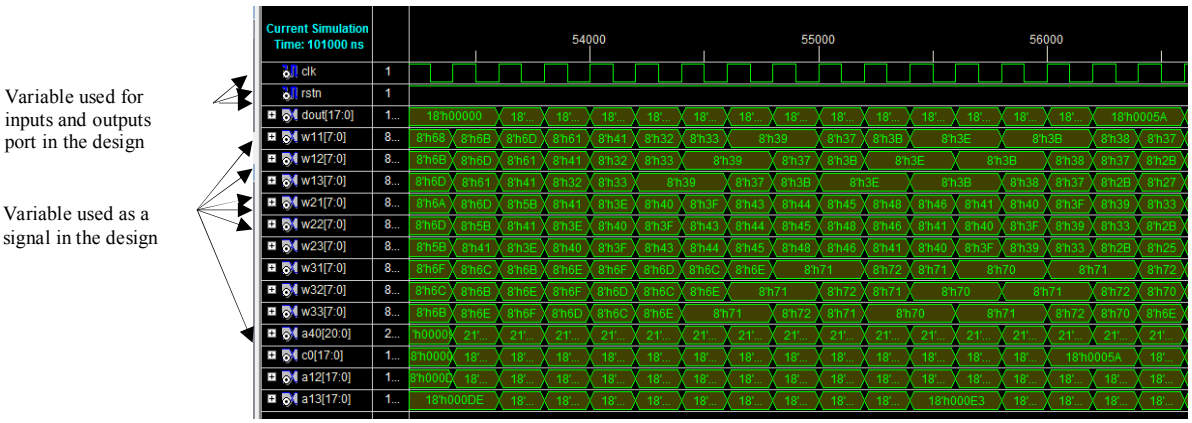


Figure 4; The simulation result of average filtering

a **b** **c**

Figure 5; (a) original images, (b) images after average filtering in MATLAB, (c) images after average filter in VHDL

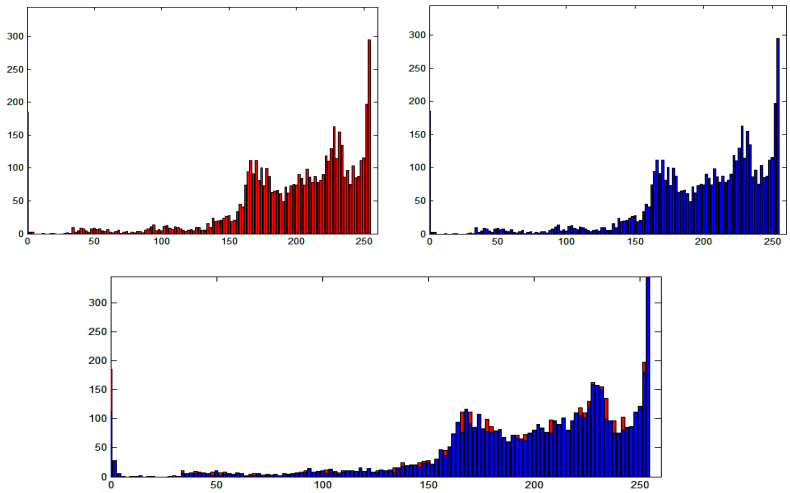


Figure 6; (a) histogram for baby face image after average filtering in MATLAB, (b)histogram for image after average filtering in VHDL, (c)the different between two images.

3.2 Smoothing/Median Filters

A median filter is a non-linear digital filter which is able to preserve sharp signal changes and is very effective in removing impulse noise (or salt and pepper noise) [9].

An impulse noise has a gray level with higher or lower value that is different from the neighborhood point[10]. Linear filters have no ability to remove this type of noise without affecting the distinguishing characteristics of the signal. Median filters have remarkable advantages over linear filters for this particular type of noise [11]. Therefore median filter is very widely used in digital signal and image/video processing applications. A standard median operation is implemented by sliding window of odd size (e.g. 3×3 window) over an image. At each window position the sampled values of signal or image are sorted, and the median value of the samples replaces the sample in the center of the window as shown in Figure 7.

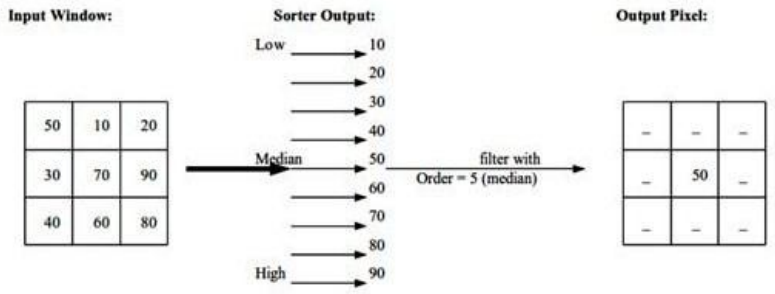


Figure7; Graphic Depiction of median Filter Operation [6]

The structure in Figure 8 represents hardware design for sorting algorithm in VHDL.

Every r_{xx} box is a register and every c_{xx} box is a compare unit, consisting of a simple decision.

This design is accomplished quite simply in VHDL by using the following if/else statement [6]:

if $w_{x1} < w_{x2}$ *then*
$c_{x1}\_L <= w_{x1};$
$c_{x1}\_H <= w_{x2};$
else
$c_{x1}\_L <= w_{x2};$
$c_{x1}\_H <= w_{x1};$
end if;

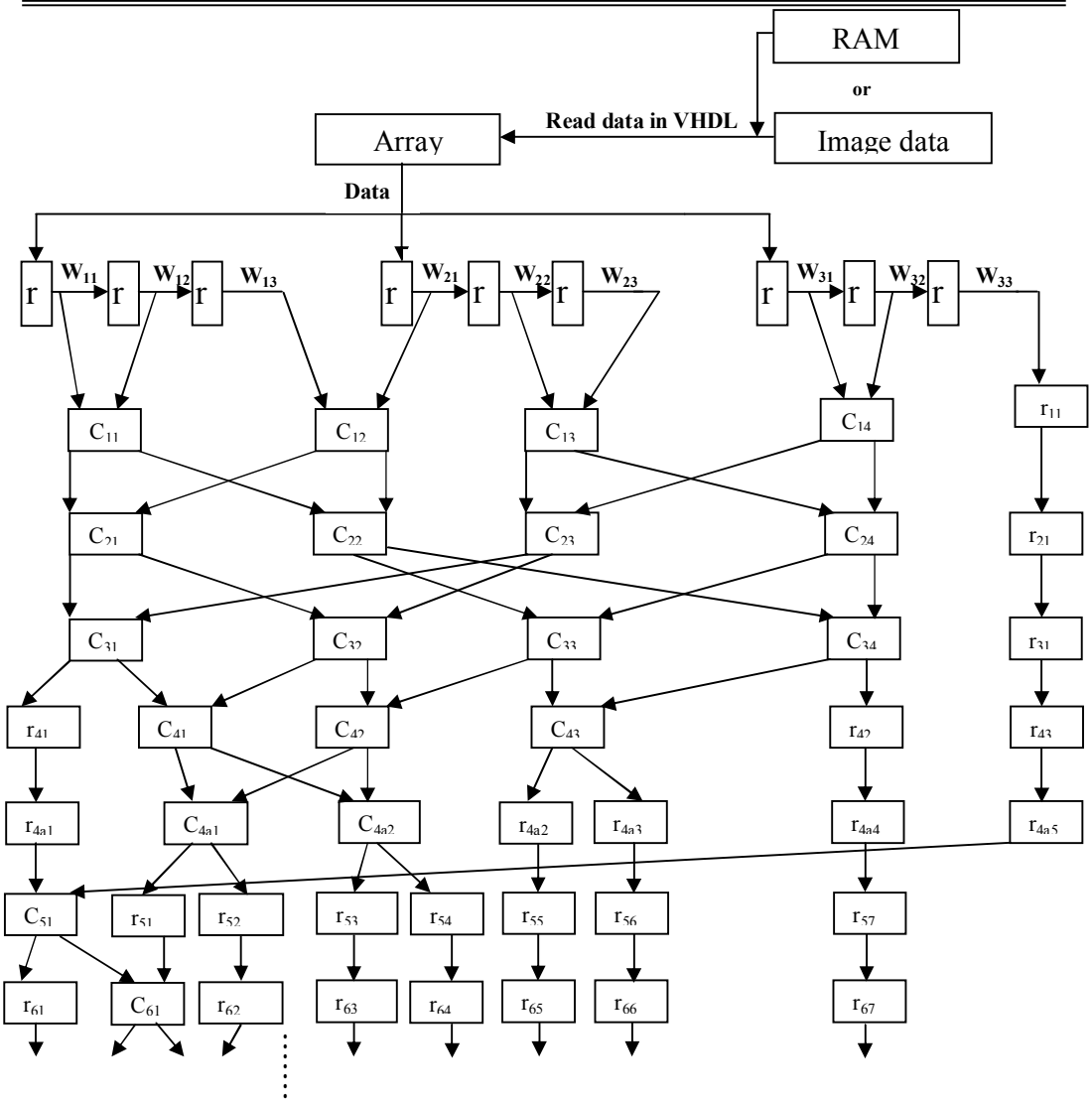


Figure 8; Hardware design for sorting algorithm in VHDL

Figure 9 shows the schematic design for the median filter (the implementation of array, registers, adder and divider).

Figure 9; schematic design for the median filter

Figure 10 shows the simulation time result of median filter in VHDL.

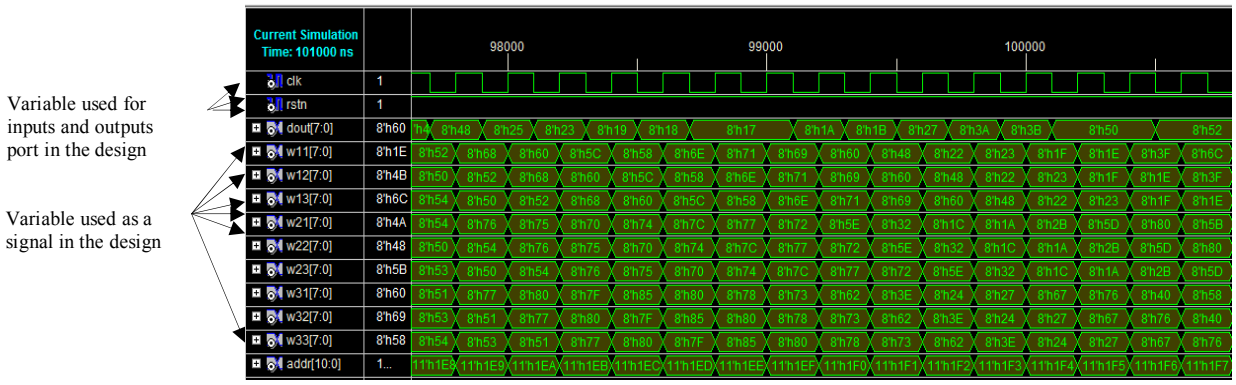
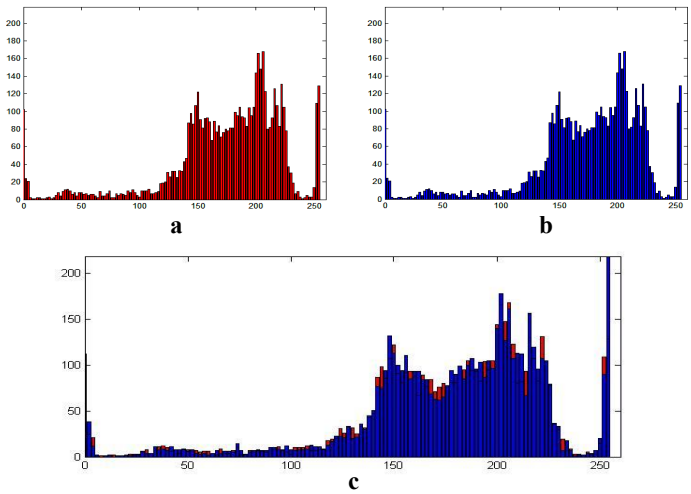


Figure 10; the simulation result of median filter

Figure 11 shows the image after applying the median filter in MATLAB and in VHDL, Figure 12 shows the histogram for image after median filter in MATLAB, VHDL and the histogram difference between the two images.

a **b** **c**

Figure 11; (a) original image, (b) image after median filter in MATLAB, (c) image after median filter in VHDL.



a **b**

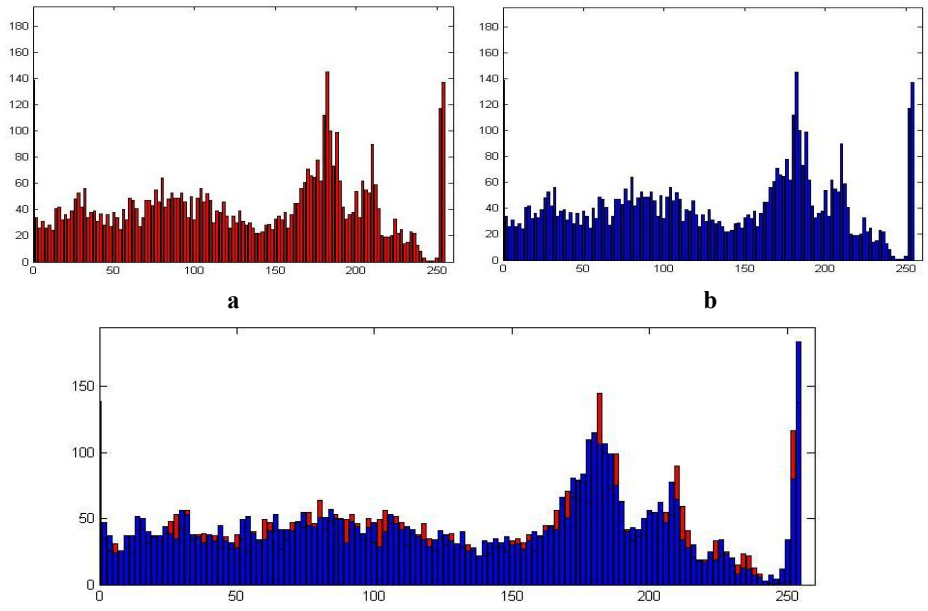
c

Figure 12; (a) histogram for baby face image after median filter in MATLAB, (b)histogram for image after median filter in VHDL, (c)the different between two images.

Figure 13 shows the image after applying the median filter for removing salt and pepper noise in MATLAB and in VHDL, Figure 14 shows the histogram for image after median filter in MATLAB, VHDL and the histogram difference between the two images.

a **b** **c**

Figure 13; (a) image with salt and pepper noise, (b) image after median filter in MATLAB, (c) image after median filter in VHDL.



a **b**

(c)

Figure 14; (a) histogram for baby face image after median filter for removing salt and pepper in MATLAB, (b)histogram for image after median filter in VHDL, (c)the different between two images.

4. Conclusion and Result Discussion:

This work presents the implementation of two filters for image smoothing, (average filtering and median filter), on FPGA Xilinx

XC3S500E Spartan-3E, the two filters are implemented using VHDL by using windowing operators that are use a window to calculate the outputs.

The resulted image gets from VHDL are compared with the results get from MATLAB and found the RMSE, PSNR, MSE, and Correlation between the images, and the results show that the two images are almost the same.

Table 1; The difference value between images that filtered using average filter by MATLAB & VHDL

Image	PSNR	RMSE	Correlation
Face image	68.4212	0.0967	0.9519
Face image1	68.7041	0.0936	0.8839
GIS image	67.2315	0.1109	0.9008
GIS image1	69.4856	0.0856	0.9411
Nature image	67.5590	0.1068	0.9516
Nature image1	69.7484	0.0830	0.9672
Medical image	71.0941	0.0711	0.9801
Medical image	73.5624	0.0535	0.9931

Table 2; The difference value between images that filtered using median filter by MATLAB & VHDL

Filter	Image	PSNR	RMSE	Correlation
Median(smooth)	Face image	67.2398	0.1108	0.9431
Median(smooth)	Face image	67.8161	0.1037	0.8872
Median(salt and pepper noise)	Face image	67.0149	0.1137	0.9400
Median(salt and pepper noise)	Face image	67.8231	0.1036	0.8914
Median(smooth)	GIS image	66.8718	0.1156	0.9157
Median(smooth)	GIS image	68.2591	0.0985	0.9405
Median(salt and pepper noise)	GIS image	69.2993	0.0874	0.9721
Median(salt and pepper noise)	GIS image	68.2391	0.0988	0.9408
Median(smooth)	Nature image	69.3592	0.0868	0.9725
Median(smooth)	Nature image	69.7207	0.0833	0.9715
Median(salt and pepper noise)	Nature image	69.2993	0.0874	0.9721
Median(salt and pepper noise)	Nature image	69.3917	0.0865	0.9696
Median(smooth)	Medical image	68.7867	0.0927	0.9681
Median(smooth)	Medical image	70.1362	0.0794	0.9849
Median(salt and pepper noise)	Medical image	69.2500	0.0879	0.9705
Median(salt and pepper noise)	Medical image	70.1627	0.0791	0.9847

As shown in table 1 and 2, the Peak signal-to-noise ratio (PSNR) is high it generally indicates that the reconstruction is of higher quality, the mean square error (MSE) is small between two images that's mean the best explaining the variability in the observations, and the correlation value is closest to one that's mean there are little different between the images.

The implementation results on FPGA for the two filters are representing in the table 3.

Table 3; The implementation results on FPGA using the two proposed filters

	Average filtering	Median filter
Number of slices	1515 out of 4656 32%	1093 out of 4656 23%
Number of slice flip flops	1054 out of 9312 11%	991 out of 9312 10%
Number of 4 input LUTS No. Used as logic No. Used as rams	7995 out of 9312 85% 1851 6144	4575 out of 9312 49% 1455 3120
Number of bonded IOBS	20 out of 232 8%	10 out of 232 4%
Number of Block RAM	1 out of 20 5%	1 out of 20 5%
Number of GCLKS	1 out of 24 4%	1 out of 24 4%

Many point could be concluded from the proposed work that is:

1. The hardware implementation gives the application higher efficiency and lower time as shown in table (4).

Table 4; the time needs for processing in MATLAB & VHDL

Filter	Time in MATLAB (Sec)	Time in VHDL (ns)
Median	1.3890	7.422
Averaging	0.2810	7.721

2. The resultant images that are obtained from implementing the algorithms on different images (face image, nature, GIS, medical images, text and car numbers) using VHDL was more clear than one that have been processed using MATLAB, because the values in MATLAB may be real numbers (because of division) while in VHDL, the values are integers (the division here is shift right).

3. For high-speed, windowing algorithms are desired, the FPGA technology is ideally suited to the task. In fact, with the aid of the window generator, a whole series of image processing techniques is available to the designer, many of which can be synthesized for high-speed applications.

5. References

[1] Benkrid, K., (2001); "High Level Programming for FPGA based Image and Video Processing using Hardware Skeletons". Proceedings of the Symposium on Field Programmable Custom Computing Machines, pp. 219-226, ISBN: 0-7695-2667-5, April 2001, IEEE Computer Society, Washington, DC.

[2] Mahdi Shaneh, Arash Golibagh Mahyari, (2011); "Image Enhancement using α-Trimmed Mean ε-Filters". World Academy of Science, Engineering and Technology 59 2011.

[3] Gajanand Gupta, (2011); "Algorithm for Image Processing Using Improved Median Filter and Comparison of Mean, Median and Improved Median Filter". International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-1, Issue-5, November 2011.

[4] Huiyu Zhou, Jiahua Wu and Jianguo Zhang, (2010); "Digital Image Processing Part I", ISNB 978-87-7681-541-7.

[5] A. Erhardt – Ferron, (2000); "theory and applications of digital image processing". university of applied sciences Offenburg, copyright 2000 A. Erhardt-Ferron.

[6] Anthony Edward Nelson, (2000); "Implementation Of Image Processing Algorithms On FPGA Hardware". M. Sc. Thesis in Electrical Engineering, Vanderbilt University.

[7] Suhaib A. Fahmy, Peter Y. K. Cheung and Wayne Luk,(2005); "Novel Fpga-Based Implementation Of Median And Weighted Median Filters For Image Processing". 0-7803-9362-7 ©2005 IEEE.

[8] Li Xu, Cewu Lu, Yi Xu, Jiaya Jia, (2011); "Image Smoothing via L0 Gradient Minimization". ACM Transactions on Graphics, Vol. 30, No. 6, Article 174, Publication date: December 2011.

[9] Subhojit Sarker, Shalini Chowdhury, Samanwita Laha, Debika Dey, (2012); "use of non-local means filter to denoise image corrupted by salt and pepper noise". Signal & Image Processing: An International Journal (SIPIJ) Vol.3, No.2, April 2012.

[10] Ery Arias, Castro, David L. Donoho, (2009); "DOES MEDIAN FILTERING TRULY PRESERVE EDGES BETTER THAN LINEAR FILTERING?". The Annals of Statistics 2009, Vol. 37, No. 3, 1172–1206 DOI: 10.1214/08-AOS604, © Institute of Mathematical Statistics, 2009.

[11] ATMEL, (2008); "AVR223: Digital Filters with AVR". Rev. 2527B-AVR-07/08. © 2008 Atmel Corporation.