Efficient Architecture and Implementation of Vector Median Filter in Co-Design Context

Anis BOUDABOUS[1], *Lazhar KHRIJI*[2], *A. BEN ATITALLAH*[1,3], *P. KADIONIK*[3], *Nouri MASMOUDI*[1]

[1] Laboratory of Electronics and Information Technology (LETI), BP W 3038 Sfax - Tunisia
[2] Dept. of Electrical and Computer Engineering, Sultan Qaboos University, Muscat, Oman
[3] IMS Laboratory –ENSEIRB - University Bordeaux I - CNRS UMR 5818, France

anis.boudabous@enis.rnu.tn; lazhar@squ.edu.om; benatita@enseirb.fr; kadionik@enseirb.fr; nouri.masmoudi@enis.rnu.tn

Abstract. *This work presents an efficient fast parallel architecture of the Vector Median Filter (VMF) using combined hardware/software (HW/SW) implementation. The hardware part of the system is implemented using VHDL language, whereas the software part is developed using C/C++ language. The software part of the embedded system uses the NIOS-II softcore processor and the operating system used is μClinux.*

The comparison between the software and HW/SW solutions shows that adding a hardware part in the design attempts to speed up the filtering process compared to the software solution. This efficient embedded system implementation can perform well in several image processing applications.

Keywords

Color image, FPGA, SoPC, NIOS-II, VMF.

1. Introduction

Recently, color image processing has been the subject of extensive research. Filtering is one of the most important elements of color image processing system. Its most important applications are noise removal, image enhancement, and image restoration [2],[8],[19],[20]. A number of sophisticated multichannel filters have been developed to date for image filtering such as order statistics filters [16], polynomial filters and morphological filters [9]. Nonlinear filters applied to images are required to suppress noise while preserving the integrity of edges and detail information. To this end, vector processing of multichannel images, which takes the advantage of color inter-channel dependence and avoids unpleasant drawbacks (pixel value rearranging and chromatic shift), is more appropriate compared to traditional approaches that use component-wise operators, instead [3], [8]. Among them, we cite the vector median filter (VMF) [3], which minimizes the distance in the vector space between the image vectors as an appropriate error criterion. It inherently utilizes the inner correlation between the channels and keeps the desirable properties of the scalar median; namely, the zero impulse response, and the preservation of signal edges. VMFs are derived as the maximum likelihood estimators for an exponential distribution when the filter output is restricted to be one of the input samples. They perform accurately when the noise follows a long-tailed distribution (e.g. exponential or impulsive); moreover, outliers in the image data are easily detected and eliminated by VMF's.

The main contribution of this paper is the implementation of the VMF using embedded system in order to accelerate the execution time of the software solution. Indeed, hardware implementation (HW) is generally better than software implementation (SW) in processing speed and power consumption. First of all the VMF algorithm is coded in ANSI C language. This SW solution was rebuilt and tested using the NIOS-II processor. The execution times have been measured using timer that provides the number of CPU clock cycles. Afterwards, the HW implementation of the SW critical part was done in VHDL (VHSIC Hardware Description Language) language. The main idea of our filter implementation is to exploit the advantages of the parallel structures which can be efficiently implemented in hardware. The interest of parallel architecture is to reduce the number of operations and to reach fast execution. For experimental test and verification, we used the STRATIX Development Board which contains EP1S40F780C5 FPGA device (Field Programmable Gate Array). The reading/writing pixels from images was running on NIOS-II softcore processor embedded in FPGA and using μClinux as operating system. This partitioning has been chosen in order to achieve good timing results.

This paper is structured as follows: section 2 presents an overview of VMF filter. In addition to the description of the HW/SW co-design platform, the parallel structure of the FPGA implementation is discussed in section 3. Section 4 presents the performance evaluations of SW and HW/SW solutions and, also, a comparison between both implementations is reported. Finally, conclusions are drawn in section 5.

2. Overview of VMF Filter

Noise reduction is an important step in many color image processing applications. The most popular nonlinear multichannel filters are based on the ordering of vectors in a predefined filter window. The output of these filters is defined as the lowest ranked vector according to a specific ordering technique [12].

Let $y(I)$ represents a multichannel image and let $W(n)$ be a window of finite size N, where the noisy image vectors inside the window are denoted as $I_j(n)$, j=1,2,…,N and the central sample $I(n)=I_{(N+1)/2}(n)$ determines the position of the filter window. If the Euclidian distance (L_2-norm) between the two vectors I_i, I_j is denoted as $\rho(I_i, I_j) = \|I_i - I_j\|_2 = \left(\sum_{l=1}^{m} |I_i - I_j|^2\right)^{1/2}$ then the scalar quantity $D_i = \sum_{j=1}^{N} \rho(I_i, I_j)$ is the distance associated with the noisy vector I_i in W. An ordering of the D_i's ($D_{(1)} \le D_{(2)} \le \ldots \le D_{(N)}$) implies the same ordering scheme to the input set $W(n)$ resulting in the ordered sequence $I_{(1)} \le I_{(2)} \le \ldots \le I_{(N)}$. Nonlinear ranked type multichannel estimators define the vector $I_{(1)} \in W(n)$ associated with the minimum aggregated distance $D_{(1)} \in \{D_1, D_2, \ldots, D_N\}$ as the filter output, which is called a Vector Median Filter (VMF). The output of the VMF is the pixel $I_k \in W$ for which the following condition is satisfied:

$$\sum_{j=1}^{N} \rho(I_k, I_j) < \sum_{j=1}^{N} \rho(I_i, I_j), \quad i = 1, \cdots, N. \tag{1}$$

Therefore, the VMF consists of computing and comparing the values of D_i inside the sliding filtering window $W(n)$ and the output is the vector I_k for which D_k reaches its minimum.

From the implementation point of view, several VMF filter designs have been developed as either software based applications [9], [10], [11] or hardware based ASIC custom chips [4], [17], or FPGAs [6]. In order to optimize and achieve the best performance a new approach based on a hardware/software (HW/SW) co-design system has been used.

3. Co-Design Implementation

With increasing device densities, audacious challenges become feasible and the integration of embedded SoPC (System on Programmable Chip) systems is significantly improved. FPGA is a general-purpose device filled with digital logic building blocks. The two market leaders in the FPGA industry, Altera and XILINX, are the focus of this study. These two most important manufacturers commercialize softcore processors; XILINX offers MicroBlaze processor [13], also, ALTERA offers NIOS and NIOS-II processors [14]. The benefit of a softcore processor is to add a micro-programmed logic that introduces more flexibility for implementation. A HW/SW co-design approach is then possible, a particular functionality can be developed in software for flexibility, and upgrading completed with hardware IP blocks (Intellectual Property) for cost reduction and performances. FPGA-based platforms are on the rise and penetrating a growing number of application areas. As the penetration of FPGAs for embedded computing converges with the penetration of FPGAs for low-cost, high-volume production applications, we could well see a dramatic shift in the overall market penetration of programmable logic. The reason, behind embedded processor performance in FPGAs is not a big issue, is that performance-critical functions can be accelerated in hardware. Besides the programmability available from the processor or microcontroller, hardware accelerators can be plumbed in, massively parallelizing critical functions. For example, many FPGAs have large numbers of hard-wired multipliers or multiply-accumulate units that can be connected into parallel data paths to make short work of math-intensive routines such as those in many digital signal processing (DSP) algorithms. The challenge in taking advantage of this capability resides in the fact that the design of these hardware accelerators is typically low-level digital hardware work. It requires the expertise in hardware description languages (HDLs), logic synthesis, and complex timing design, simultaneously. The development of the hardware acceleration portion of the design begins to become a bottleneck in the schedule, requiring considerable additional expertise. In fact, the low-cost programmable logic provides another option for systems designers trying to achieve higher performance without taking the drastic action of changing processors or architectures. Compute-intensive functions can be converted into hardware-accelerated functions with programmable logic. From the software perspective point of view, it is easy to make a function call into a custom block of hardware. It can run many times faster than the same code optimized in tight assembly language or algorithms converted into look-up tables. Hardware acceleration means replacing a software algorithm with a hardware module taking advantage of its intrinsic speed. According to the same perspective, interfacing to a hardware-accelerated module is the same as calling a function. The only difference is now the function resides in hardware, which is transparent to the calling function.

The other implementation of a hardware-accelerated module is hardware peripheral. Instead of passing data to a software function, data is written into a memory-mapped hardware peripheral. The computation is done outside the CPU, so the CPU can continue running codes while the peripheral is working. It looks like a normal hardware peripheral replacing a software algorithm.

3.1 NIOS-II Development Board

The NIOS-II development Kit [15], which is used in our work, includes software, hardware, accessories, and documentation to create embedded systems projects. The NIOS development board is shipped with a factory default 32 bits reference design. The core of the board is the ALTERA STRATIX EP1S40F780C5 FPGA. Several peripheral devices and connectors (UART, LCD, VGA and Ethernet...) serve as interfaces between the STRATIX FPGA and the external environment. 8 MByte FLASH, 16 MByte SDRAM and 1 MByte SRAM authorize implementation of complex FPGA image applications. For the embedded system, we have used flash memory, SRAM, SDRAM, UART, timer and Ethernet. ALTERA introduces the SoPC builder tool [18], for quick creation and easy evaluation of embedded systems.

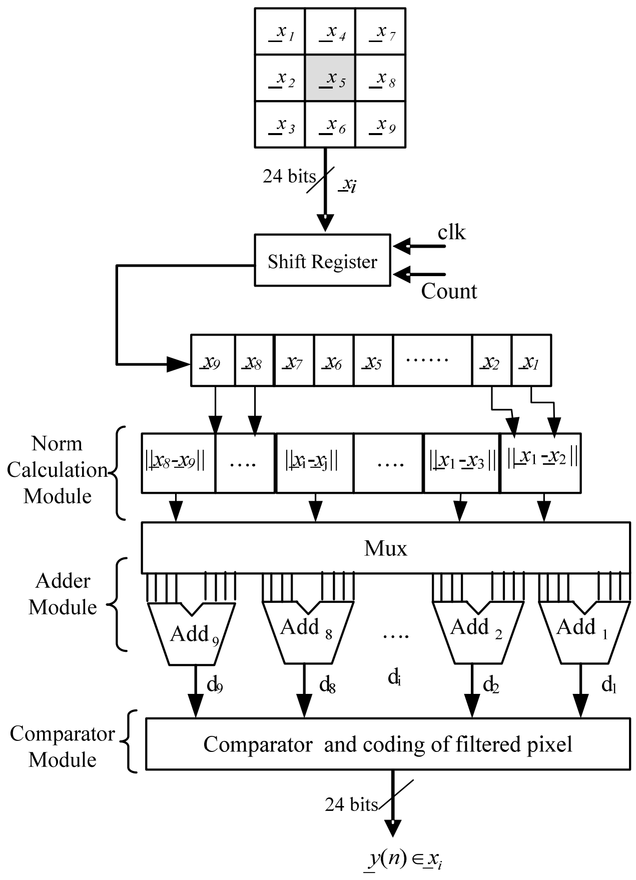
3.2 NIOS-II CPU

For SW implementation of image processing algorithms, the use of a microprocessor is required. The use of additional HW, for optimization, affects the overall performance of the algorithm [1]. For the highest degree of HW/SW integration, a softcore processor was used. The SoPC Builder MegaWizard is used to create NIOS embedded processor systems with 32-bits CPU core, built-in peripherals, on-and off-chip ROM and RAM support and bus support for external hardware modules. The ALTERA NIOS-II softcore processor (Fast version) is a 32-bits scalar RISC with Harvard architecture, 6 pipelined stages, 1-way direct-mapped 64KB data cache, 1-way direct-mapped 64 KB instruction cache and can execute up to 150 MIPS [14]. The main feature of this softcore processor is its extensibility and adaptability. Indeed, users can incorporate custom logic directly into the NIOS-II Arithmetic Logic Unit (ALU) [7]. Furthermore, users can connect into the FPGA the on-chip processor and custom peripherals to a dedicated bus (Avalon Bus). Thus, users can define their instructions and processor peripherals to optimize the system for a specific application.

3.3 Hardware Implementation Part

The entire architecture is organized in linear multi-stage in order to achieve high throughput as shown in Fig. 1. It reflects the sequence of operations in the proposed hardware algorithm which is described in VHDL language. It consists of three main modules, namely, the *norm calculation module* with 36 blocks to calculate simultaneously the norm between two pixels from the filter window, the *adder module* consumes nine adders (with 8 inputs) to perform the distances D_i, the *comparator module* to sort the results of previous module and deliver the system output, i.e. the filtered pixel.

STRATIX FPGA is optimized to improve the performance benefits of SoPC integration based on NIOS-II embedded processor. Also, STRATIX FPGA introduces DSP cores for signal processing applications. These embedded DSP blocks have been optimized to implement several DSP functions with maximum performance and minimum logic resource consumption. The DSP blocks comprise a number of multipliers and adders. They are configurable in various widths to support multiply-add operations ranging from 9x9-bit to 36x36-bit, and including a wide range of operations from multiplication only, to sum of products, and complex arithmetic multiplications. The hardware subsystem contains the hardware function and the interface logic. The latter deals with the communication between the hardware subsystem and the other entities, i.e. software subsystem and the SDRAM memory. However, the controller state machine has to be altered to include the interface logic of both subsystems, through the SDRAM memory. Memory read operations (to obtain the pixels values) as well as memory write operations (to store the filtered pixels values) are embedded in the specification of the hardware subsystem and performed by the interface logic. The interface between the hardware function and the software subsystem uses a control register through which a handshake protocol is implemented. When the software subsystem wants to call the hardware function, it asserts the start bit of control register. When the hardware function completes the execution, it asserts the done bit of control register.



$\underline{x}_1$ $\underline{x}_4$ $\underline{x}_7$
$\underline{x}_2$ $\underline{x}_5$ $\underline{x}_8$
$\underline{x}_3$ $\underline{x}_6$ $\underline{x}_9$
24 bits
$\underline{x}_i$
Shift Register
clk
Count
$\underline{x}_9$ $\underline{x}_8$ $\underline{x}_7$ $\underline{x}_6$ $\underline{x}_5$ …… $\underline{x}_2$ $\underline{x}_1$
Norm Calculation Module
$\|\underline{x}_8-\underline{x}_9\|$ …. $\|\underline{x}_i-\underline{x}_j\|$ …. $\|\underline{x}_1-\underline{x}_3\|$ $\|\underline{x}_1-\underline{x}_2\|$
Mux
Adder Module
Add 9
Add 8
….
Add 2
Add 1
d_9 d_8 d_i d_2 d_1
Comparator Module
Comparator and coding of filtered pixel
24 bits
$\underline{y}(n) \in \underline{x}_i$

Fig. 1. Hardware architecture of VMF filter.

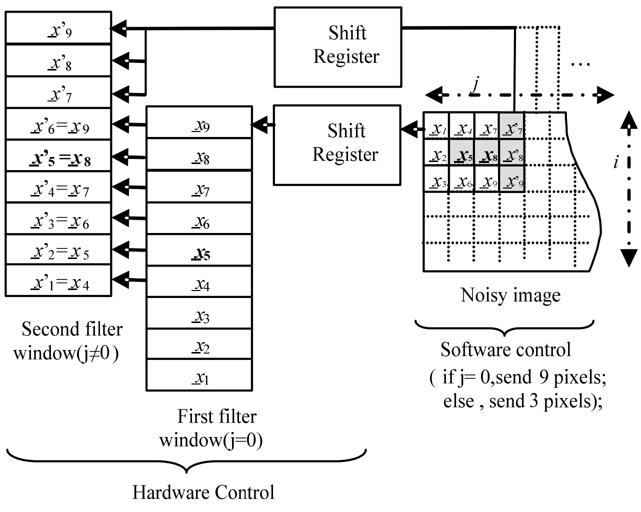
3.4 Software Implementation Part

The software part is used, basically, to optimize the reading of noisy image, the reconstruction of filtered image and the loading pixels to and from hardware core. The used idea of loading pixels is shown in Fig. 2 and described by,

- The first filter window is formed by sending nine pixels to hard core.

- The second filter window is obtained by exploiting the last six pixels from the previous window and sending only three new pixels to hard core.
- The j^{th} filter window is obtained by exploiting the last six pixels from the $(j\text{-}1)^{th}$ window and sending only three new pixels to hard core.

In order to perform the filter computation, the hardware subsystem needs the filter windows, which are sent by the software subsystem. Integer pixels values are passed via memory-mapped registers, while data arrays are stored in the registers. The Algorithm (C language) must be modified as well, so as to include the necessary hardware interaction. After the pixels reading, the hardware subsystem can be started to execute the computation. The software subsystem, then, waits for an interrupt from the hardware subsystem, indicating it has completed the operation.



Shift Register
Shift Register
Noisy image
Second filter window(j≠0)
First filter window(j=0)
Hardware Control
Software control (if j= 0,send 9 pixels; else , send 3 pixels);

Fig. 2. Optimization of the pixels loading.

3.5 HW/SW Components and Design

The whole design is structured in three parts: NIOS-II CPU, VMF hardware part and the Avalon bus and Peripherals. The main processing core of our system is the NIOS-II CPU, which is connected to hardware peripherals via a custom ALTERA's Avalon bus. The latter is configurable bus architecture. It is auto generated to fit the interconnection needs of the designer peripherals. The Avalon bus controls data and address signals and arbitration logic that are connected to the peripheral components. For implementation, we have used NIOS-II softcore processor in a single STRATIX EP1S40F780C5 FPGA device. The functional body of VMF hardware part is shown in Fig. 3. The VMF filter implemented entity is shown in Fig. 4. It presents various inputs/outputs hardware part presented in the previous figure of co-design HW/SW partitioning.

The signals Data_in and Data_out are connected to the Avalon bus. The VMF coprocessor reads/stores the data from/to SDRAM via Avalon bus. The solution of using processor, to move data between SDRAM and VMF coprocessor, is less efficient way. To overcome this point, we used a hardware solution to pass data to the coprocessor, which greatly improved the system performances. Tab. 1 shows the implementation results of the VMF filter in STRATIX EP1S40 FPGA.

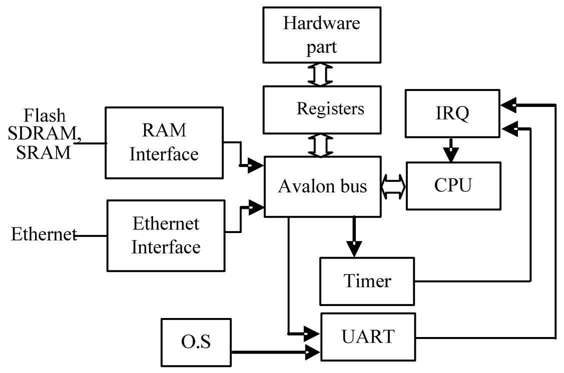


Fig. 3. Co-design HW/SW partitioning.

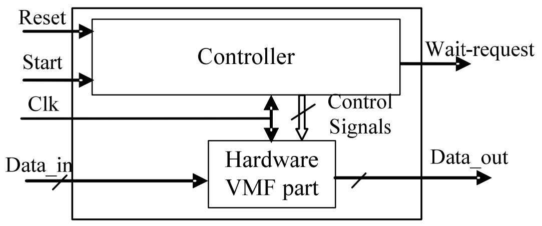


Fig. 4. VHDL entity of Hardware Architecture of VMF filter.

	NIOS-II (FAST)	VMF Hardware part	Whole VMF Design
LEs	12%	11%	30%
ESBs	33%	0%	33 %
DSPs	7%	96%	100%
IOBs	22%	12%	22%
F_{max}(MHz)	150,90	201,68	100

Tab. 1. Synthesis results.

The totality of VMF design requires 30% of Logic Elements (LEs). Indeed, the 12% of the NIOS-II design and the 13% of the hardware part give normally 25%. However, we have an overtaking in the DSPs blocks (because the enormous quantity of calculation and especially of multiplications and adders in excess) which will be implemented on the FPGA target. In addition, the whole VMF design requires 33% of memory blocks "ESBs" (33%+0%), 100% of DSP blocks and 22% of Input/Output blocks (the 12% of VMF hardware will be included in the 22% of the IOBs of the NIOS-II design). The whole design works with 100 MHz system clock. The implementation of filtering process on FPGA enables us to obtain a SoPC system.

4. Performance Evaluation

The HW/SW co-design process uses different kinds of peripherals and memories. The basic idea is to use Linux in an embedded system context. Linux for embedded systems (or embedded Linux) gives us several benefits:

- It is ported to most of processors with or without Memory Management Unit (MMU).
- A Linux port is available for the NIOS-II soft-core.

Most of classical peripherals are ported to Linux. A file system is available for data storage; a network connectivity based on Ethernet protocols is well suited for data recovering.

In this work we use the μClinux as an operating system to control the functionality of the design and extract the filtered image. The results discussed in this section are based on SW and HW/SW implementation of the VMF filter, which is tested on the ALTERA NIOS-II development board. The results illustrate the tradeoffs between process performance and filtering speed. The whole design is described using VHDL (RTL level) fitted into the FPGA. We have determined the filter-processing time before timing's optimization. Both clocks of the processor core and the system are set to 100 MHz, (10 ns period). Tab. 2 shows a comparison of the clock cycles necessary to filter images using software solution (SW) and Hardware/Software (HW/SW) solution. Different tested images (size 176x144x3) are displayed in Fig. 5.

Asides from visually examining the filtered test images, the quality of the filtered images is also measured by the peak-signal-to-noise-ratio metric i.e. PSNR that gives a distinction between two digital images.

$$PSNR = 10 \times \log_{10}\left(\frac{255^2}{\frac{1}{M \times N}\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left\|\underline{o}_{i,j} - \underline{f}_{i,j}\right\|^2}\right) \quad (2)$$

where MxN is the number of vector samples, $\underline{o}_{i,j}$ and $\underline{f}_{i,j}$ are the amplitudes of the original and filtered sample vectors, respectively.

	Images	Noisy	SW	HW/SW
PSNR	Friuts	19.59	26.18	26.18
	Butterfly	17.62	25.99	25.99
	Monkey	15.48	18.17	18.17
	Flower	19.06	32.03	32.03
	Peppers	18.79	27.62	27.62
Number of cycles			109340143	2314822
Total time (ms)			1093.40143	23.14822

Tab. 2. Comparison between SW and HW/SW implementation.

We can note that the HW/SW implementation of the VMF filter improves considerably the filtering speed (48 times faster) compared to the software solution. In addition, our implementation is efficient giving the same PSNR for SW and HW/SW solutions, which demonstrates that the quality of the filtered image is not affected. The whole project was made under μClinux and performed on the Nios-II soft-core processor.

5. Conclusions

In this work, we have described an efficient fast parallel architecture of the VMF filter by means of co-design platform to constitute SoPC system. We have developed a hardware acceleration portion to achieve better filtering speed. Therefore, the actual co-design based implementation constitutes a balance between the two well known requirements: time and area. Our design is working at 100 MHz system clock. The execution time using hardware acceleration is acceptable and goes very well with a number of image processing applications such as video image processing (24 or 30 frames/s). Moreover, better improvement in processing time is possible via different FPGA platforms having higher operating frequency. Finally, we showed that our HW/SW solution improves considerably the filtering speed (48 times faster) compared to software solution.

Acknowledgement

We would like to thank the reviewers and the editor for their valuable and fruitful comments and suggestions which contributed to greatly improve the quality of the original version of this paper.

References

[1] ATITALLAH, A.B., KADIONIK, P., GHOZZI, F., NOUEL, P., MASMOUDI, N., LEVI, H., An FPGA implementation of HW/SW codesign architecture for H.263 video coding. *AEU - International Journal of Electronics and Communications,* Dec 2006.

[2] ASTOLA, J., KUOSMANEN, P. *Fundamentals of Nonlinear Digital Filtering.* Boca Raton–New York: CRC Press, 1997.

[3] ASTOLA, J., HAAVISTO, P., NEUVO, Y. Vector median filter. *Proceedings IEEE*, 1990, vol.78, no.4, pp.678-689.

[4] BERNACCHIA, G., MARSI, S. A. VLSI implementation of a configurable rational filter. In *IEEE International Conference on Consumer Electronics*, ICCE'98, 2-4 June 1998.

[5] CASELLES, V., SAPIRO, G., CHUNG, D. H. Vector Median Filters, inf-sup operations, and coupled PDE's: Theoretical connections. *Journal of Mathematical Imaging and Vision*, 2000, 8, p. 109–119.

[6] CHANUSSOT, J., PAINDAVOINE, M., LAMBERT, P. Real time vector median like filter: FPGA design and application to color image filtering. In *International Conference on Image Processing*. ICIP 99.

[7] CONG, J., FAN, Y., HAN, G., JAGANNATHAN, A., REINMAN, G., ZHANG, Z. Instruction set extension with shadow registers for configurable processors. In *FPGA'05*. Monterey (California, USA), Feb. 20–22, 2005.

[8] KHRIJI, L., GABBOUJ, M. Vector median-rational hybrid filters for multichannel image processing. *IEEE Signal Processing Letters*, July 1999, vol. 6, no. 7, pp.186-190.

[9] KHRIJI, L., GABBOUJ, M. Generalized class of nonlinear-type hybrid filters. *IEEE Electronics Letters*, Dec. 2002. vol.38, no 25, pp. 1650 –1651.

[10] KIM, J., WILLS, D.S. Fast vector median filter implementation using the colour pack instruction set. *IEEE Digital Signal Processing Workshop*, Oct. 2002, pp 339 – 343.

[11] KOSCHAN, A., ABIDI, M. A comparison of median filter techniques for noise removal in color images. In *Proc. of the 7th Germany Workshop on Color Image Processing*. Oct. 2001, vol. 34 no. 15, pp. 69 -79.

[12] LUKAC, R., SMOLKA, B., MARTIN, K., PLATANIOTIS, K.N., VENETSANOPOULOS, A.N. Vector filtering for color imaging. *IEEE Signal Processing Magazine*, January 2005, pp 74-86.

[13] MA, Z., WU, H.R., QIU, B. A robust structure-adaptive hybrid vector filter for color image restoration. *IEEE Transactions on Image Processing*, December 2005, vol. 14, no. 12.

[14] Nios-II Integrated Development Environment http://www.altera.com/literature/lit-index.html.

[15] Nios-II Development Kit, STRATIX Edition, ALTERA 2006, http://www.altera.com/products/devkits/altera/kitniosii 2S30.html.

[16] PITAS, I., VENETSANOPOULOS, A. N. Order statistics in digital image processing. *Proceedings IEEE*, December 1992, vol. 80, no. 12, pp. 1893-1921.

[17] RIZKALLA, M. E., PALANISWAMY, K., SINHA, A. S. C., EL-SHARKAWY, M., SALAMA, P., LYSHEVSKI, S., GUNDRUM, H. ASIC memory design of 2-D median filters. In *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, 2000.

[18] SOPC Builder Applications ALTERA 2006, http://www.altera.com /products/software/products/sopc/sop-index.html

[19] VENETSANOPOULOS, A. N., PLATANIOTIS, K. N. Adaptive filters and multichannel signal processing. In *IEEE Symposium on Advances in Digital Filtering and Signal Processing*, 5-6 June, 1998.

[20] WANG, S., LI, Y., CHUNG FU, L., XU, M. An iterative self-adaptive algorithm to impulse noise filtering for color images. *International Journal of Information Technology,* 2005, vol. 11, no. 10.

About Authors...

Anis BOUDABOUS received electrical engineering degree from the National School of Engineering of Sfax (ENIS) in 2003, and his MS degree in electronic engineering from the University of Sfax in 2004. He is currently researcher in the Laboratory of Electronics and Information Technology (E.N.I.S.) and an assistant at the University of Sfax, Tunisia. His research interests include signal and image processing, nonlinear filtering, hardware implementation using FPGA, embedded systems technology and DSP.

Lazhar KHRIJI received his BS degree in electronics, and his MS and PhD degrees in electrical engineering from the University of Tunis II, Tunisia, in 1990, 1992 and 1999, respectively. In 2002, he received the Doctor of Technology degree in Information Technology from the Signal Processing Institute, Tampere University of Technology, Finland. He is currently Associate Professor at the University of Sousse, Tunisia. From 2002, he is in sabbatical leave with the Sultan Qaboos University, Oman. From 1997 to 1999 he was a research scientist with the Research Institute on Information Technology, Tampere, Finland. His research interests include signal and image processing and analysis, nonlinear filtering, adaptive filtering, image coding, image encryption, genetic algorithms, fuzzy logic, hardware implementation of DSP algorithms.

Nouri MASMOUDI received electrical engineering degree from the Faculté des Sciences et Techniques de Sfax, Tunisia, in 1982, the DEA degree from the Institut National des Sciences Appliquées de Lyon and Université Claude Bernard de Lyon, France in 1982. From 1986 to 1990, he prepared his thesis at the laboratory of Power Electronics (LEP) at the Ecole Nationale d'Ingénieurs de Sfax (ENIS). He received his PhD degree at the Ecole Nationale d'Ingénieurs de Tunis (ENIT), Tunisia in 1990. From 1990 to 2000, he was an assistant professor at the electrical engineering department at ENIS. Since 2000, he has been an associate professor and head of the group 'Circuits and Systems' in the Laboratory of Electronics and Information Technology. Currently, he is responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics: design, telecommunication, embedded systems and information technology.

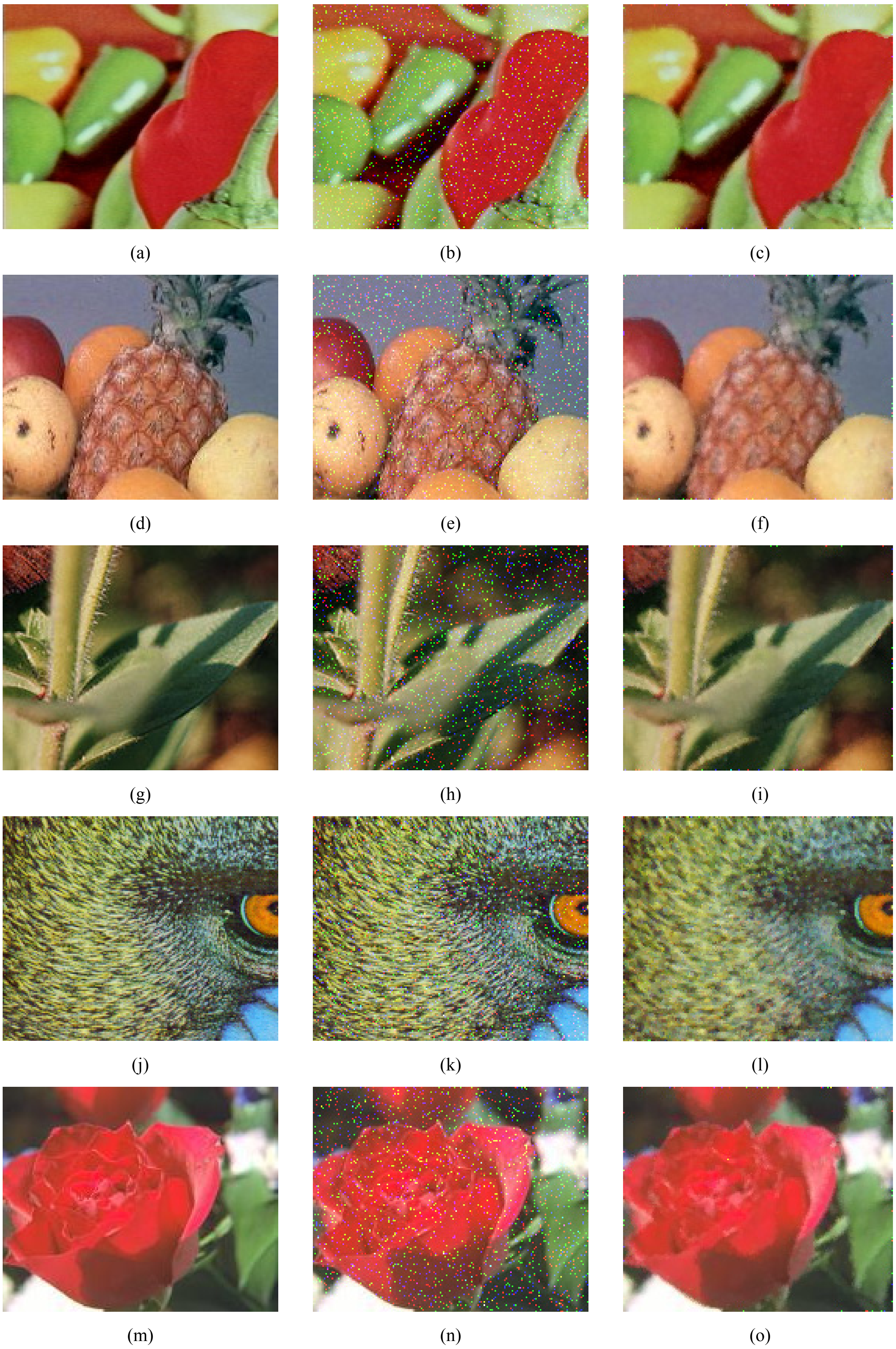


Fig. 3. Images (a), (d), (g), (j) and (m) are the original test images; Images (b), (e), (h), (k) and (n) are 3% Impulsive noisy images; Images (c), (f), (i), (l) and (o) are the filtered images using HW/SW implementation.