An FPGA-based Architecture for Linear and Morphological Image Filtering

Juan Manuel Ramírez, Emmanuel Morales Flores,
Jorge Martínez-Carballido, Rogerio Enriquez
Instituto Nacional de Astrofísica, Optica y Electrónica
Coordinación de Electrónica
Tonantzintla, Puebla, 72000, Mexico

Vicente Alarcón-Aquino, David Báez-López
Universidad de las Américas, Puebla
Departamento de Electrónica, Computación y
Mecatrónica, Cholula, Puebla, 72820
Mexico

***Abstract*—Field Programmable Gate Array (FPGA) technology has become a viable target for the implementation of real time algorithms suited to video image processing applications. The unique architecture of the FPGA has allowed the technology to be used in many applications encompassing all aspects of video image processing. Among those algorithms, linear filtering based on a 2D convolution, and non-linear 2D morphological filters, represent a basic set of image operations for a number of applications. In this work, an implementation of linear and morphological image filtering using a FPGA NexysII, Xilinx, Spartan 3E, with educational purposes, is presented. The system is connected to a USB port of a personal computer, which in that way form a powerful and low-cost design station. The FPGA-based system is accessed through a Matlab graphical user interface, which handles the communication setup. A comparison between results obtained from MATLAB simulations and the described FPGA-based implementation is presented.**

Keywords: Image, processing, FPGA, filtering.

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are part of current reconfigurable computing technology, which in some ways represent an ideal alternative for image and video processing [1]. FPGAs generally consist of a system of logic blocks, such as look up tables, gates, or flip-flops, just to mention a few, and some amount of memory, all wired together using a vast array of interconnects. All of the logic in an FPGA can be rewired, or reconfigured, with a different design, according to the designer needs. FPGAs generally consist of a system of logic blocks (usually look up tables and flip-flops) and some amount of Random Access Memory (RAM), all wired together using a vast array of interconnects. This type of architecture allows a large variety of logic designs for a number of real time applications [2,3,4,5]. There is currently a vast amount of software for digital image processing applications, for industrial or educational purposes. Among these, MATLAB has been extensively used becoming a standard mathematical language for engineers and scientists around the world [6,7,8]. MATLAB is a software environment that allows problems and solutions to be expressed in familiar mathematical notations. Numerous toolboxes and other software packages have been developed for MATLAB to facilitate a variety of engineering and educational tasks such as algorithm development, modeling, simulation, data analysis, visualization, engineering graphics, and application development. Simulink is a software package that works in conjunction with MATLAB for modeling, simulating, and analyzing dynamic systems through an intuitive block diagram-based GUI that utilizes various block-set libraries to incorporate preconfigured blocks and connectors by simple drag and drop operations. MATLAB can be used also for generating hardware description language in order to synthesize FPGA-based designs [9]. In this paper, a Nexis II system based on the Xilinx FPGA Spartan 3E has been used to implement a low-cost image processing system for real time applications with educational purposes. A MATLAB graphical user interface allows the designer to open the image to be processed, setup the communication parameters, specify the required processing, send the input image, and receive the corresponding result after the process.

A. Space-domain linear filtering based on a 2D convolution.

In a linear system, the output is obtained through the convolution of the input function with the system impulse response. In digital image processing, the impulse response is defined as a convolution mask or kernel with a size typically much less than the size of the input image. The window size of the convolution mask is related to the order of the system. 2D convolution is expressed as [10]:

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x[j,k]\,h[m-j,n-k] \qquad (1)$$

If the convolution mask is given by the following 3X3 matrix:

$$h = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \qquad (2)$$

The output image is then obtained as:

$$\begin{aligned} y[m,n] = {} & h_{11}x(m-1,n-1) + h_{12}x(m-1,n) + h_{13}x(m-1,n+1) + \\ & h_{21}x(m,n-1) + h_{22}x(m,n) + h_{23}x(m,n+1) + \\ & h_{31}x(m+1,n-1) + h_{32}x(m+1,n) + h_{33}x(m+1,n+1) \end{aligned} \qquad (3)$$

According to the definition, the operations involved in a 2D convolution can be easily performed by addition and multiplication of the neighborhood inside of the convolution mask for each pixel in the input image [11].

B. *Non-linear image filtering; morphology.*

Mathematical morphology is a geometric approach to non linear image processing that was developed as a powerful tool for shape analysis in binary and grayscale images [12,13]. Morphological operators are defined as combinations of basic numerical operations taking place over an image A and a small object B, called a *structuring element.* B can be seen as a probe that scans the image and modifies it according to some specified rule. The shape and size of B, which is typically much smaller than image A, in conjunction with the specified rule, define the characteristics of the performed process. Binary mathematical morphology is based on two basic operators: *Dilation, and erosion.* Both are defined in terms of the interaction of the original image *A* to be processed, and the structuring element *B*. Morphological dilation is defined as the set union of the objects *A* obtained after the translation of the original image for each coordinate pixel *b* in the structuring element B.

$$A \oplus B = \bigcup_{b \in B} T_b(A) \qquad (4)$$

The erosion is the morphological dual of the dilation. It is defined in terms of set intersections as :

$$A \Theta B = \bigcap_{b \in B} T_{-b}(A) \qquad (5)$$

A very important characteristic of morphological image processing comes from the fact that the basic described operators, performed in different orders, and following different heuristics, can yield to many useful results. For example, if the output of an erosion operation is dilated, the resulting operation is called an opening. The dual of opening, called closing, is a dilation followed by an erosion. These two secondary morphological operations can be useful in image restoration, and their iterative use can yield further interesting results, such as skeletonization and granulometries of an input image, for pattern recognition applications, among others. The morphological operations can be performed using Boolean logic in connection with delay lines at pixel level, using an adequate memory organization for the input image and the structuring element [12]. The system used in this work consists of a development platform based on the Xilinx FPGA Spartan 3E. The system includes a high speed USB2 port, 16 Mb of RAM and ROM memory, input and output ports, and support for embedded processors based on the Xilinx MicroBlaze system. The card connected to a personal computer conform a powerful design station. In each step it is required to evaluate the neighborhood of each pixel according to the specific operation. This decision can be performed by checking if the pixel under test can be induced by a translational displacement of pixels from the original image, while only displacements defined by the structuring element are considered. In other words: the structuring element is seen as a set of translation vectors, given by its pixel coordinates. A pixel in the result image can be obtained by a translational displacement of an original image pixel by the vector forming the structuring element. Accordingly, the erosion and dilation operations are carried out on a set of pixel pairs obtained by vector addition in the neighborhood of some pixel in the input image, and the vector positions in the structuring element, through Boolean operations.

II. Hardware Description

The kit NexisII is a development platform based on the Xilinx FPGA Spartan 3E with 500 000 gates. The kit has integrated a high speed USB2 port, 16 Megabytes of RAM and ROM memory, input and output ports, and several expansion connectors, which make it an ideal platform for digital systems design, including support for embedded processors based on the Xilinx MicroBlaze system. Fig. 1 shows the Xilinx Spartan 3E block diagram.

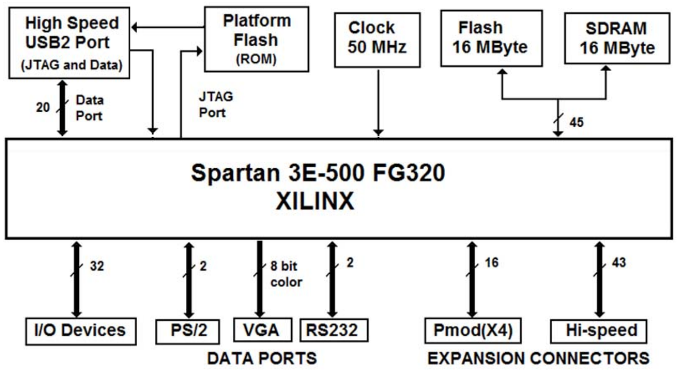
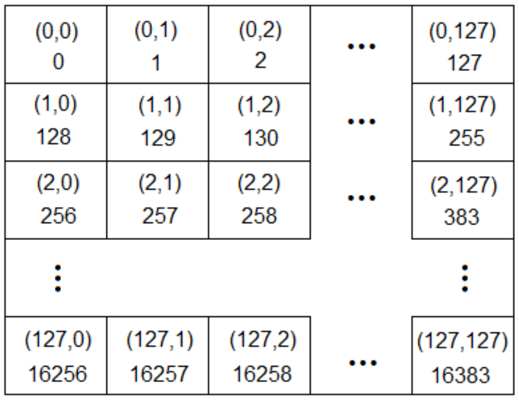


Fig. 1 Xilinx Spartan 3E block diagram.

The kit is connected to a PC through the USB2 port. A MATLAB interface allows the user to open the image to be processed, setup the communication parameters, specify the required processing, send the input image, and receive the corresponding result after the process. The operations are performed on a 128X128 input binary image, organized in a linear form as shown in Fig. 2.



(0,0) 0	(0,1) 1	(0,2) 2	•••	(0,127) 127
(1,0) 128	(1,1) 129	(1,2) 130	•••	(1,127) 255
(2,0) 256	(2,1) 257	(2,2) 258	•••	(2,127) 383
⋮				⋮
(127,0) 16256	(127,1) 16257	(127,2) 16258	•••	(127,127) 16383

Fig. 2. Pixel distribution of the input image

The module *Rx_address* shown in Fig. 3 is composed of a serial receiver module, which receives the input data in serial form, shows the parallel output result, and generates the memory address where the data is temporarily stored. The output is connected to the following module called *pruebax2*. When the total number of the image elements has been received, a signal enables a buffer and put the address output in high impedance state, which allow the memory to be read using the addresses generated by the module *control_submatrix* shown in Fig. 4.

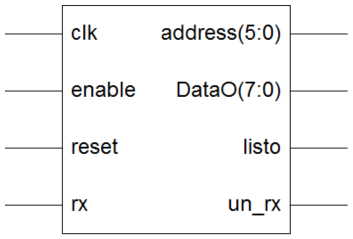


Fig. 3. Module *Rx_address*.

The block described allows generation of the memory addresses required to perform the convolution. The module *control_submatrix* obtains the memory indexes where the image is stored. These indexes are automatically generated when the *enable* signal is activated. The signal *bloque* is used to indicate the whole block required to process the neighborhood around the pixel and inside the convolution mask, while *habilitador* is a control signal which indicates when a memory data is ready. The signal address provides the memory address to be read, while the pair *fila-column* specify the pixel position in process.

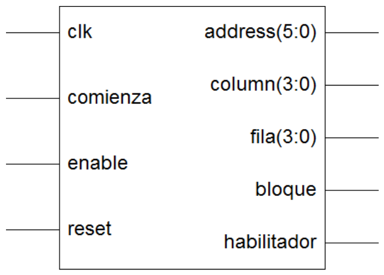


Fig. 4. Module *control_submatrix*.

Once the memory addresses have been read, the value in each memory location is temporarily stored in order to perform the required multiplications with the values in the convolution mask. This process is done in the unit *SIPO,* shown in Fig. 5. This unit receives the data, and performs a serial to parallel conversion, locating the values in the *dataN* terminals, and generating a signal to indicate that the data is ready. The data is entered to the next unit called *zero_padding* shown in Fig. 6, which selects the operation to be done: Linear 2D convolution or morphological filtering. This unit is also in charge of fill with zeroes the pixel positions wherever is required, such as the values in the border of the input image.

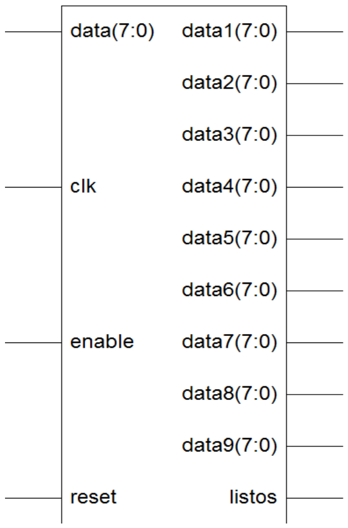


Fig. 5. Module *SIPO*.

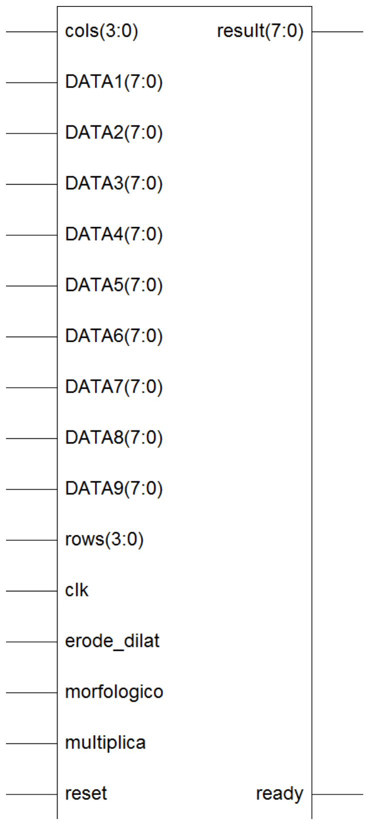


Fig. 6. Module *zero_padding.*

The module *pruebax2* represented in Fig. 7 is formed by the units *RAM1, control_submatrix, SIPO* and *zero_padding*. When an enable signal is entered, the following process is carried out: The corresponding memory addresses are generated, data is temporarely stored in the SIPO register, the module zero_padding is enabled to perform the operations, the result is presented in the output terminals, and a signal indicating that the data has been processed is generated. This result is stored in memory. The module *count_mem_tx* shown in Fig. 8 has a counter which is increased every time that the module pruebax2 complete its routine. Once the input image has been totally processed, the signal *todos_guardados* is enabled and the system is ready to send back the processed data to the PC. The result is visualized through the MATLAB interface.

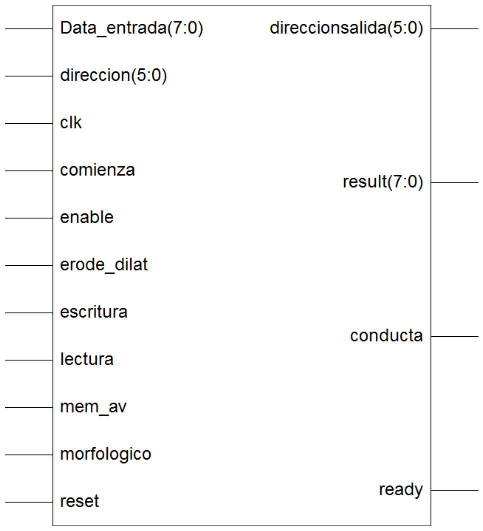


Fig. 7. Module *pruebax2.*

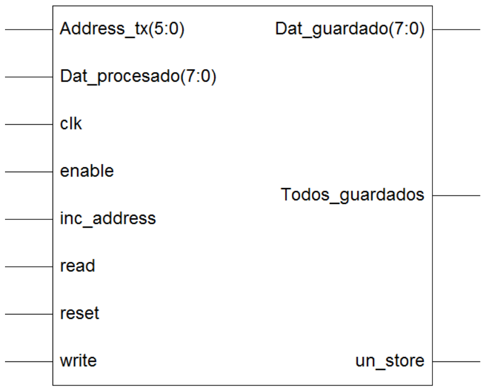


Fig. 8. Memory module *count_mem_tx.*

This operation is implemented through the unit *Mem_tx* represented in Fig. 9, which receives an enable signal at the terminal *envía_regreso*, and it generates the memory address where the processed image has been stored. The data coming from the memory module *count_mem_tx* is consecutively located in the input *DATA* and serially send it back to the PC. The processed image is visualized through the MATLAB interface.

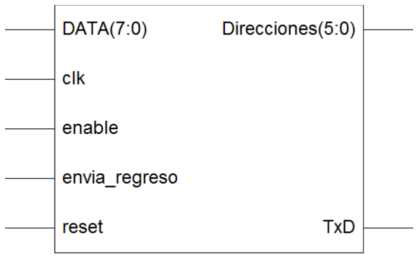


Fig. 9. Module *Mem_tx*.

The master unit called *TOP-TOP1* which englobe the modules previously described, is shown in Fig. 10.

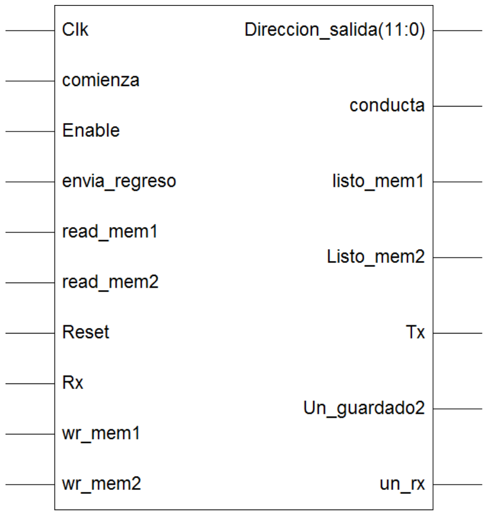


Fig. 10. Master unit *TOP-TOP1*.

III. MATLAB GRAPHICAL USER INTERFACE

Fig. 11 shows the MATLAB graphical user interface designed for this system. The interface allows the user to select the serial port to be used, specify a baud rate, open, and close the serial port, and select the image to be processed in jpg format. According to the type of processing specified by the user, a 3X3 pixels operational mask is defined. If a convolution linear filter is selected, the coefficients define the system impulse response. If a morphological filter is required, the coefficients define form and size of the structuring element. Once the operational mask coefficients are entered, the system is ready to send the information to the FPGA in order to perform the required processing, obtain the output image, and send it back to the local computer, displaying the result through the graphical user interface. Fig. 11 shows the user interface when a Laplacian-based sharpening filter is applied to the image *´lenna´*. Fig. 12 shows the result obtained with a dilation morphological filter applied to the binarized image *´flower´*.

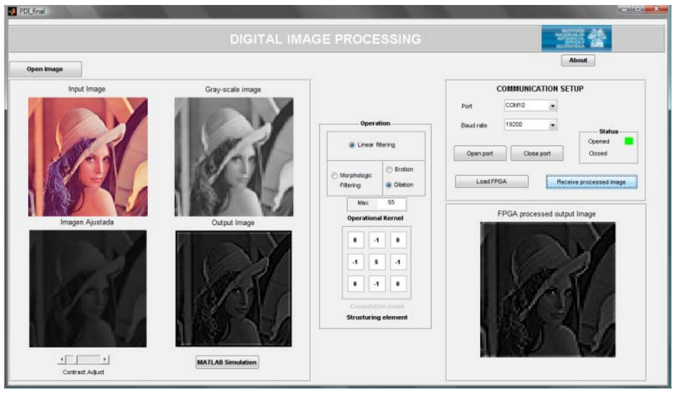


Fig. 11. Graphical user interface main screen; sharpening filter applied to the image *´lenna´*.

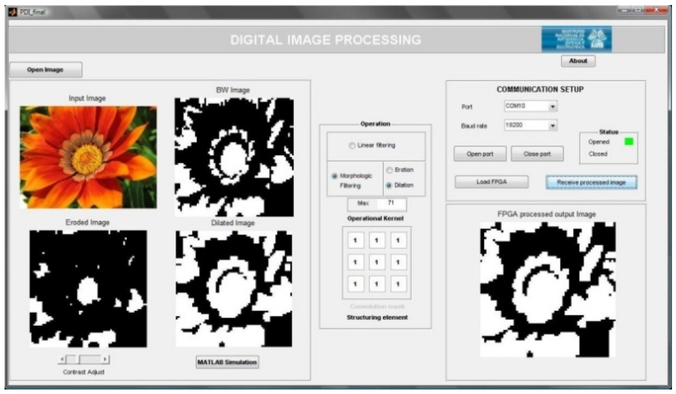


Fig. 12. Binarization and morphological dilation applied to the image *´flower´*.

IV. RESULTS

Fig. 13 shows the time signals involved in the simulation of the module *conv2,* which performs the processing in the neighborhood of the central pixel with the corresponding element in the operational mask. The process starts whit a pulse at the signal *comienza*. During the following nine clock cycles the nine pixels values of the input image and the corresponding nine values in the operational mask are read. During the tenth clock cycle a signal which indicates to perform the multiplication is sent. An eleventh clock cycle is required in order to write the result obtained from the multiplication. In a 64X64 image we have 4096 pixels, so a total number of 4096X11 clock cycles are required during the process. The system is operating at a clock frequency of 10 MHz, which gives an approximate execution time of 4.5 mS to process the image. Table I shows a summary of the resources used in the FPGA.

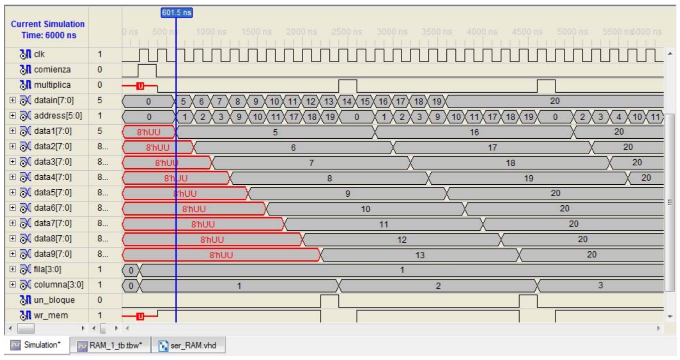


Fig. 13. Time signals in the module *conv2.*

Comienza: Inputs signal; it starts and controls the processing of the image.
Datain: Input signal which receives the memory value to form the 3X3 matrix.
Data1-data9: Pixel values from the 3X3 operational mask.
Address: It indicates the next memory address to be read.
Multiplica: Output signal used to indicate that the total number of elements involved in the operation has been processed.

Table I. Device utilization summary

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	332	9 312	3 %
Number of 4-input LUTs	952	9312	10 %
Logic Distribution			
Number of occupied Slices	573	4 656	12 %
Number of Slices containing only related logic	573	573	100 %
Number of Slices containing unrelated logic	0	573	0 %
4-input LUTs			
Number used as logic	952		
Number used as route-thru	43		
Number of bonded IOBs	30	232	12 %
Number of RAMB 16s	4	20	20 %
Number of BUFGMUXs	4	24	16 %

V. CONCLUSIONS

A low-cost image processing system for real time applications with educational purposes has been presented. The system takes advantages of the available resources in a Nexis II system based on the Xilinx FPGA Spartan 3E. A MATLAB graphical user interface allows the designer to operate the system in a friendly way, through the basic required operations such as image manipulation, and setup of the communication parameters. The described FPGA-based real-time image processing system was shown to provide a very good tool for further computer vision applications. At the same time, it is worthwhile to mention the educational value of the developed prototype as a laboratory tool in modern digital system courses.

VI. ACKNOWLEDGEMENT

The second author would like to thank CONACYT Mexico for the scholarship granted to do master studies.

REFERENCES

[1] C.T. Johnston, K.T.Gribbon, D.G.Bailey, "Implementing Image Processing Algorithms on FPGAs", Eleventh Electronics New Zealand Conference, Palmerston North, New Zealand, 2004.

[2] D.G. Bariamis, D.K. Iakovidis, D.E. Maroulis, S. A. Karkanis, "An FPGA-based Architecture for Real Time Image Feature Extraction", Proceedings of the 17th International Conference on Pattern Recognition, August 23-26, Cambridge, UK, 2004.

[3] Bruce A. Draper, J. Ross Beveridge, A.P. Willem Böhm, Charles Ross, Monica Chawathe, "Accelerated Image Processing on FPGAs", IEEE Transactions on Image Processing, Vol. 12, No. 12. Pp. 1543-1551, 2003.

[4] A. Castillo, J. Vázquez, J. Ortegón y C. Rodríguez, "Prácticas de laboratorio para estudiantes de ingeniería con FPGA", IEEE Latin America Transactions, Vol. 6, No.2, pp. 130-136, 2008.

[5] K. T. Gribbon, D. G. Bailey and C. T. Johnston, "Design Patterns for Image Processing Algorithm Development on FPGAs", TENCON 2005, pp. 1-6, November 21-24, 2005.

[6] Bob L. Sturm and Jerry D. Gibson, "Signals and Systems Using MATLAB: An Integrated Suite of Applications for Exploring and Teaching Media Signal Processing", 35th ASEE/IEEE Frontiers in Education Conference, pp. 21-25, October 19 – 22, Indianapolis, Indiana, USA, 2005.

[7] Javier Vicente, Begoña García, Ibon Ruiz, Amaia Méndez, Oscar Lage, "EasySP: Nueva Aplicación Para la Enseñanza de Procesado de Señal", IEEE-RITA, Vol. 2, No. 1, 2007.

[8] David Báez-López, David Báez-Villegas, René Alcántara, Juan José Romero, Tomás Escalante, "A package for filter design based on MATLAB", Computer Applications in Engineering Education, Vol. 9, No. 4, pp. 259-264, 2002.

[9] Malay Haldar, Anshuman Nayak, Alok Choudhary, Prith Banerjee, **"**A system for synthesizing optimized FPGA hardware from MATLAB", International Conference on Computer Aided Design, pp. 314-319, San Jose, California, 2001.

[10] Rafael C. Gonzales, Richard E. Woods, Digital Image Processing using Matlab, Prentice Hall, 2002.

[11] Tanvir A. Abbasi, Mohd U. Abbasi,"A Proposed FPGA Based Architecture for Sobel Edge Detection Operator", *J. of Active and Passive Electronic Devices,* Vol. 2, pp. 271–277, 2007.

[12] A. Pitas and A. N. Venetsanopoulus, *Non-linear Digital Filters; Principles and Applications*, Kluwer Academic Publisher, 1990.

[13] A. Ledda and W. Phillips, "Majority Ordering and the Morphological Pattern Spectrum**",** *Conf. Proc. Advanced Concepts for Intelligent Vision Systems*, Antwerp, Belgium, 356-363, 2005.