FPGA-based educational platform for real-time image processing experiments

ARTICLE *in* COMPUTER APPLICATIONS IN ENGINEERING EDUCATION · MARCH 2013

Impact Factor: 0.45 · DOI: 10.1002/cae.20461

Juan Manuel Ramirez-Cortes
Instituto Nacional de Astrofísica, Óptica y Ele…
57 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Pilar Gomez-Gil
Instituto Nacional de Astrofísica, Óptica y Ele…
60 PUBLICATIONS **80** CITATIONS
SEE PROFILE

Vicente Alarcon-Aquino
Universidad de las Americas Puebla
95 PUBLICATIONS **252** CITATIONS
SEE PROFILE

FPGA-Based Educational Platform for Real-Time Image Processing Experiments

JUAN MANUEL RAMIREZ-CORTES,[1] PILAR GOMEZ-GIL,[2] VICENTE ALARCON-AQUINO,[3] JORGE MARTINEZ-CARBALLIDO,[1] EMMANUEL MORALES-FLORES[1]

[1]*Department of Electronics, National Institute of Astrophysics, Optics, and Electronics, Tonantzintla, Puebla, Mexico*

[2]*Department of Computer Science, National Institute of Astrophysics, Optics, and Electronics, Tonantzintla, Puebla, Mexico*

[3]*Department of Electronics, University of the Americas, Cholula, Puebla, Mexico*

Received 2 December 2009; accepted 3 May 2010

ABSTRACT: In this paper, an implementation of an educational platform for real-time linear and morphological image filtering using a FPGA NexysII, Xilinx®, Spartan 3E, is described. The system is connected to a USB port of a personal computer, which in that way form a powerful and low-cost design station for educational purposes. The FPGA-based system is accessed through a MATLAB graphical user interface, which handles the communication setup and data transfer. The system allows the students to perform comparisons between results obtained from MATLAB simulations and FPGA-based real-time processing. Concluding remarks derived from course evaluations and lab reports are presented. © 2010 Wiley Periodicals, Inc. Comput Appl Eng Educ; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20461

Keywords: image; processing; hardware; education; filtering

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. FPGAs generally consist of a system of logic blocks, such as look up tables, gates, flip-flops, and some other resources, all wired together using a vast array of interconnections forming a switching network. All of the logic in an FPGA can be rewired, or reconfigured with a different design, according to the designer needs. This characteristic allows application specific hardware to be constructed, while maintaining the ability to change the functionality of the reprogrammable system with ease [1–4]. As such, an FPGA offers a compromise between the flexibility of general purpose processors and the hardware based speed of application-specific integrated circuits (ASICs). The logic blocks can be configured so as to exploit data, pipeline process, I/O parallelism, or all of the above. This type of architecture allows a large variety of logic designs for a number of real-time applications. In computer vision and image processing applications, for instance, FPGAs can be used to do real-time object tracking [5], stereo analysis [6], color-based segmentation [7], spectral analysis [8], or video and image compression [9], just to mention a few. FPGAs are generally programmed in hardware design languages, such as VHDL, which requires digital design and programming skills from the developer. Hardware synthesis from software languages is also achieved through many available compilers such as C, C++, Handel C, FORTRAN, Occam, and others. For science professionals, who may not necessarily be proficient in traditional programming languages, algorithm development might be more tractable in higher level tools such as MATLAB and Simulink from MathWorks. However, the high level of abstraction in the MATLAB language adds complexity to the compilation. MATCH and Compaan/Laura are examples of hardware synthesis from MATLAB code [10–12]. In a digital image processing course, the student is usually urged to experiment with the algorithms and theoretical concepts, and experimental software and hardware platforms are required, so the student can concentrate in the algorithms and their effect on test images. It then becomes very interesting to establish the proper balance between the presentation of theoretical concepts and their implementation, either in software or hardware, on modern courses of image processing, digital systems and reconfigurable computing [13,14].

Spatial domain image processing methods are procedures that operate directly on the pixel values of the input image. The operation can be performed on a single pixel or a collection of pixels in the neighborhood. In the first case, each output pixel depends

Correspondence to J.M. Ramirez-Cortes (jmram@inaoep.mx).

only upon the value of the corresponding input pixel and some transformation function. In the second case, each output pixel is obtained through operations performed upon the values of the image pixels in some neighborhood and the corresponding values in some operational mask. Hardware implementation of point to point transformations can be achieved by simply passing the image through a hardware block designed to perform the required operation. Complex point transformations, such as piece-wise functions, can be implemented using look-up-tables. Hardware implementation of linear filters based on 2D convolution requires the buffering of sub-images in the neighborhood corresponding to the operational mask, or the whole input image, according to the design. The buffered values are used to perform the intermediate operations based on some computational strategy. Nonlinear spatial filters also operate on sliding neighborhoods, or windows, and its interaction with some operational mask. In this case, Boolean or conditional operators can be used, according to the required filter.

BACKGROUND ON THE SUPPORTED DIGITAL IMAGE PROCESSING TECHNIQUES

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 [15]:

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

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

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

The output image is then obtained as:

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

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. Figure 1 shows the linear filtering of the image 'peppers', using a 3×3 Laplacian operator as convolution mask.

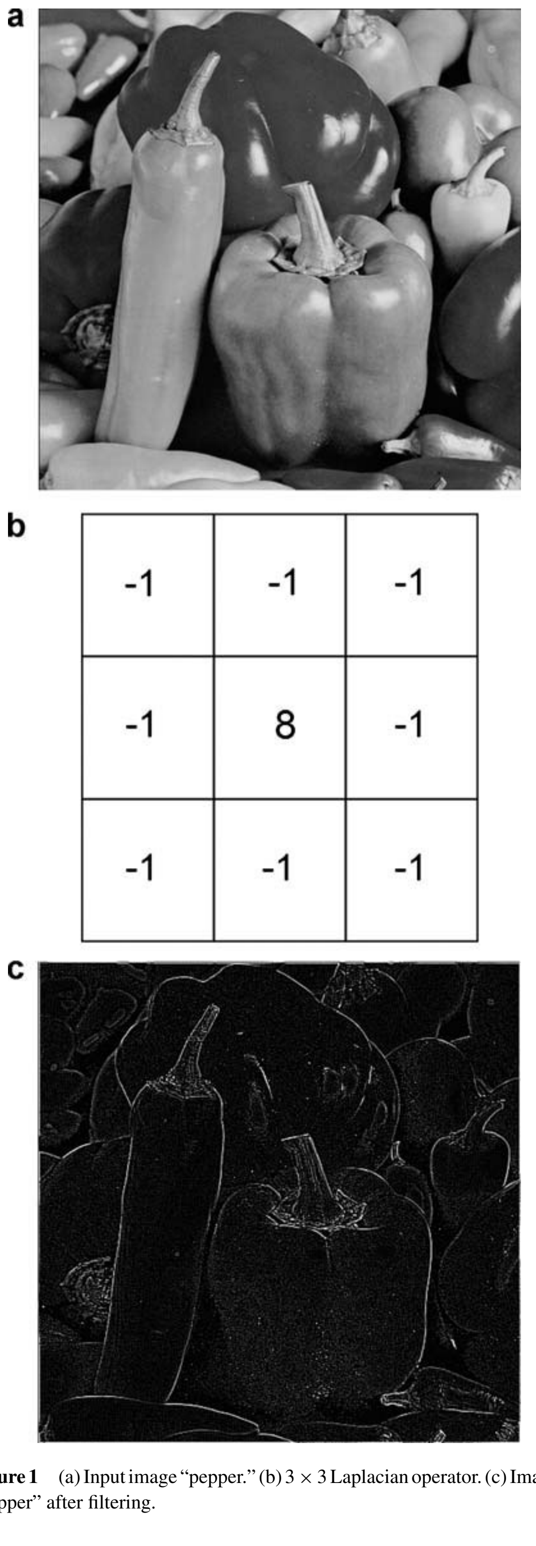


Figure 1 (a) Input image "pepper." (b) 3×3 Laplacian operator. (c) Image "pepper" after filtering.

Nonlinear Image Filtering; Image Morphology

Mathematical morphology is a geometric approach to nonlinear image processing that was developed as a powerful tool for shape analysis in binary and grayscale images. 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

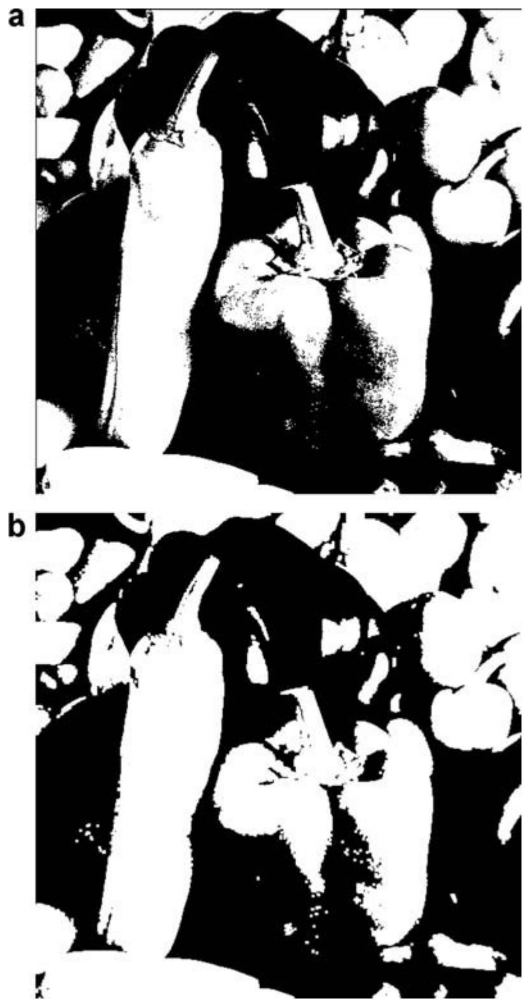


Figure 2 (a) Input image "pepperbw." (b) Dilated image using a structuring element given by a 3 × 3 square.

process [16,17]. 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.

$$A \oplus B = \cup_{b \in B} T_b(A) \tag{4}$$

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

$$A \Theta B = \cap_{b \in B} T_{-b}(A) \tag{5}$$

Figure 2 shows the result obtained when a dilation is applied to the test image "pepperbw."

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 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 defined as 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. 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. The card connected to a personal computer conform a powerful design station. The operations are performed on a 128 × 128 input binary image, organized in a linear form. 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.

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 MB 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. Figure 3 shows the 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 128 × 128 input binary image, organized in linear form as shown in Figure 4.

A schematic block diagram of the system is shown in Figure 5.

The operations inside each block are implemented in a modular way. A brief description of each module is as follows. The module *Rx_address* shown in Figure 6 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 are temporarily stored. 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 Figure 7.

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-I* signal is activated. The signal *block* is used to indicate the whole block required to process the neighborhood around the pixel and inside the convolution mask, while *enable-O* is a control signal which indicates when a memory data are ready. The signal address provides the memory address to be read, while the pair *row–column* specify the pixel position in process.

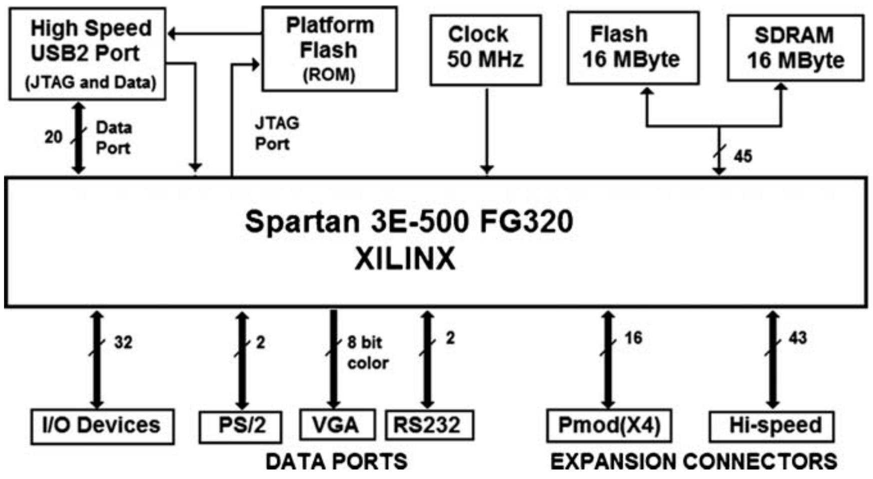
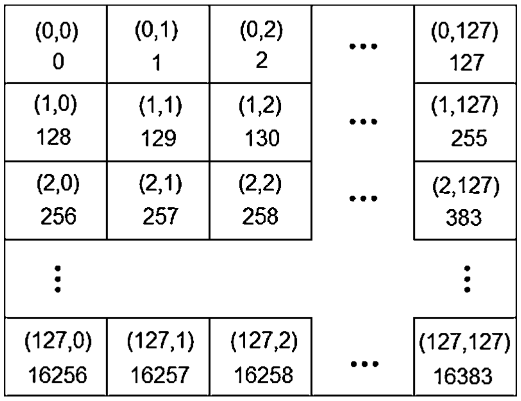


Figure 3 Xilinx® Spartan 3E block diagram.

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 *SIPO* unit, shown in Figure 8. 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 are ready. The data are entered to the next unit called *zero_padding* shown in Figure 9, which selects the operation to be executed: 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.

The module *testx2* represented in Figure 10 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 are temporarily 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 Figure 11 has a counter which is increased every time that the module *testx2* complete its routine. Once the input image has been totally processed, the signal *stored* is enabled and the system is ready to send back the processed data to the PC.



(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

Figure 4 Pixel distribution of the input image.

This operation is implemented through the *Mem_tx* unit represented in Figure 12, which receives an enable signal at the terminal *send_back*, 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 sent it back to the PC. The processed image can be visualized then through the MATLAB interface.

The master unit called *TOP-TOP1* which englobe the modules previously described, is shown in Figure 13.

MATLAB GRAPHICAL USER INTERFACE

Currently, there is 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 [18–20]. MATLAB is a widely used software environment that allows problems and solutions to be expressed in familiar mathematical notations. 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. 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. The MATLAB image processing toolbox provides a comprehensive collection of resources appropriate for a number of applications. In the work described in this paper, a MATLAB graphical user interface allows the designer to communicate with a Digilent's Nexis II system based on the Xilinx FPGA Spartan 3E. Figure 14 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. According to the type of processing specified by the user, a 3×3 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,

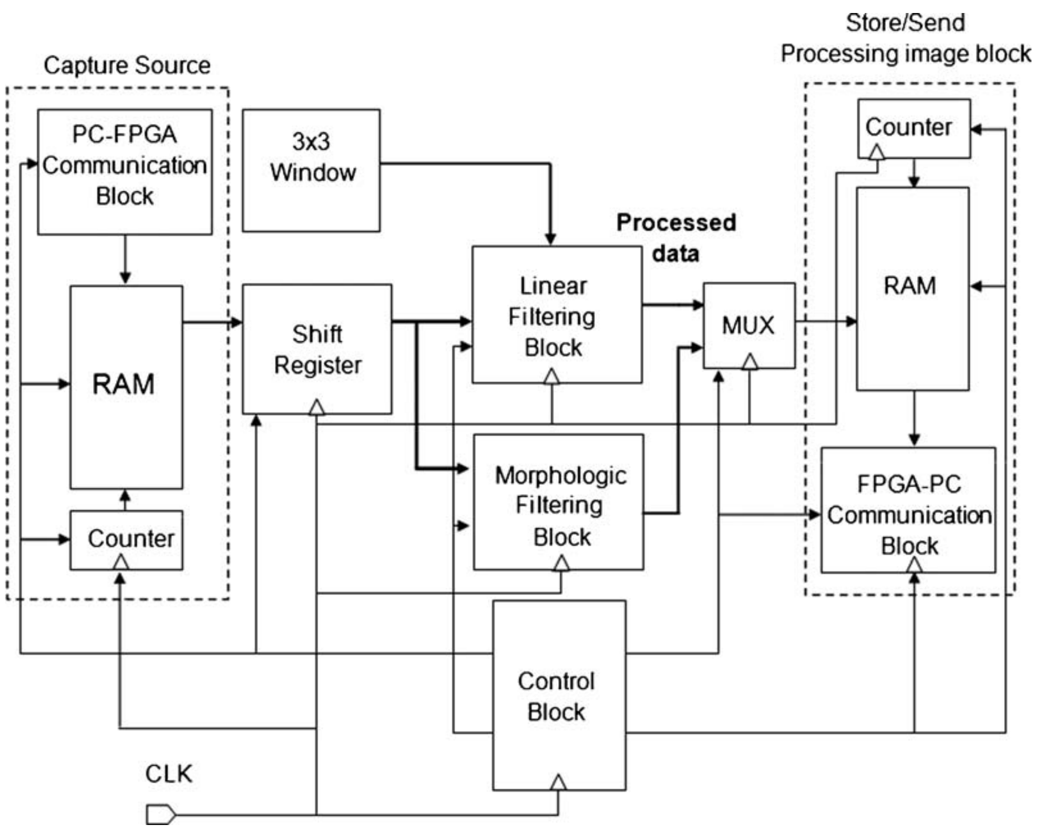

Capture Source
PC-FPGA Communication Block
RAM
Counter
3x3 Window
Shift Register
Linear Filtering Block
Morphologic Filtering Block
Processed data
MUX
Control Block
CLK
Store/Send Processing image block
Counter
RAM
FPGA-PC Communication Block

Figure 5 Block diagram for hardware Implementation of linear and morphological window filtering.

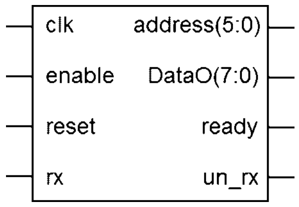


Figure 6 Module *Rx_address*.

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. Figure 14 shows the user interface when a Laplacian-based sharpening filter is applied to the image "lenna." Figure 15 shows the result obtained with a dilation morphological filter applied to the image "flower" after binarization.

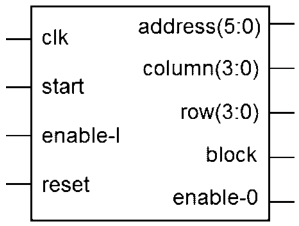


Figure 7 Module *control_submatrix*.

RESULTS

Hardware Performance

Figure 16 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 with a pulse at the signal *start*. The signal *address* contains the data addresses to be read from memory. This signal is directly connected to the RAM memory, and the data are entered sequentially in each clock cycle using the input signal *datain*. 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 10th 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

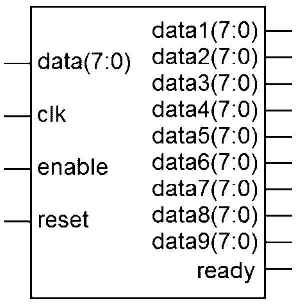


Figure 8 Module *SIPO*.

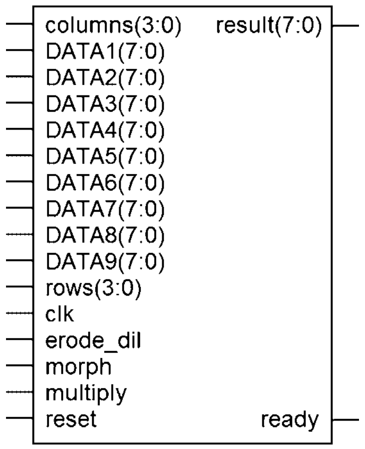


Figure 9 Module *zero_padding*.

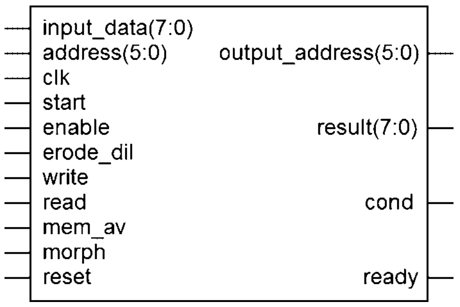


Figure 10 Module *testx2*.

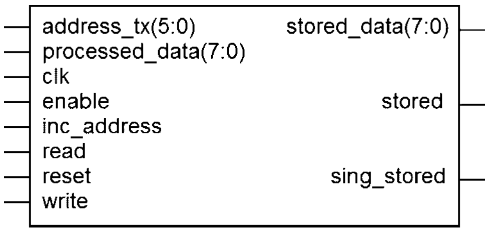


Figure 11 Memory module *count_mem_tx*.

from the multiplication. As an example of total execution time, we see that in a 128 × 128 image we have 16,384 pixels, so a number of 16,384 × 11 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 18.022 ms to process the

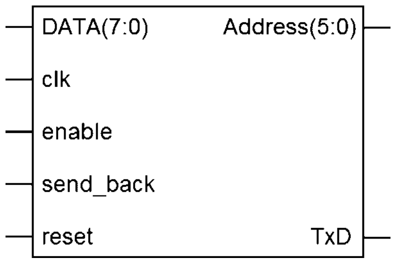


Figure 12 Module *Mem_tx*.

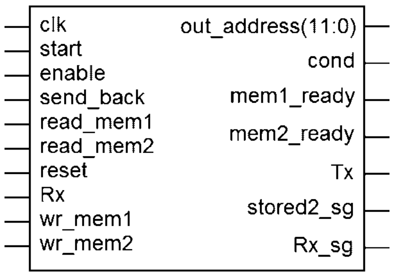


Figure 13 Master unit *TOP-TOP1*.

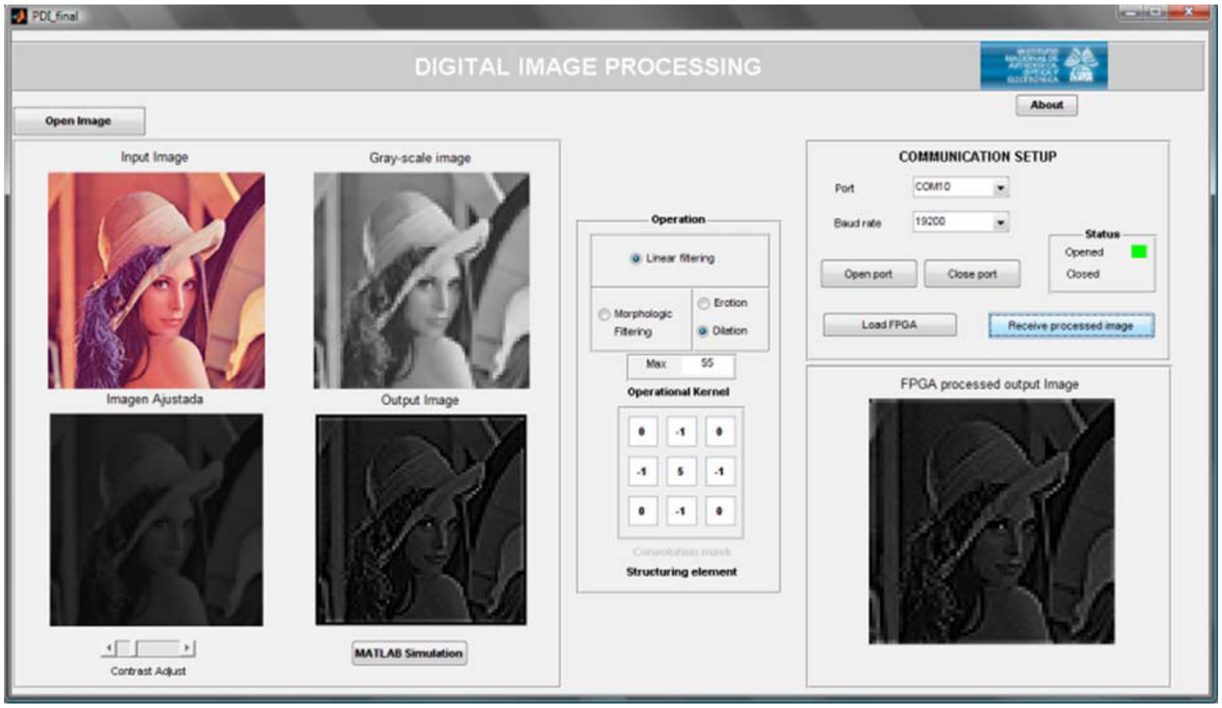
image. Table 1 shows a summary of the resources used in the FPGA. Description of signals in Figure 16 is as follows. *Start:* Input signal; it starts and controls the processing of the image. *Datain:* Input signal which receives the memory value to form the 3 × 3 matrix. *Data1-data9:* Pixel values from the 3 × 3 operational mask. *Address*: It indicates the next memory address to be read. *Multiply:* Output signal used to indicate that the total number of elements involved in the operation has been processed.

Academic Remarks

This section describes comments obtained from a standard evaluation of courses, as well as a summary of qualitative evaluations of the work in the laboratory and written observations included in the lab reports. The FPGA-based educational platform presented in this paper has been used in the graduate course G-30386 Digital Image Processing at the National Institute of Astrophysics Optics, and Electronics, Puebla, Mexico. This course is regularly attended by students of both majors: electronics and computer science. The students majoring in electronics take simultaneously a course on digital systems. According to the written comments in the lab reports, the FPGA-based educational platform helped those students to understand the data manipulation in matrix form usually associated with image processing techniques, while it motivates them to explore in the implementation of additional techniques suggested by the instructor. Moreover, comments from students majoring in computer science highlighted the fact that the system allowed them to concentrate on theoretical aspects of the algorithms and their effect on test images when using several convolution masks in the case of linear filters, or several structuring

Table 1 Device Utilization Summary

	Used	Available	Utilization (%)
Logic utilization			
Number of slice flip flops	332	9,312	3
Number of 4-input LUTs	952	9,312	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



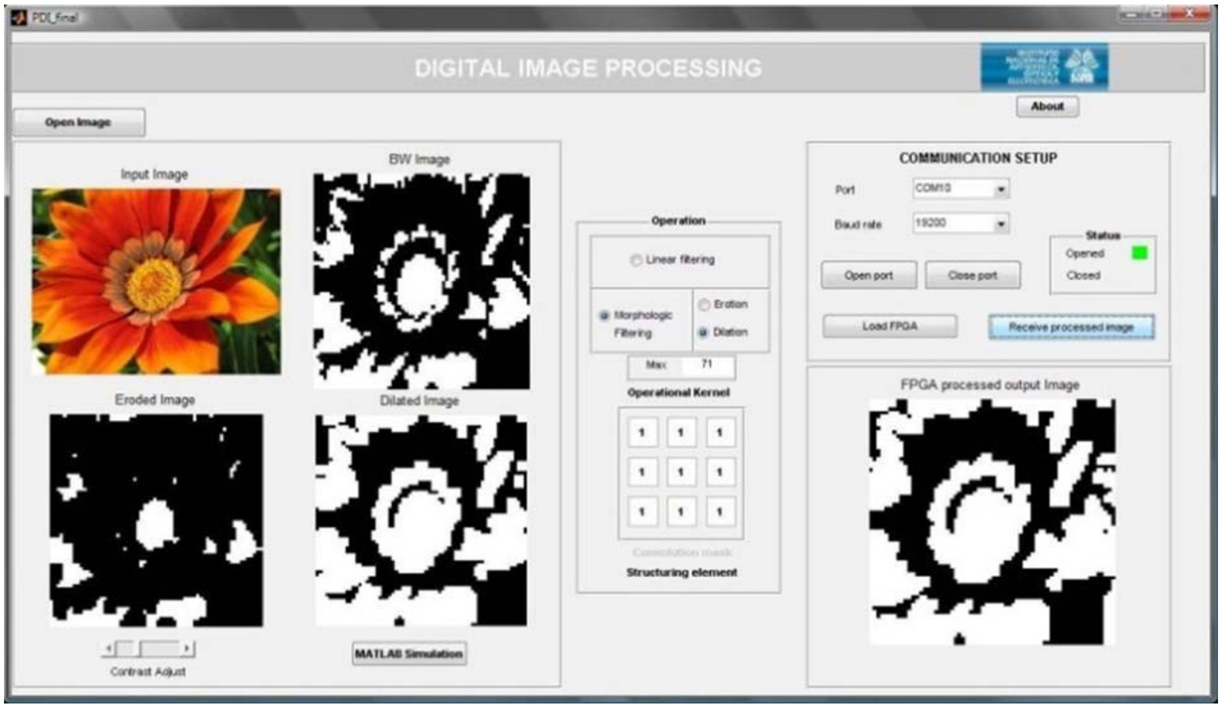
Port: COM10
Baud rate: 19200
Operation: (● Linear filtering) Morphologic Filtering; Erotion (● Dilation)
Max 55

Operational Kernel

0	-1	0
-1	5	-1
0	-1	0

Status: Opened 🟩

Figure 14 Graphical user interface main screen; sharpening filter applied to the image "lenna." [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]



Port: COM10
Baud rate: 19200
Operation: Linear filtering; (● Morphologic Filtering); Erotion (● Dilation)
Max 71

Operational Kernel

1	1	1
1	1	1
1	1	1

Status: Opened 🟩

Figure 15 Binarization and morphological dilation applied to the image "flower." [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

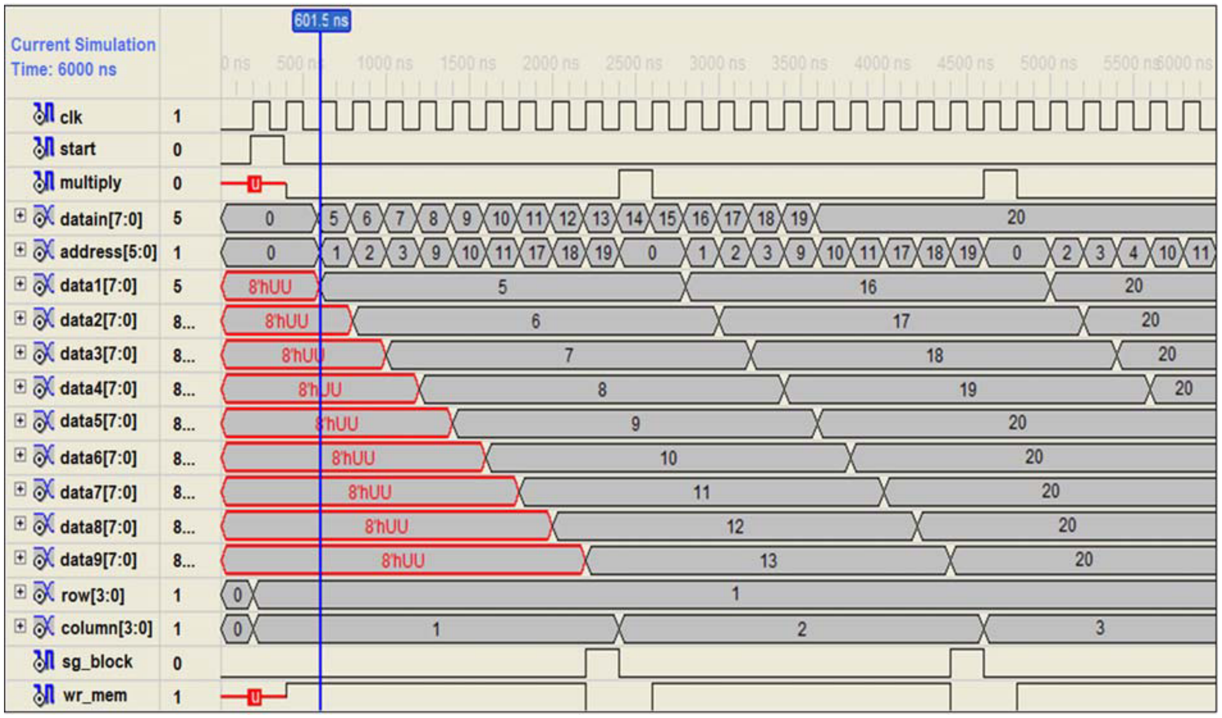


Figure 16 Time signals in the module *conv2*. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

elements in the case of nonlinear morphological filters. According to the students, the system helped them to understand the digital image concepts while doing comparisons between results provided by MATLAB, and results obtained directly in the FPGA. The students also highlighted the differences in execution time in both cases; MATLAB and FPGA, which eases the understanding on the usefulness of real-time image processing operations directed to video applications. In general the students expressed having enjoyed the material and also that they are walking away from the course with some applicable knowledge, and further applications in mind. An exit survey of students from the lab supports this contention. The overall consensus is that they enjoyed the lab, particularly the opportunity to make something with potential applications. The lab also appears to improve interest and performance in the lecture course. The Xilinx Spartan 3E Starter Kit is an affordable and complete development board system with a cost under 200 dollars, which makes it an appropriate tool to introduce the students in the use and design of digital systems with different applications such as image processing, as is the case with the system described in this work. Results indicate that this lab is successfully achieving its goals as an important part of electrical engineering, computer engineering, or computer science curricula.

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 prototype provides a very good supporting tool in modern digital system courses. At the same time, the system is a good developing platform for further real-time computer vision applications. A standard evaluation of the course and its corresponding laboratory reflect a very good acceptance by the students, regarding the incorporation of image processing experiments supported by the FPGA system described in this work.

REFERENCES

[1] S. Hauck and A. Dehon, Reconfigurable computing: The theory and practice of FPGA-based computing, Morgan Kauffman Publishers, Elsevier, Burlington, MA, 2008.
[2] M. K. Birla, FPGA Based Reconfigurable Platform for Complex Image Processing, IEEE International Conference on Electro/Information Technology, East Lansing, MA, May 7–10, 2006, pp. 204–209.
[3] K. Kumar, A. Jain, and A. K. Srivastava, FPGA implementation of image enhancement techniques, in: Proceedings of SPIE Conference on Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments, Wilga, Poland, May 2009, p. 750208.
[4] B. A. Draper, J. R. Beveridge, A. P. Willem-Böhm, C. Ross, and M. Chawathe, Accelerated image processing on FPGAs, IEEE Trans Image Process 12 (2003), 1543–1551.
[5] C. T. Johnston, K. T. Gribbon, and D. G. Bailey, FPGA based remote object tracking for real-time control, in: 1st International Conference on Sensing Technology, Palmerston North, New Zealand, November 21–23, 2005.
[6] D. K. Masrani and W. J. MacLean, A Real-Time Large Disparity Range Stereo-System using FPGAs, in: IEEE International Conference on Computer Vision Systems, New York City, NY, January 5–7, 2006, pp. 13–18.

[7] V. G. Moshnyaga, K. Hasimoto, and T. Suetsugu, FPGA design for user's presence detection, in: 15th IEEE International Conference on Electronics, Circuits and Systems, St. Julians, Malta, August 31–September 3, 2008, pp. 1316–1319.

[8] T. E. Ustun, N. V. Iftimia, R. D. Ferguson, and D. X. Hammer, Real-time processing for Fourier domain optical coherence tomography using a field programmable gate array, Rev Sci Instrum 79 (2008), 1–10.

[9] S. Saif, H. M. Abbas, S. M. Nassar, and A. A. Wahdan, An FPGA implementation of a neural optimization of block truncation coding for image/video compression, Microprocessors Microsyst 31 (2007), 477–486.

[10] M. Haldar, A. Nayak, A. Choudhary, and P. Banerjee, A system for synthesizing optimized FPGA hardware from MATLAB, International Conference on Computer Aided Design, San Jose, California, 2001, pp. 314–319.

[11] G. Martin and G. Smith, High-level synthesis: Past, present, and future, IEEE Des Test 26 (2009), 18–25.

[12] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, and E. Deprettere, System design using Kahn process networks: The Compaan/Laura approach, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France, February 16–20, 2004, pp. 340–345.

[13] J. Amaral, P. Berube, and P. Mehta, Teaching digital design to computing science students in a single academic term, IEEE Trans Educ 48 (2005), 127–132.

[14] A. Castillo, J. Vázquez, J. Ortegón, and C. Rodríguez, Prácticas de laboratorio para estudiantes de ingeniería con FPGA, IEEE Latin Am Trans 6 (2008), 130–136.

[15] R. C. Gonzalez, R. E. Woods, Steven, and L. Eddins, Digital image processing using Matlab, 2nd edition, Gatesmark Publishing, Gatesmark, LLC, 2009.

[16] F. Y. Shih, Image processing and mathematical morphology: Fundamentals and applications, CRC Press, Taylor and Francis Group, Boca Raton, FL, 2009.

[17] A. Ledda and W. Phillips, Majority Ordering and the Morphological Pattern Spectrum, in: Conf. Proc. Advanced Concepts for Intelligent Vision Systems, Antwerp, Belgium, 2005, pp. 356–363.

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

[19] B. S. Hendriks and C. W. Espelin, DataPflex: A MATLAB-based tool for the manipulation and visualization of multidimensional datasets, Bioinformatics 26 (2009), 432–433.

[20] P. Kovesi, MATLAB and Octave Functions for Computer Vision and Image Processing, people.csse.uwa.edu.au/pk/Research/MatlabFns/index.html, 2006.

BIOGRAPHIES

Juan Manuel Ramirez-Cortes was born in Puebla, Mexico. He received the BSc degree from the National Polytechnic Institute, Mexico, the MSc degree from the National Institute of Astrophysics, Optics, and Electronics (INAOE), Mexico, and the PhD degree from Texas Tech University, all in electrical engineering. He is currently a titular researcher at the Electronics Department, INAOE, in Mexico. His research interests include signal and image processing, biometry, neural networks, fuzzy logic, and digital systems. He is a senior member of IEEE.

Pilar Gomez-Gil was born in Puebla, Mexico. She received the BSc degree from the Universidad de las Americas A.C, Mexico, the MSc and PhD degrees from Texas Tech University, USA, all in computer science. She is currently an Associate Researcher in computer science at INAOE, Mexico. Her research interests include neural networks, image processing, pattern recognition, and software engineering. She is a senior member of IEEE, and a member of ACM.

Vicente Alarcon-Aquino received the BSc degree from the Instituto Tecnologico de Veracruz in 1989, the MSc degree from the Instituto Nacional de Astrofísica, Optica y Electrónica, Mexico, in 1993, and the PhD and DIC degrees from Imperial College London, University of London, London, UK, in 2003, all in electrical engineering. From 1998 to 2000, he was a Laboratory Demonstrator in the Communications and Signal Processing Laboratory, Imperial College London. Currently, he is a titular professor in the Department of Electronic Engineering at the Universidad de las Américas, Puebla, Mexico. His research interests include network protocols, time-series prediction, wavelet-based digital signal processing, hardware description languages, and multiresolution neural networks.

Jorge Martinez-Carballido received the BSc degree in electrical engineering from the Universidad de las Americas, Mexico, the MSc degree and the PhD degree in electrical engineering, both from Oregon State University. He is currently a Titular Researcher at the National Institute of Astrophysics, Optics, and Electronics (INAOE), Mexico. His research interests include digital systems, reconfigurable hardware, digital signal and image processing, and instrumentation.

Emmanuel Morales-Flores received the BSc degree in electronics and communications engineering from Benemérita Universidad Autónoma de Puebla, Mexico, in 2008. He is currently working toward the MSc degree in electrical and computing engineering at the National Institute of Astrophysics, Optics, and Electronics, Mexico. His interests include digital signal and image processing, biomedical engineering, and digital systems.