Available online at www.sciencedirect.com
SCIENCE DIRECT®
Microprocessors and Microsystems xxx (2006) xxx–xxx
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

FPGA architecture for fast parallel computation of co-occurrence matrices

D.K. Iakovidis *, D.E. Maroulis, D.G. Bariamis

Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilisia, 15784 Athens, Greece

Abstract

This paper presents a novel architecture for fast parallel computation of co-occurrence matrices in high throughput image analysis applications for which time performance is critical. The architecture was implemented on a Xilinx Virtex-XCV2000E-6 FPGA using VHDL. The symmetry and sparseness of the co-occurrence matrices are exploited to achieve improved processing times, and smaller, flexible area utilization as compared with the state of the art. The performance of the proposed architecture is evaluated using input images of various dimensions, in comparison with an optimized software implementation running on a conventional general purpose processor. Simulations of the architecture on contemporary FPGA devices show that it can deliver a speedup of two orders of magnitude over software.

Keywords: Image analysis; Co-occurrence matrix; FPGA; Texture

1. Introduction

The co-occurrence matrix is a powerful statistical tool which has proved its usefulness in a variety of image analysis applications, including biomedical [1,2], remote sensing [3], quality control [4] and industrial defect detection systems [5]. It captures second-order grey-level information, which is mostly related to human perception and the discrimination of textures.

Although the computational complexity of the co-occurrence matrix for an image of $N \times N$ dimensions is only $O(N^2)$, the processing power requirements for the computation of multiple co-occurrence matrices per time unit can be prohibiting for the analysis of large image streams, using software co-occurrence matrix implementations on general purpose processors. Such demanding applications include video analysis [1,6], content-based image retrieval [7], real-time industrial applications [5] and high-resolution multispectral image analysis [2].

Field Programmable Gate Arrays (FPGAs) are low cost, reconfigurable high density gate arrays capable of performing many complex computations in parallel while hosted by conventional computer hardware [8]. Their features enable the development of a hardware system dedicated to performing fast co-occurrence matrix computations, thus meeting the requirements of real-time image analysis applications. On the other hand, the Very Large Scale Integration (VLSI) architectures could be considered as competitive alternatives [9]. However, they are not reconfigurable and they involve high development cost and time-consuming development procedures.

Within the first FPGA architectures, dedicated to co-occurrence matrix computations, was the one presented in [5,6] for the computation of two statistical measures of the co-occurrence matrix. However, these measures were being approximated, without needing to compute the matrix itself. In a later work, Tahir et al. [2] developed an FPGA architecture for the computation of 16 co-occurrence matrices in parallel. The implementation considerations include symmetry, but do not include sparseness. As a result, a large FPGA area is utilized even for small input images.

* Corresponding author. Tel.: +30 210 7275317; fax: +30 210 7275333.
E-mail address: rtsimage@di.uoa.gr (D.K. Iakovidis).

0141-9331/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2006.02.013

In this paper, we present a novel FPGA architecture for parallel computation of 16 co-occurrence matrices that exploits both their symmetry and sparseness to achieve improved processing times and smaller, flexible area utilization.

2. The co-occurrence matrix

The co-occurrence matrix of an $N \times N$-pixel image I, comprises of the probabilities $P_{d,\theta}(i,j)$ of the transitions from a grey-level i to a grey-level j in a given direction θ at a given intersample spacing d

$$P_{d,\theta}(i,j) = \frac{C_{d,\theta}(i,j)}{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} C_{d,\theta}(i,j)}, \quad (1)$$

where $C_{d,\theta}(i,j) = \#\{(m,n),\ (u,v) \in N \times N:\ f(m,n) = j,\ f(u,v) = i,\ |(m,n) - (u,v)| = d,\ \angle((m,n),\ (u,v)) = \theta\}$, # denotes the number of elements in the set, $f(m,n)$ and $f(u,v)$ correspond to the grey-levels of the pixel located at (m,n) and (u,v), respectively, and N_g is the total number of grey-levels in the image [11]. In accordance with [2], we choose $N_g = 32$ (5-bit representation).

The co-occurrence matrix can be regarded symmetric if the distribution between opposite directions is ignored. The symmetric co-occurrence matrix is derived as $P_{d,\theta}(i,j) = (P_{d,\theta}(i,j) + P_{d,\theta}(i,j)^{\mathrm{T}})/2$, where symbol T denotes the transpose matrix. Therefore, the co-occurrence matrix can be represented as a triangular structure without any information loss, and θ is chosen within the range of 0° to 180°. Common choices of θ include 0°, 45°, 90° and 135° [1,2,6,12]. Moreover, depending on the image dimensions, the co-occurrence matrix can be very sparse, as the number of grey-level transitions for any given distance and direction, is bounded by the number of image pixels.

3. Architecture

The presented architecture was developed in Very high speed integrated circuits Hardware Description Language (VHDL). It was implemented on a Xilinx Virtex-XCV2000E-6 FPGA, which is characterized by 80×120 Configurable Logic Blocks (CLBs) providing 19,200 slices (1 CLB = 2 slices). The device includes 160 256×16-bit Block RAMs and can support up to 600 kbit of distributed RAM. The host board, Celoxica RC-1000 has four 2 MB static RAM banks. The RAM banks can be accessed by the FPGA and the host computer independently, whereas simultaneous access is prohibited by the board's arbitration and isolation circuits.

An overview of the proposed FPGA architecture is illustrated in Fig. 1. The FPGA includes a control unit, four memory controllers (one for each memory bank) and 16 Co-occurrence Matrix Computation Units (CMCUs). Up to four input images of N_g grey-levels can be loaded in parallel to the available RAM banks. In accordance with [2], a 5-bit grey-level representation was used, i.e., $N_g = 32$. However, in [2] each image is loaded into a corresponding RAM bank using a 5-bit per pixel representation whereas in the proposed architecture a 25-bit per pixel representation is used. Each pixel is represented by a vector $\bar{a} = [a_p, a_0, a_{45}, a_{90}, a_{135}]$ that comprises of five 5-bit components, namely, the grey-level a_p of the pixel and the grey-levels a_0, a_{45}, a_{90} and a_{135} of its neighboring pixels at 0°, 45°, 90° and 135° directions.

All FPGA functions are coordinated by the control unit which generates synchronization signals for the memory controllers and the CMCUs. The control unit also handles communication with the host, by exchanging control and status bytes, and requesting or releasing the ownership of the memory banks. Each CMCU is used

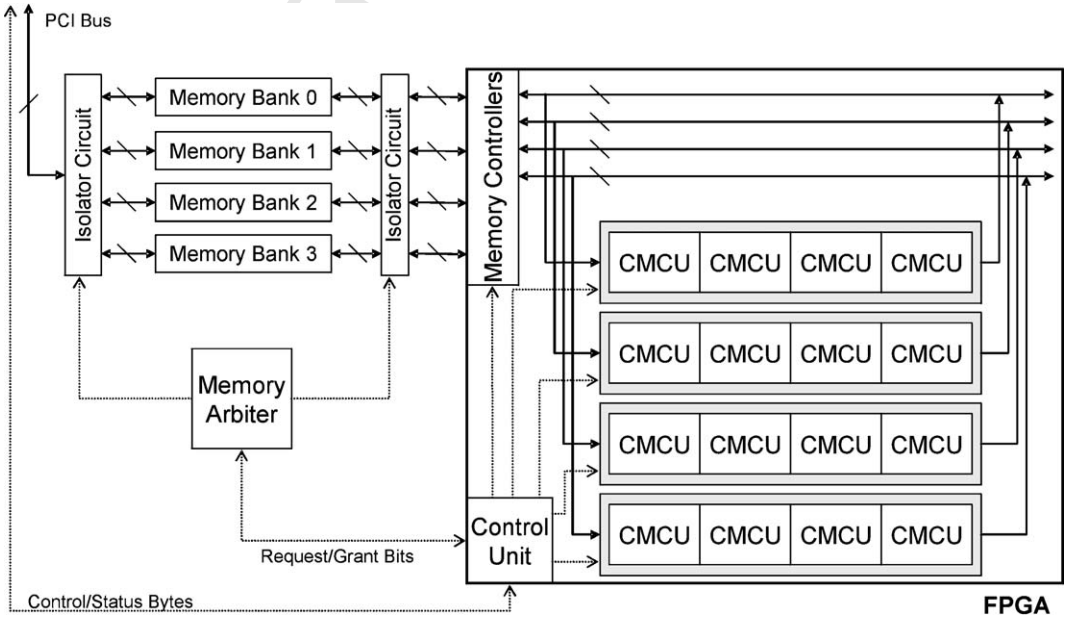


Fig. 1. Overview of the FPGA architecture.

for the computation of the co-occurrence matrix of an image for a particular direction and distance.

3.1. Co-occurrence Matrix Computation Units

Three main objectives have been determined upon the requirements of the proposed application, for the development of a CMCU: (a) small FPGA area utilization to allow for a potential expansion of the proposed architecture, (b) high throughput of one result per cycle to achieve a high per-clock performance and (c) low design complexity that will contribute to achieving high operation frequency. To meet these objectives we have considered various alternatives for the implementation of the CMCUs. These include the utilization of the existent FPGA BlockRAM arrays, the implementation of standard sparse array structures that store pairs of indices and values, and the implementation of set-associative sparse arrays. The BlockRAM arrays and the standard sparse array structures would not suffice to meet all three objectives. The BlockRAM arrays would lead to a larger area utilization compared with the sparse implementations. The standard sparse arrays would result in a lower throughput compared with the other two implementations, as the cycles needed to traverse the indices of the array are proportional to its length. In comparison, the set-associative arrays could be considered as a more flexible alternative that can be effectively used for achieving all our three objectives.

Fig. 2 illustrates a CMCU as implemented by means of an n-way set-associative array of N_c cells and auxiliary circuitry which include n comparators, a n-to-$\log_2 n$ priority encoder and an adder.

The set-associative arrays can be utilized for efficient storage and retrieval of sparse matrices, ensuring a throughput of one access per cycle with a latency of four cycles. An n-way set-associative array consists of n independent tag arrays ($\text{tags}_0 - \text{tags}_{n-1}$) as illustrated in Fig. 2. The tag-arrays are implemented in the FPGA's distributed RAM and each of them consists of N_c/n cells. The set-associative array uniquely maps an input pair of 5-bit grey-level intensities (i, j) into an address of the N_c-cell data array. The data arrays are implemented using FPGA Block RAMs, each of which can hold up to 256 co-occurrence matrix elements. The data array cells contain the number of occurrences of the respective (i, j) pairs. Each of these pairs are represented by a single 10-bit integer k, resulting from the concatenation of i and j. This integer can be considered to consist of two parts: the first is called $set(i, j)$ and comprises of the $\log_2(N_c/n)$ least significant bits of k, whereas the second part is called $tag(i, j)$ and comprises of the 10-$\log_2(N_c/n)$ most significant bits of k. The increment of a data array cell that corresponds to an input pair (i, j) is implemented in four pipeline stages:

Stage 1. The tag array cells located in the $\text{set}(i, j)$ row are retrieved and stored in temporary registers.

Stage 2. The values of the temporary registers values are compared with $\text{tag}(i, j)$.

a. If a match is found the column number of the matching tag is written in the offset register.

b. If there are not any matches the $\text{tag}(i, j)$ is stored in the tags array, at the first available cell of the $\text{set}(i, j)$ row.

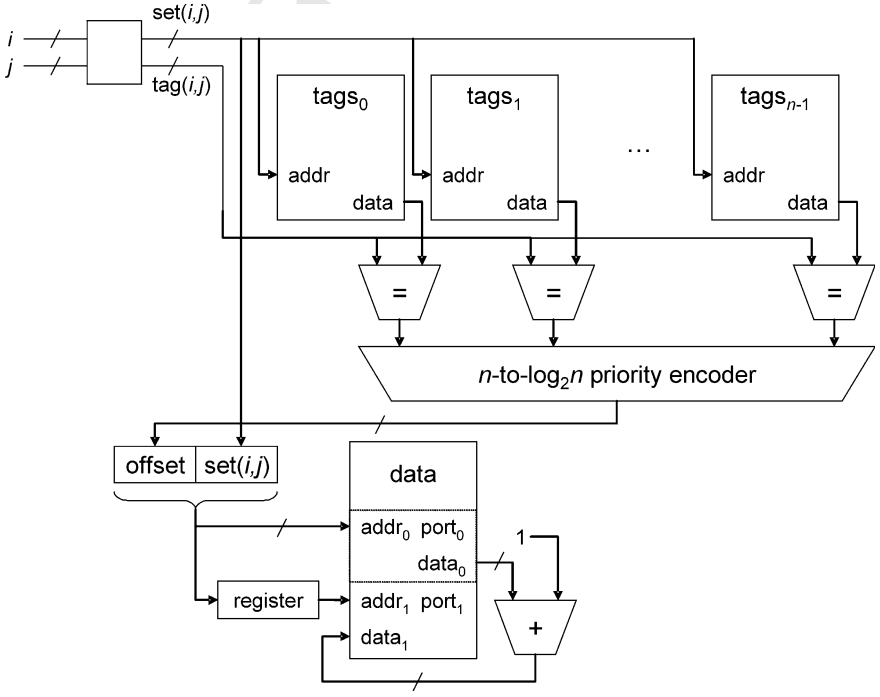


Fig. 2. The co-occurrence matrix computation unit (CMCU).

Stage 3. The contents of both the offset register and set(i, j) form an address a. The data array element stored in a is read.

Stage 4. The value read in the previous cycle increases by one and it is written back to a.

After all input pairs are read and processed the data array will contain the co-occurrence matrix of the input image.

4. Results

Experiments focusing to the evaluation of the time performance and the area utilization of the proposed architecture were performed using standard texture images from the Brodatz album of 16×16, 32×32, 64×64, 128×128, 256×256 and 512×512-pixel dimensions (Fig. 3) [13].

Given a triangular co-occurrence matrix of $N_g = 32$, the number of pixel pairs that can be considered for its computation in the case of a 16×16-pixel input image, is smaller than the total number of co-occurrence matrix elements, and reaches the number of all image pixels. Therefore, the co-occurrence matrix will be sparse and N_c is set to a maximum possible value of $16\times16 = 256$. In the case of a 32×32-pixel or a larger input image, the co-occurrence matrix is not considered sparse as the number of all possible pixel pairs that can be considered for its computation is larger than the total number of its elements (i.e., 528). Therefore, N_c is set to 528. It is worth noting that the effect of sparseness in area utilization is amplified and becomes more useful as N_g increases. For example, if N_g was set at 64 or at 128 grey-levels, the co-occurrence matrix could be considered sparse for images up to 32×32 or 64×64-pixel dimensions, respectively. By following a grid search approach for the determination of n, it was found that the 16-way set-associative arrays ($n = 16$) result in the optimal tradeoff between time performance and area utilization.

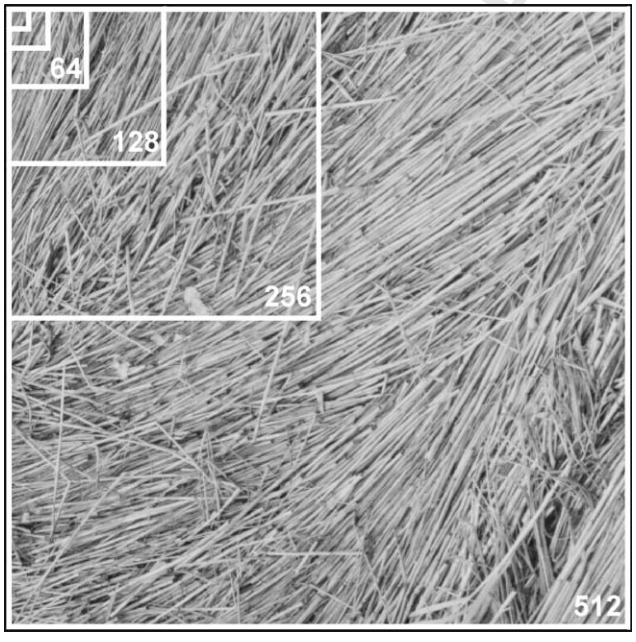


Fig. 3. Texture image D9 from the Brodatz album cropped at various sizes.

The proposed architecture, as implemented on the Xilinx Virtex-XCV2000E-6 FPGA, operates at 38.4 MHz and utilizes only 39% of the FPGA area for 16×16 input images, where the sparseness of the co-occurrence matrices is exploited. The use of larger input images results in approximately the same operating frequency reaching 38.2 MHz and a larger area utilization of 45%. In comparison, the architecture proposed in [2] operates at 50 MHz and utilizes a larger area percentage (59%) on an FPGA of the same type, for the same N_g, regardless of the image dimensions. The time performance reported in [2] for the computation of a total of 64 co-occurrence matrices in 512×512-pixel 16-band multispectral images was 6.3×10^5 μs. For the same computations, the proposed architecture requires $28.041\times4 = 1.1\times10^5$ μs (Table 1), which can be interpreted in approximately 500% reduction of the processing time. This improvement in time performance is mainly attributed to the use of vectors $\bar{a}$ for the retrieval of five pixels in one cycle instead of the five cycles required in the per pixel retrieval used in [2].

Even though the implementation of the proposed architecture was based on the Xilinx Virtex-XCV2000E-6 FPGA, we run several simulations on state of the art FPGA devices, such as Virtex-XCV2000E-8 (19200 slices), Virtex2-XC2V6000-6 (33792 slices) and Spartan3-XC3S4000-5 (27648 slices). The processing times achieved for the computation of 16 co-occurrence matrices in hardware and software, respectively, are presented in Table 1.

Software processing times were measured using an MMX optimized software implementation developed in C programming language and executed on an Athlon XP2700+ processor. The optimizations were based on the guidelines suggested by Intel and AMD [14,15]. These include contiguous arrays allocation for improving CPU caching performance, system call overhead reduction by allocation of static arrays for data used iteratively within the program, usage of efficient C library functions such as memset() and memcpy(), and vectorization of several functions using the MMX instruction set [16]. Additional code fine-tuning includes code rearrangement for breaking dependencies in tight loops, dereferencing of commonly used pointers and reduction of the function call overhead using inline functions.

The results reveal the superior performance of the hardware implementations of the proposed architecture over the software implementation. The speedup factors achieved in hardware vary depending on the FPGA model used. The minimum speedup is approximately 20 in the case of XCV2000E-6 for 512×512 images, whereas it exceeds 100 in the case of XC2V6000-6 for 16×16 images. The variance in speedup is mainly attributed to the different frequencies of the various FPGA models and does not

Table 1
Processing times (μs) achieved for various input image dimensions using various FPGA devices, and software

Implementation	Frequency (MHz)	Image dimensions (pixels)					
		16 × 16	32 × 32	64 × 64	128 × 128	256 × 256	512 × 512
Processing times (μs)							
Hardware							
XCV2000E-6	38	30	113	442	1756	7013	28,041
XCV2000E-8	51	22	83	323	1283	5123	20,483
XC3S4000-5	72	15	59	230	915	3653	14,606
XC2V6000-6	83	13	51	198	788	3149	12,590
Software							
Athlon XP 2700+	2167	1371	3247	10,018	36,320	143,600	562,080

correlate with the sparseness of the co-occurrence matrix, which mainly affects the area utilization. In Table 1 it can be observed that the increase in processing times as the image dimensions increase by two is not exactly divided by four, as it would have been expected by the quadruplication of the image pixels. This is explained by the constant time period spent for resetting the FPGA circuit.

5. Conclusions

We presented a novel FPGA architecture which is capable of performing fast parallel co-occurrence matrix computations in grey-level images. It performs better than the state of the art FPGA architecture presented in [2]. The proposed architecture and the architecture in [2] have two main differences pinpointed to the input data format and the co-occurrence matrix representation. The vector representation of the input image pixels and the use of set-associative arrays for the sparse representation of the co-occurrence matrix result in a higher time performance and smaller area utilization respectively. Its advantageous time performance compared with the architecture in [2] and with an optimized software implementation for general purpose processors, makes it appealing for use in high throughput applications. Moreover, the smaller FPGA area it utilizes, allows for the exploitation of the remaining area for other tasks, such as the computation of co-occurrence matrix features [11], or the computation of more co-occurrence matrices in parallel, if the host board is equipped with more RAM banks.

It is worth noting that the computation of co-occurrence matrices in conjunction with feature extraction in the same FPGA design still remains a challenge. In [2], two different FPGA designs, one for the computation of co-occurrence matrices and one for the feature extraction, are interchangeably configured on a single FPGA.

Within our future perspectives are the extension of the current architecture for efficient on-chip extraction of multiple textural features from grey-level and colour images, in the same FPGA design, and its integration in a complete, hardware/software system with real-time video analysis capabilities.

6. Uncited reference

Ref. [10].

Acknowledgments

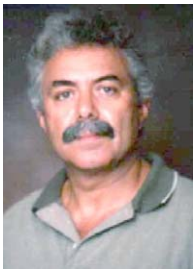
This research was funded by the Operational Program for Education and Vocational Training (EPEAEK II) under the framework of the project "Pythagoras – Support of University Research Groups" co-funded by 75% from the European Social Fund and by 25% from national funds.

References

[1] S.A. Karkanis, D.K. Iakovidis, D.E. Maroulis, D.A. Karras, M. Tzivras, Computer aided tumor detection in endoscopic video using color wavelet features, IEEE Trans. Inf. Technol. Biomed. 7 (2003) 141–152.
[2] M.A. Tahir, A. Bouridane, F. Kurugollu, An FPGA based coprocessor for GLCM and haralick texture features and their application in prostate cancer classification, Analog Integr. Circ. Signal Process. 43 (2005) 205–215.
[3] A. Baraldi, F. Parmiggiani, An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters, IEEE Trans. Geosci. Remote Sens. 33 (2) (1995) 293–304.
[4] K. Shiranita, T. Miyajima, R. Takiyama, Determination of meat quality by texture analysis, Pattern Recogn. Lett. 19 (1998) 1319–1324.
[5] J. Iivarinen, K. Heikkinen, J. Rauhamaa, P. Vuorimaa, A. Visa, A defect detection scheme for web surface inspection, Int. J. Pattern Recogn. Artif. Intell. (2000) 735–755.
[6] D.K. Iakovidis, D.E. Maroulis, S.A. Karkanis, I.N. Flaounas, Color texture recognition in video sequences using wavelet covariance features and support vector machines, in: Proceedings of 29th EUROMICRO, September 2003, Antalya, Turkey, 2003, pp. 199–204.
[7] C.-H. Wei, C.-T. Li, R. Wilson, A content-based approach to medical image database retrieval, in: Z. Ma (Ed.), Database Modeling for Industrial Data Management: Emerging Technologies and Applications, Idea Group Publishing, 2005.
[8] T.A. York, Survey of field programmable logic devices, Microprocess. Microsyst. 17 (7) (1993) 371–381.
[9] M. Ba, D. Degrugillier, C. Berrou, Digital VLSI using parallel architecture for co-occurrence matrix determination, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 4, 1989, pp. 2556–2559.
[10] K. Heikkinen, P. Vuorimaa, Computation of two texture features in hardware, in: Proceedings of the Tenth International Conference on Image Analysis and Processing, September 1999, Venice, Italy, 1999, pp. 125–129.

[11] R.M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cybern. 3 (1973) 610–621.
[12] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, San Diego, 1999.
[13] P. Brodatz, Textures: A Photographic Album for Artists and Designers, Dover Publications, New York, 1966.
[14] IA-32 Intel Architecture Optimization Reference Manual, Intel Corp., 2004.
[15] Athlon Processor x86 Code Optimization Guide, AMD Inc., 2002.
[16] Intel Pentium 4 Processor Optimization Reference Manual, Intel Corporation, 1999–2000.

Dimitris K. Iakovidis received his B.Sc. degree in Physics from the University of Athens, Greece. In April 2001, he received his M.Sc. degree in Cybernetics and in February 2004 his Ph.D. degree in Computer Science from the Department of Informatics and Telecommunications, University of Athens, Greece. Currently he is working as a Research Fellow in the same Department and he has co-authored more than 30 papers on image analysis, systems, and biomedical applications. Also he is a regular reviewer for many international journals. His research interests include image analysis, system development, pattern recognition and bioinformatics.



Dimitris E. Maroulis received the B.Sc. degree in Physics, the M.Sc. degree in radioelectricity, the M.Sc. in electronic automation and the Ph.D. degree in Computer Science, all from the University of Athens, Greece, in 1973, 1977, 1980 and 1990, respectively. In 1979, he was appointed Assistant in the Department of Physics, in 1991 he was elected Lecturer and in 1994 he was elected Assistant Professor, in the Department of Informatics of the same university. He is currently working in the above Department in teaching and research activities, including Projects with European Community. His main areas of activity include data acquisition systems, real-time systems, signal processing and biomedical systems.

Dimitris G. Bariamis is a student of the Department of Informatics and Telecommunications, pursuing a Ph.D. degree in hardware architecture design. His research interests include FPGA design, and software programming and optimization techniques.