EURASIP Journal on Applied Signal Processing 2005:7, 989–992

Editorial

Magdy A. Bayoumi
The Center for Advanced Computer Studies, University of Louisiana at Lafayette, LA 70504-4330, USA
Email: mbayoumi@cacs.louisiana.edu

Bertrand Zavidovique
Institut d'Électronique Fondamentale, Université Paris-Sud, 91405 Orsay Cedex, France
Email: betrand.zavidovique@ief.u-psud.fr

Perception can have different meanings to different groups. It has a wide range of connotations; from sensing to understanding to decision making. Perception is a basic fundamental core in many systems, such as autonomous systems and vision-based applications. A typical scenario of a perception system is as follows: data are asserted by sensors (visual, auditory, or others) to be converted into corresponding signals. Common operations at this stage are noise removal, transcoding, compression, and so forth. Specific features can be extracted in view of more semantic compression and then object recognition can take place. The main requirements of such perception systems include

(i) intelligent and computational sensors;
(ii) perception and sensing algorithms;
(iii) vision-based algorithms; for example, collision avoidance, trading, autonomous motion, and so forth;
(iv) highly parallel architecture;
(v) adaptive and configurable hardware;
(vi) software environment;
(vii) vision computational kernels; for example, decision-based tasks, mixed-based functions, region-based modules, and so forth.

Perception systems are highly dependent on technology. From sensors technologies (such as infrared, carbon tubes, MEMS, etc.) to integrated circuit technologies (CMOS, submicron, FPGA), realization of the perception system is affected in capabilities, cost, and performance. The main challenge is that these systems require integration of several different technologies. Moreover, current technologies do not allow supporting all semantic levels of data and information processing. Available systems at present times are a mixture of hardware implementation at various scales with software implementation of some specific modules leading to hybrid systems. The evolving system-on-a-chip technology offers the most promising technology for perception systems because of the different technology needs of the system's building blocks.

The ultimate definition of system-on-a-chip (SoC) is the integration of disparate technologies, built around a monolithic processor, that will act as a complete system. These disparate technologies include microelectronic circuits, MEMS (microelectromechanical systems), microfluidic devices, sensors (magnetic, electric field, chemical, etc.), wireless circuits and devices (including adaptive antennas), and optoelectronic and photonic devices. The integration issue of designing, simulating, verifying, and interconnecting such disparate technologies is the certain evolution of SoC.

Currently, SoC is identified with designing mainly digital systems with tens of millions of transistors and the associated issues with design tools, design verification, testing strategies, and reliability. In general, SoC phrase is applied to a wide variety of technological issues at different abstraction levels.

SoC paradigm leads to reduction in power consumption, design cost, and time to market. New design methodologies are needed with drastically different capabilities from the existing ones, such as how to deal with more than 10 million transistors in a single chip, IP cores and circuit macro-reuse, and prototyping.

Perception systems fit, naturally, SoC technology. Having sensing, processing, and displaying on a single chip will lead to optimal performance. Due to the experimental nature of machine perception algorithms and design process, prototyping is considered a significant step towards having perception chips. This special issue presents several case studies of prototyping of modules and systems as an essential design step towards perception on a chip.

The first paper by Reyna-Rojas et al. addresses two of the main design issues in SoC design and implementation; SoC platform architectures and SoC rapid prototyping.

Platform-based design has been recognized as an efficient design approach for SoCs where high degree of reconfigurability and IP reuse can be achieved. This is essential for realistic design time (i.e., time to market) and cost. Platform architectures experience the same classic dilemma of applicability (i.e., generality) versus efficiency. The authors have followed a widely accepted and proven approach by developing an algorithmic-specific platform. It is specific for DSP and image processing applications that are based on matrix/vector and MAC operations.

A case study of real-time object recognition based on a neural-network support vector machine (SVM) approach is analyzed. The SVM is a classifier that is based on statistical learning theory. The number of required operations and the computing time increases proportionally to the number of supporting vectors. It has been proven as an efficient technique in several applications, for example, matrix bar codes detection (which is discussed in the paper), face detection in an automobile cockpit, and white lines detection. The SVM recognition system is mapped into the proposed platforms where the parallel feature of the SVM has been exploited. The reconfigurability of the platform provides high degree of flexibility in implementing several SVM functions dedicated to any object recognition problem.

The authors have also proposed an experimental rapid-prototyping platform for SoC emulation. The main computational core of the system is a Xilinx XC2V3000 FPGA package and PCI-X controller. An interesting discussion of prototyping the SVM application is given in the paper.

The second paper by Coudarcher et al. addresses software prototyping (SP). It is a specific case study of a parallel programming environment, called SKiPPER, developed at the Blaise-Pascal University. It is based on skeleton nesting technique. Its main goal is designing and implementing parallel image processing and vision applications. The parallelization capabilities of this environment will lead to faster and less complex implementations of several computational-intensive vision tasks. The significance of their prototyping environment is that it can be used for software implementation systems or as a first stage for hardware prototyping. Moreover, this environment can also be used for prototyping of embedded vision applications. This feature is not available in many other prototyping systems, although it represents a challenge as systems get more heterogeneous, in functions and technology.

The SKiPPER system provides the users with guidance and mostly automatic procedure for designing and implementing parallel applications. A case study of parallel prototyping of a 3D face-tracking algorithm is discussed and analyzed. It is a computation-intensive application. It is based on representing the face by a collection of 2D images (reference views). The tracking technique is a matrices computation-intensive task. The paper presents a systematic and logical procedure in parallelization and implementation of this case study.

The third and fourth papers present case studies of FPGA prototyping of two basic computational kernels in vision perception; window-based image processing and decision rule function, respectively. Both cases use the FPGA's different capabilities to realize efficient implementation rather than a mere direct mapping. FPGAs provide massive parallelism at the logic and bit levels. Moreover, datapaths can be controlled at the bit level.

In the third paper, Torres-Huitzil and Arias-Estrada present a detailed example of mapping a window-based image processing function into efficient FPGA kernel. This function is a fundamental operation in many computer vision applications. The main features of this function are intensive computation, intensive memory requirements, and high potentiality for parallelism. The authors developed a detailed and systematic approach in exploiting the inherent data parallelism in window-based operations. It is combined with loop unrolling to compute more than one window in parallel, either in the vertical or in the horizontal direction. This scheme will lead to reducing the memory access overhead. The developed architecture is reconfigurable. It is based on module reuse through a column-based memory addressing mode and a set of processing elements operating in parallel.

The building block unit of this system is a configurable processor. It has a simple architecture that can be configured by a control word, selected by the user, in runtime. It performs three disjoint operations, in parallel, in each clock cycle. The architecture is synthesized with XST tool and prototyped using a Virtex XCV2000E-6 device and RC1000PP board. The board is connected to the PCI bus of a host computer. The prototyped board has been evaluated for several window operations; image convolution, template matching, and gray-level image morphology. Both architecture and the prototype provide high level of configurability. It can be used as a platform to explore more complex algorithms such as block matching for motion estimation, stereo disparity computation, and discrete cosine transform.

While this paper presents detailed treatment of the window-based image processing fundamentals, architectures, and their FPGA mapping, the fourth paper by Mitéran et al. focuses on developing a tool of automatic FPGA prototyping. A case study of hardware prototyping of a decision rule function (DRF) is demonstrated. DRF is a basic functional kernel in several computer vision tasks, such as hi-resolution image segmentation and pattern recognition in very large databases. The main computational kernel of DRF is the classifier, which can be implemented in several approaches, such as neural networks, support vector machine (SVM) (as in the first paper), or the AdaBoost algorithm. This paper employs the latter. The AdaBoost algorithm offers an efficient compromise between the system resolution and hardware implementation.

The paper offers both integrated design methodology and a design tool called Boost2VHDL. This tool is developed in C++. It generates, automatically, the hardware description of a given decision function, while finding an efficient trade-off between the classification resolution, computation speed, and silicon area. The generated hardware is programmable. Experimental validation has taken place using real databases. The developed design's method and tool are applied in a

quality control process. The goal is to detect some anomalies on manufactured parts at a rate of 10 pieces per second.

The fifth paper, by Nair et al., and the first in a series of four papers on system prototyping, is an astute example of adaptation of an application under technological constraints. The application is "people detection" in a structured environment. Detection is not an end in itself; rather it is an intermediate step to localization, identification, and tracking. A network of cameras is assumed for capturing the scene which, in turn, necessitates compression for communication as well as supervised classification for region detection. Supervised machine-learning-based classification allows the system to work without failure in the case of sudden illumination as well as shifts from one camera to another. The beginning of the study shows how procedure choices make a logical and progressive tradeoff between application requirements, such as detection using multiple sensors, compression, classification, and system capabilities (network processing, control, etc.) of FPGAs.

FPGAs can realize both an embedded microprocessor for controlling general operations and network protocols, and application-specific firmware, which can be as parallel as needed and can be tightly coupled to cameras and memories. Yet, implementing classification in FPGAs can be difficult because of large on-chip memory requirements for data streaming. Again, looking for tradeoffs leads to a choice of AdaBoost algorithm for generating a chain of classifiers that candidates go through for rejection. Candidates, here, are sets of sum differences over well-chosen couples of neighborhoods at various scales and locations. Such computation is efficiently performed on a scan-plus version of the input image that commutes with the DCT in a JPEG compression since both are linear. The remaining question is purely hardware at that stage which is minimizing the memory access. This can be achieved by computing features at selected points and interpolating to approximate the internal image. The interpolation requires a single multiplier for all points due to coefficient repartitions. The whole work is most representative of practical problem solving by a mix of algorithm selection and adaptation to suit the architectural and hardware capabilities. It puts forward two other illustrated points that might interest the reader: (1) how image processing and pattern recognition are to be balanced on a chip with the help of implementation constraints, and (2) how the same type of procedure raises different concerns depending on prototyping goals. For example, AdaBoost is an alternative to support vector machines and choices are different under system considerations.

The sixth paper by Barbaro and Raffo, also on vision, adds the sensory dimension in place of the traditional image recognition. The presented design utilizes analog-computing cells for real-time filtering at the pixel level. From a system's point of view, the rationale for a smart sensor approach comes from the idea that only significant results are output in some specific applications. Many examples of such applications stem from mobility—they range from car safety (e.g., time to collision) to character and optical recognition in cell phones (e.g., address storing, privacy protection), assistance for disabled people, and surveillance. Physical constraints deal with environment concerns (e.g., wave length, secrecy, or sensitivity/exposure), embarkability (e.g., hardening, cumbersomeness) or power supply (e.g., low power), not to mention the unavoidable economic dimension. In the present example, a low-power CMOS sensor is enriched with filtering capabilities for the sensor to deliver salient, limited information. The work exemplifies how procedures and device implementations have to be conceived jointly with tradeoffs between algorithmic versatility (or modularity) and partitioning.

Gabor filters are selected for versatility as they are proven efficient in depth or motion finding, as well as in texture analysis. For simplicity, photodiodes are taken for the light to current transducers and image scans use ring counters that exist in standard cell to save some design effort. As for the design, linear filtering is an interaction among pixels, except for the possible sensitivity to external perturbations (often lightening conditions) that induces less local comparison with an average (e.g., mean, median, or Laplacian). A fast factor in analog implementation evaluation is the degree of parameterization. Performances, in the given example, range in the order of $1\,\mu$W per pixel and 24 microseconds per filtering (independent of the size of the retina). This example helps to understand how many such elementary processing can be stacked depending on the technology capabilities.

The last two papers address auditory systems in two very different manners. One is more system oriented with digital design, it is analyzed at a functional design level. The other paper is more device oriented with analog techniques at the cell level. The lower dimensionality in this case, however, favors more generic prototyping as well as more thorough simulation.

The seventh paper, by Sawan et al., tackles the design of a DSP core dedicated to image formation in ultrasound imagery. Recognition by system is not required in this case as intelligent recognition is expected to be performed by a human expert. The difficulty is mostly in miniaturization. The main processing comprises reading data from an external ultrasonic sensor array (in B-mode), computing the beamforming, filtering high-frequency noise, outputting the phase from lowpass-filtered IQ demodulation, and computing the magnitude. FPGA technology is integrated with ARM hardware for rapid prototyping. Difficulty may arise when using a particular technology-bound design tool, for example, FPGA, with simulation tools such as Simulink in Matlab. Constraints such as module intercommunication ability, memory type, memory size, clock generation, pipeline depth, and tabulation/computation are taken into consideration by algorithmic simulation and procedural choices. The paper illustrates the need for intuitive design techniques beyond the interest of the application.

The last paper, by Ravindran et al., illustrates an inspired human-like biological model of perception. The paper embeds the sample design within the framework of a more general approach named CADSP. The approach aims at adding digital functionality to analog devices to balance the activity between the analog massive processing and the digital ex-

ternal control, including acquisition management and signal recoding for conversion. The CMOS floating gate is introduced as the vector of increased programmability or flexibility. It saves power consumption too. It supports translinear function design by merely adapting capacitive coupling. The adopted FPAA (A for analog) design technology is highlighted with reference to the FPGA (G for gate) regarding the lack of availability of tools as well as the design-specific challenges. Several generic functions in audio processing are then described covering all stages of the processing chain such as noise suppression, feature extraction, Mel-cepstrum coefficient (regular MFCC), Yang's model coefficient (MFCC-like), classification, and vector quantization. Due to the unique variations of the floating-gate technology, tradeoffs deal with how to approximate filters to fit them in the abilities of buildable devices assuming noise stationarity or frequency repartitioning for transforming SNR into gain factor.

Finally, we would like to thank all colleagues who have contributed to this special issue, including all the authors who submitted papers. We appreciate the reviewers for all their efforts, Dr. Ashok Kumar (CACS), Dr. Marc Moonen, and the Editorial Board, without their support this special issue would not be possible. We acknowledge the support from CACS, University of Louisiana at Lafayette, and University de Paris-SUD XI.

Magdy A. Bayoumi
Bertrand Zavidovique

Magdy A. Bayoumi is the Director of the Center for Advanced Computer Studies (CACS) and Department Head of Computer Science Department, University of Louisiana at Lafayette. Dr. Bayoumi received the B.S. and M.S. degrees in electrical engineering from Cairo University, Egypt; an M.S. degree in computer engineering from Washington University, St. Louis; and a Ph.D. degree in electrical engineering from the University of Windsor, Canada. Dr. Bayoumi was the Vice-President for the technical activities of the IEEE Circuits and Systems Society. He is the Chairman of the Technical Committee (TC) on Circuits and Systems for Communications and he was the Chairman of the TC on Signal Processing Design and Implementation. He was a Founding Member of the VLSI Systems and Applications Technical Committee and was its Chairman. He is the General Chair for ISCAS 2007. He was the General Chairman of the 1994 MWSCAS. He was an Associate Editor of the Circuits and Devices Magazine, the IEEE Transaction on VLSI Systems, the IEEE Transactions on Neural Networks, and the IEEE Transactions on Circuits and Systems II. He was the Cochairman of the Workshop on Computer Architecture for Machine Perception, 1993, and is a Member of the Steering Committee of this workshop. He was the General Chairman for the 8th Great Lake Symposium on VLSI, 1998. He was also the General Chairman of the 2000 Workshop on Signal Processing Design and Implementation. Dr. Bayoumi is an Associate Editor of Integration, the VLSI Journal and the Journal of VLSI Signal Processing Systems.

Bertrand Zavidovique received the M.S. degree in mathematics from Paris VII University in 1971, the Ph.D. degree in computer science from the University of Tours, France, in 1975, and the Doctorate of Science degree in robot vision from the University of Franche Comté, France, in 1981. He is currently a Professor in the Electrical Engineering Department, Paris XI University, and was a Director of Doctoral Studies in electrical engineering from 1992 to 2000. From 1981 to 1996, he was a Scientific Advisor at the French Ministry of Defence for the problems of real-time processing and robotics, heading the Perception System Laboratory. He was an Invited Professor in the Department of Electrical Engineering and Computer Science, the University of California, Berkeley, in 1985–1986, and in the Center for Advanced Computer Studies, the University of Louisiana, Lafayette, in 1996–1997. His research interests include perception systems, the impact of their internal organization on external efficiency, real-time implementation, and architecture and programming methods within the frame of circuit integration. He has published more than 250 scientific papers in image processing, sensor fusion, computer architecture, and intelligent control and learning. Professor Zavidovique was awarded several national and international scientific prizes.