Algorithms and VLSI Architectures for Pattern Recognition Based on the Gabor Wavelets

Mourad Aberbour, Habib Mehrez, François Durbin*, Thierry Garié*, André Tissot*

Laboratoire LIP6/ASIM UPMC Paris VI, 4 Place Jussieu, 75252 Paris Cedex 05
*CEA-DAM, SEIM, Centre d'Etudes de Bruyères-le-Chatel, BP 12, 91680 Bruyères-le-Chatel

ABSTRACT - *In this paper, we present some pattern recognition algorithms based on the Gabor filtering. We then select an algorithm showing best performance and readiness for a hardware implementation. We focus this paper on the problem of algorithm/architecture adequation. We present a design methodology considering algorithm study and refining, hardware resources, data widths and final cost of the system implementing the considered algorithm. The whole system is simulated using a hardware/software cosimulation tool and will be implemented following a macro-block generator design strategy. This is of great importance as to the emergence of the new techniques of IP block design. The design will thus be portable, upgradable and parametrizable.*

1. Introduction

It is difficult to extract from an algorithm the topological regularity and the parallelism inherent to it for a specific hardware implementation without reformulating or maybe modifying the algorithm. This fact is evident in the applications of artificial vision. In such applications the computation process goes usually from a 2-D data structure to end up with a list of attributes used for the recognition. From a theoretical specification of the algorithm we have studied its hardware implementation. The challenge was then to transform or readapt the algorithm to make it adequate for an optimal hardware structure without affecting its performance and capacity to resolve the problem. This process makes by itself a whole discipline called Algorithm-Architecture Adequation.

Artificial vision systems must take into account several types of irregularities and deformations that may affect the object to be analyzed. These variations are orientation, scale, translation, perspective, lighting and occlusion.

In our system the information extraction relies on the Gabor filtering [1]. The corresponding filters are constructed from dilatations and rotations of a Gabor type mother wavelet.

Let's remind the definition of the impulse response of a Gabor filter

$$G(x, y, \sigma, k_x, k_y) = e^{\frac{-\left(k_x^2+k_y^2\right)}{2\sigma^2}\left(x^2+y^2\right)} \bullet e^{j\left(k_x x+k_y y\right)}$$

where σ = spatial resolution of the filter and k = spatial frequency. From each point on the original image we extract a corresponding activity. These activities will build up a representative vector of the point and its immediate neighborhood. The spatial frequencies k and the rotations $\theta = arctg\left(\frac{k_y}{k_x}\right)$ will be defined later in the paper.

2. Global Recognition System: Algorithmic Aspects

The global recognition system relies on three distinct but related modules:

- The reference images library
- The Gabor filtering
- The pertinent information extraction
- The classification and decision by an artificial neural network (ANN).

The recognition of an item is carried out in two phases, a learning phase and a classification and decision phase.

The learning phase consists of finding the state of the ANN which permits the storage and separation of the examples of the reference image library. Each image undergoes the whole processing through the system to extract its signature. This one is then fed to the ANN.

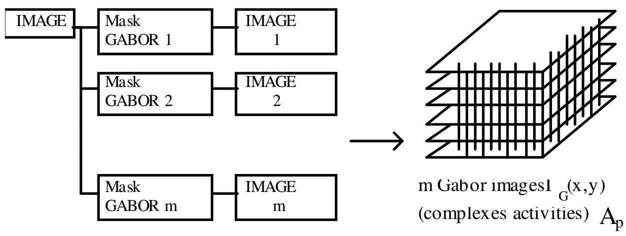

Figure 1. Computation of the Gabor Activities

It is necessary to precisely choose the filters characteristics (spatial frequency and orientation). Getting the m Gabor images, is equivalent to getting a vector of m components for each pixel of the initial MxN image. Thus we must store MxN vectors each of which is of m components (fig. 1). These MxN vectors are too many to be easily exploitable. This is why it is necessary, at this stage of analysis, to extract those that best characterize the initial image.

2.1 Image Characteristic Activities Extraction Methods

There exist in the literature several methods for the extraction of pertinent information in an image [2,3]. We will present here the chosen method that is the Clark & al method [2]. It is based on the saliency notion. It has been chosen for its adequacy for a hardware implementation (this constitutes the first algorithmic choice guided by the architecture) in regard of the data precision and relative regularity of the data processing in general.

2.1.1 Clark & al Saliency.

The saliency is defined to be a measure of the specificity of the information in a vector of Gabor activities.

We define the normalized and the non-normalized saliencies of two vectors V_i, V_j by these two formulae:

$S(V_i, V_j) = \frac{V_i \bullet V_j}{\|V_i\|\|V_j\|}$ for the normalized saliency corresponding to the cosine of the angle between the two vectors.

and $S_u(V_i, V_j) = V_i \bullet V_j$ for the non-normalized saliency.

The saliency of a vector (normalized or not) is defined as $S(V_i) = \sum_{V_j \in 3x3} S(V_i, V_j)$, for a certain neighborhood.

This quantity is defined for all the vectors at all points of the image. The vectors are then ranked in the descending order according to their non-normalized saliencies. This list must be shortened considerably. Eliminating the vectors of saliencies smaller than a certain threshold will eliminate all the points corresponding to the background of the image which contains no significant information relative to the object of interest. However the pixels close to the object require further processing. The spatial extent of the Gabor filters is inversely proportional to the spatial frequency. Thus, for the low frequencies, we filter the images with bigger masks that generate Gabor activities of big values. And the corresponding jets cannot be removed by the previous processing.

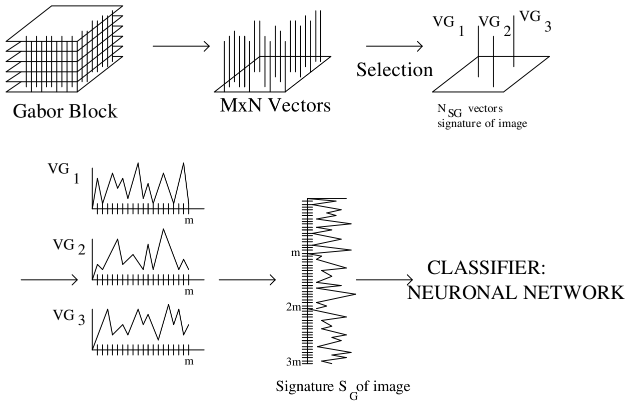

Figure 2. Clark & al. Saliency Algorithm

We define a ratio R between the first m1 components of a characteristic vector corresponding to the higher spatial frequencies (low extent), and the m1 components corresponding to the low spatial frequencies. If this ratio is inferior to a certain threshold then we can suppose that the big value of the saliency of that vector is due to the spatial extent of the filters rather than that the vector belongs to the object of interest in the image. So we do not keep this vector. In the opposite case, the pixel under study belongs to the object and we keep the associated value. The choice of the threshold defines the number of eliminated vectors, and their proximity to the object.

The above selection is still insufficient and must be enhanced. We now seek to keep a vector and only one in a certain neighborhood (3x3 for example), we call this paving. The vector of largest saliency will be kept. We apply this method to all vectors remaining after the R ratio criteria. Once this operation finished we are left with only the pixels belonging to the object under analysis.

We must then determine among the remaining vectors the ones that are most significant. First we form clusters of vectors according to their normalized saliencies which evaluates the parallelism of the vectors. Second we keep for each cluster the vector having the largest non-normalized saliency. At the end of these selections only a reduced set of vectors is kept. These when concatenated will build up the signature of the image (fig. 2).

3. Hardware Implementation

3.1 Introduction

Most of the image processing systems are build up around ASICs, general purpose DSPs or DSP accelerator boards associated to a host processor.

If we use a general-purpose processor for a certain image processing we must worry about the bandwidth, the cache memory and addressing problems. Dedicated architectures will always be a good alternative. Especially for embedded mass-produced systems [4, 5].

Dedicated architectures offer three main advantages: high speed processing with low latency, implicit and easy addressing and the readiness to upgrade and enhance performance.

3.2 Design Methodology

In order to design an optimal architecture that implements a specific algorithm with speed and silicon constraints, it is necessary to follow a design methodology associating intimately the algorithm/architecture aspect. The methodology we present here takes into account the hardware constraints early in the design process. It relies on the following steps:

3.2.1 *Functional Validation of the Algorithm*

The first step of the algorithm architecture adequation consists of validating the algorithm functionally. A software validation is necessary. The algorithm is implemented using a high-level general-purpose language (like C) and executed on a general-purpose machine. This step is essential. It ensures that the algorithm solves right the problem at hand. At this stage no knowledge or reflection is necessary about the architecture. The software validation is not necessarily an easy step, especially in image analysis in which the programs are executed on a large set of image databases.

3.2.2 *Numerical Validation of the Algorithm*

The algorithm being functionally validated, it is necessary now to validate it numerically. This constitutes the first interaction with the architecture. In fact in a hardware architecture the data widths are limited. It is important to know the data dynamics and types needed in the processing. This step consists of rewriting the functional model in a high level language but now controlling constantly the data types at the input/output and in the intermediate computations. At this level we get some interesting facts about the architecture and the first modifications of the algorithm can take place. In fact, we will try to modify the algorithm in order to guaranty reasonable data widths for a hardware implementation.

This process can be very long and may lead to important modifications of the algorithm.

3.2.3 *Necessary Hardware Resources Identification*

Once the data have been correctly resized, we identify the hardware resources necessary for the implementation of the algorithm (memory, arithmetic operators, communications, ...). These resources are directly derived from the algorithm relying on the results of the analysis in the previous phase.

3.2.4 *Architecture Proposal and Hardware/Software Codesign*

The algorithm being validated functionally and numerically, and the hardware resources identified, we start the crucial part of the design process. We now study and evaluate many architectural alternatives. This process is iterative, because now the constraints of the system are taken into account. Many architectures will be studied and analyzed. At the end of this phase an optimal hardware/software system will be elected.
We will at this stage partition the design into hardware/software parts. Some functionalities will be best suited for a software implementation whereas others would make a better compromise or will be more efficient if realized in hardware.
Our complete system will be designed around the PI-bus architecture [9]. A RISC core, the MIPS R3000, is used as a processor for managerial and software duties.
This step defines a complete system with all the specifications of the macro-cells realizing each function of the algorithms.
The next step consists of describing the system level architecture and simulate it using the cycle-precise simulator PI-sim developed in our laboratory [10].

3.2.5 *Simulation and Software Validation of the Architecture " emulation "*

The previous phases end up with a hardware/software architecture that must be validated before going to the physical realization. The first validation consists of writing a model of the architecture. The model is written in a high level language that implements the architectural aspects using procedural operators and which describe the cycle-precise behavior of the system.

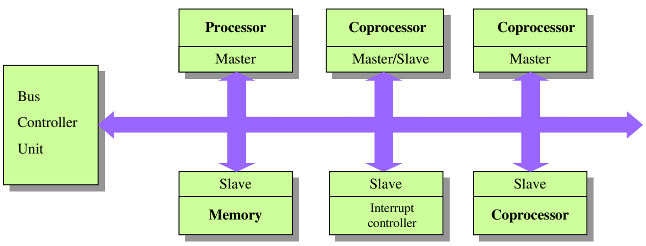

Figure 3. Typical PI-bus system

The PI-bus is a multi-master on-chip bus. This standard is developed jointly by many European leader companies. It furnishes a standardized communication interface for communication between the macro-blocks also said co-processors [9]. A simulator supporting this bus architecture has been developed. The simulator is cycle precise and its main advantage is its speedup against VHDL simulators. It allows us to monitor cycle by cycle our architecture. A typical PI-bus system is depicted in figure 3. We test the system using the same benches of the functional validation phase.

3.2.6 *Hardware Implementation and Validation*

Having chosen and validated the architecture to be implemented, we now go for the physical implementation. Our VLSI design methodology relies on the virtual macro block generators. This methodology is implemented in the CAD design environment GenOptim [6, 7]. GenOptim is a VLSI CAD tool allowing a very fast implementation of architecture ideas on different technologies and different commercial sets of CAD tools. It furnishes the designer a set of high level C functions allowing to write quickly the architecture, the test patterns and the placement. During the mapping on the wanted technology placement and timing optimization operations are carried out by GenOptim, see figure 4.

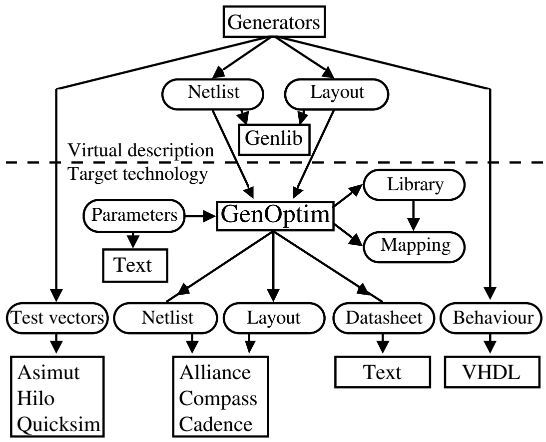

Figure 4. GenOptim description.

Generators created with this tool can be mapped on any standard cell library as well as FPGAs. This shows to be of great interest in regard of the new trends in design with IP blocks.

3.3 Hardware Implementation of the Recognition System

The complete system architecture comprises
- An acquisition module
- Memory to store Gabor filtered images
- A processor, a MIPS R3000 RISC
- Coprocessors for the convolution, saliency and neural network

The complete system is depicted in the next figure.

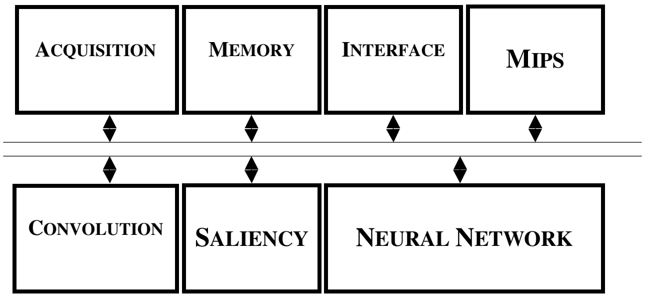

Figure 5. The complete system architecture

The communication between the components in parallel permits us the realization of a macro-pipeline. In fact, the convolver can get its data on the fly from the external interface through a traffic generator feeding the input FIFOs. Then the saliency, R criteria macro-blocks get their data through the output FIFOs of the convolvers. The final signature extraction needs the maximum saliency value. This breaks the pipeline. We implemented a 3 stage macro-pipeline depicted in the figure below (fig. 6).

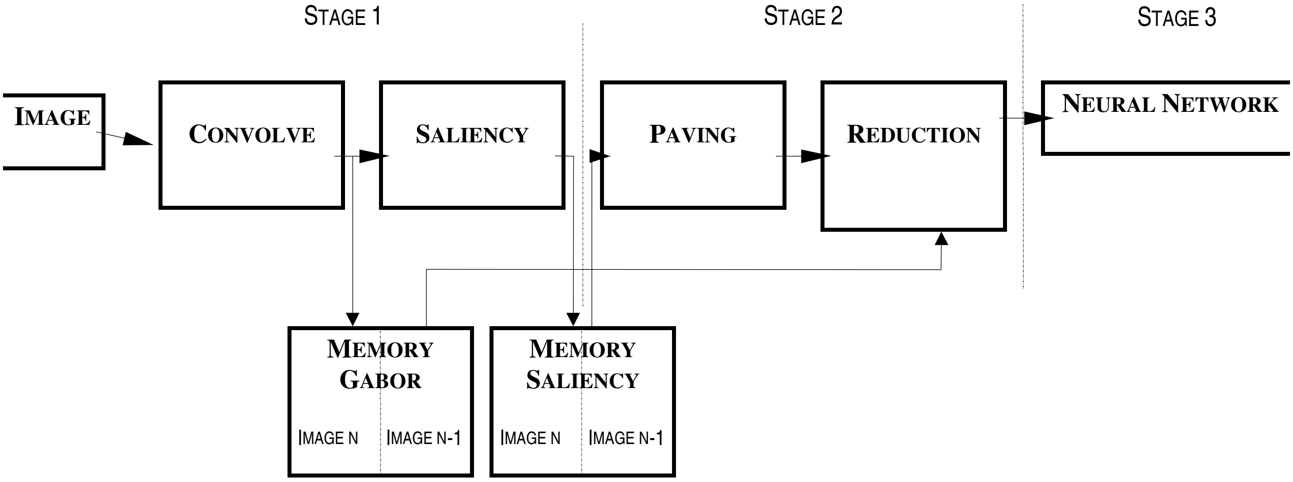

Figure 6 : Macro-pipeline of the Complete Architecture

The recognition method under study necessitates the convolution of an input image with a certain number of Gabor filters. A VLSI ASIC convolver has been designed and fabricated for our application [8]. This convolver is also available as a macro-block for instantiation as a module in a larger system, it was fully designed under the GenOptim environment. The Gabor filters can be as large as 31x31 values. The architecture of the convolver used some flexible and performant concepts: use of systolic arrays, vectorization of data, cascadability and finally reconfigurability. The convolved images must be memorized for subsequent processing.

It is not possible in this paper to discuss the architecture of all the modules. In the next section we take as an example the saliency module and present its implementation.

3.4 Hardware Implementation of the Saliency

Factorizing the saliency computation formula of a vector.

$$S(V_i) = \sum V_i \bullet V_j = V_i \bullet \sum V_j$$

the computations can be simplified, the multiplication will only be executed after the accumulation of the components of the vectors. So we get the systolic architecture depicted in figure 7.

In this figure two saliency blocks are shown. Each block computes the saliency on a single plane. We need 24 such blocks for the complete system.

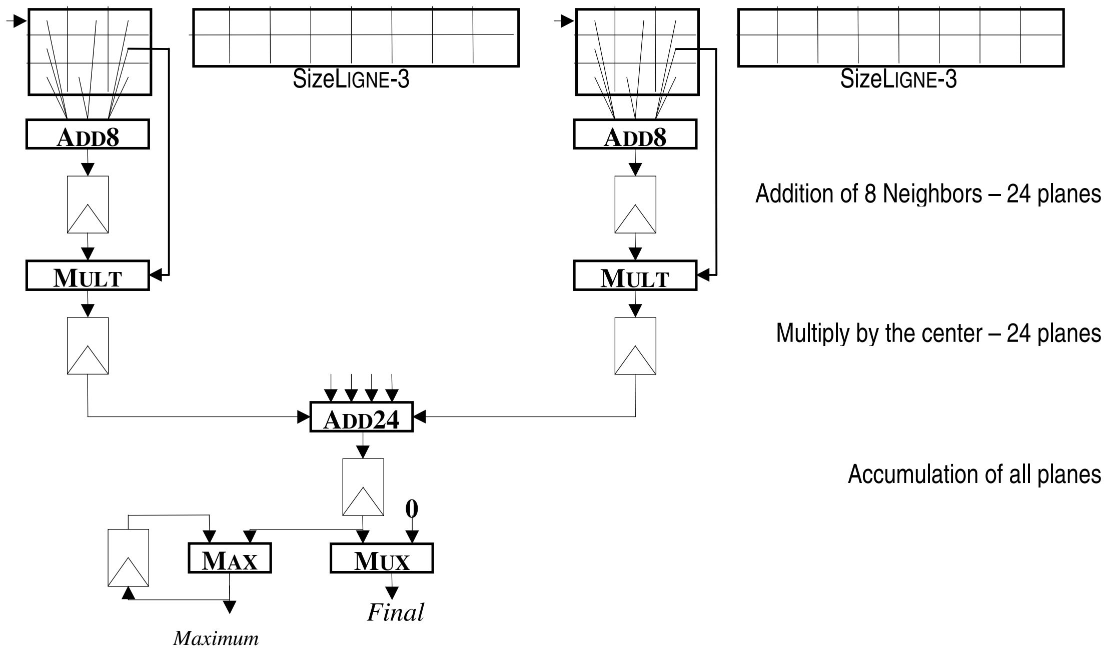

Figure 7. Architecture of the Saliency Coprocessor

3.5 The Complete System Simulation

The whole system is presented in figure 8. Beyond the study and realization of the different coprocessors, a hard challenge imposed to us was the definition of all the communications. In fact in this system most of the processing is carried out by different components each having a very precise functionality. All the communications is achieved through FIFO interfaces. Each coprocessor has an interface to the bus. A Master interface for the MIPS, a salve interface for the memory and Neural Network, and a Master/Slave interface for the convolution, saliency, paving,…The MIPS has been kept to carry out a minimal amount of computations. It is used as a manager of the whole system. It issues orders to the coprocessor and processes the different exceptions and interruptions. Many software routines are executed by this processor, for example the learning algorithm of the neural network. Since the saliency and the R criterium coprocessors use the same data, and they must write the same data relative to the same pixel at the output, they have been tied together in the same component. Instead of writing the result of the R criterium in memory, it is used directly to invalidate the pixel with disregard of its saliency. The paving and reduction coprocessors output a reduced number of vector positions to be fetched from the Gabor memory and fed to the neural network for learning or classification. We cannot go through the details of the architecture of the neural processor in this article du to its shortness.

4. Conclusion

In this paper, we have presented a real case of the Algorithm/Architecture adequation and system level integration problem.

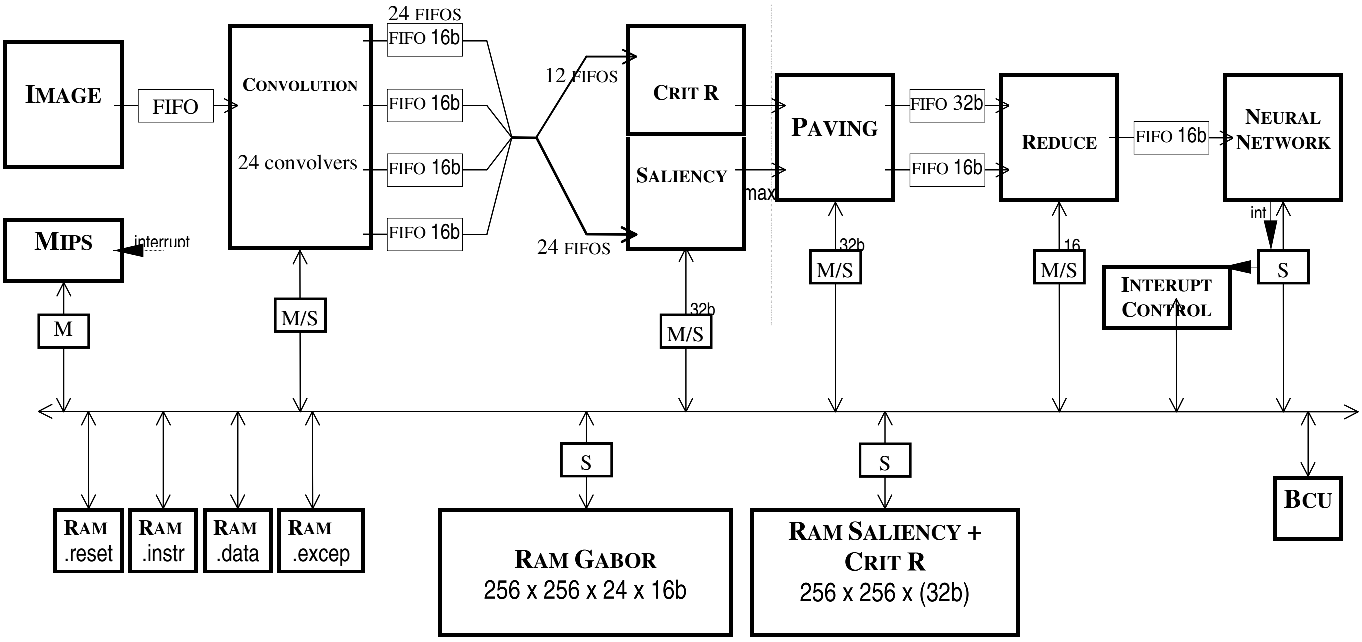

Figure 8. The complete System Architecture

A deep study has been necessary to modify and simplify the algorithms for an optimal hardware implementation keeping a reasonable successful recognition ratio. We have proposed a methodology that takes into account early in the design stages the implementation constraints. The implementation methodology relies on the module generator concept and the use of a cycle precise simulator to simulate the complete system. A VLSI chip for convolution has been designed successfully using this methodology. As seen through this paper, we have concentrated our approach on the speed/silicon optimization, a very interesting point is to add a third dimension: power.

4.1 References

[1] Gabor, D., *Theory of Communication*, Journal of the IEE 93, Part III, no. 26, pp. 429-457.

[2] Clark, G., & al, *Gabor Transforms and Neural Networks for Automatic Target Recognition*, Workshop on Neural Networks, 1991.

[3] Flaton, K. A., & al, *An Approach to Image Recognition Using Sparse Filter Graphs*, IJCNN 1, pp 1313-1320, 1989.

[4] Child, J., *Image Processing Architectures Take New Twists*, Computer Design Magazine, March 1997.

[5] Ruetz, A., & al, *Architectures and Design Techniques for Real time Image Processing IC's*, IEEE Journal of Solid State Circuits, vol sc-22, No.2, 1987.

[6] Houelle, A., *GenOptim: un environnement d'aide à la conception de générateurs de circuits portables optimisés en performance et en surface*, Thèse de doctorat de l'université Paris VI, 1997.

[7] Vaucher, N., *Méthodologie de conception d'architectures VLSI génériques appliquée au traitement numérique*, Thèse de doctorat de l'université Paris VI, 1997.

[8] Aberbour, M., *A Real-Time Image Convolver,* International Conference on Signal Processing Applications and Technology, Boston, Oct. 1996.

[9] Open Microprocessor Systems Initiative, Draft Standard, OMI 324: PI-Bus, Rev. 0.3d

[9] Pétrot, F., Hommais D., Greiner A., *Cycle Precise Core Based Hardware/Software System Simulation with Predictable Event Propagation*, Proceedings of the 23rdEuromicro Conference