The Abingdon Cross Benchmark on the two PAPRICA Systems*

Alberto Broggi
Dipartimento di Ingegneria dell'Informazione
Università di Parma
Parma, ITALY, I-43100
broggi@CE.UniPR.IT

Abstract

This work presents the results of the Abingdon Cross Benchmark on the two PAPRICA systems, which are here briefly described focusing especially on their different architectural solutions. Starting from some considerations on their computational paradigm and hardware characteristics, the most efficient algorithm for the Abingdon Cross benchmark is discussed.

The algorithm has been implemented on the two systems taking advantage of specific ad-hoc *optimizations; the final* Quality Factor *is derived for both architectures and compared against that of other different non-commercial systems developed worldwide.*

1 Introduction

The PAPRICA project [5, 4, 7] started in 1988, aimed to the development of an experimental VLSI architecture devoted to the efficient handling of data with a two-dimensional structure. The main goal of the project was to develop a subsystem that could operate as an attached processing unit to a standard workstation and, in perspective, as a specialized *low-cost* processing module in dedicated systems devoted to low level image analysis, and/or cellular neural networks emulation.

The two PAPRICA prototypes follow the SIMD classification: both of them are composed of a high number of Processing Elements (PEs), each one processing different data according to the same algorithm. The computational paradigm implemented on such systems is based on binary mathematical morphology; in other words, the processing of 2D non-binary data structures is performed one-bit at a time. In order to obtain always the maximum efficiency from the hardware, the PAPRICA systems feature in fact single-bit processing units, as well as the possibility to execute more than one operation in pipeline. Moreover, since one of the goals of the project is to keep the cost of the system low, the architecture cannot integrate a large number of PEs, and each PE can only own a small amount of internal memory.

Generally (e.g. in image processing applications) the number of PEs is far smaller than the number of image pixels, thus requiring a sort of *virtualization* of the Processing Array (PA). This is done utilizing an external memory [13, 9, 11] (hereinafter referred to as Image Memory, IM): the PA is loaded from the external memory with a sub-window of the data set; then the computation is performed until a special instruction is reached; finally, the results are stored back again into the external memory. These steps are iterated until all the sub-windows have been processed. Then the first sub-window is reloaded again into the PA and the computation is resumed. According to this solution, data flow back and forth from the IM to the PA and viceversa through the data-bus.

The logical organization of the data-bus, the external organization of the PEs, and the size of their accessible neighborhood are the features that differentiate the two considered PAPRICA prototypes.

1.1 The First PAPRICA Prototype

In the first PAPRICA prototype, the PA is composed of an array of 4×4 full custom ICs (1.5 μm CMOS, 45 mm^2, $\simeq 35000$ transistors), each of them containing a sub-array of 4×4 PEs. In the present implementation, the PA is a 16×16 square matrix of 1-bit PEs each one with full 8-neighbors connectivity [9, 6, 11]; each PE has an internal memory composed of 64 bits;

*This work was partially supported by CNR Progetto Finalizzato Trasporti under contract number 91.01031.PF93.

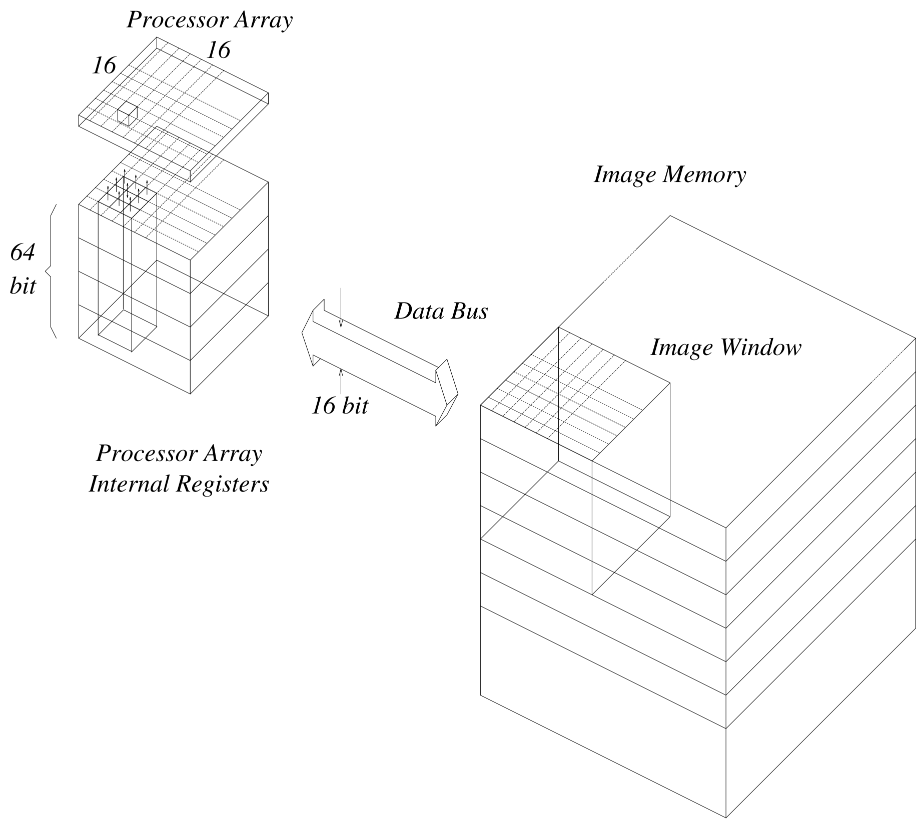


Figure 1: The logical organization of the first PAPRICA prototype

data flow between the IM and the PA through a 16-bit data bus. A block diagram depicting the logical organization of the PAPRICA-1 prototype is shown in figure 1.

In the current implementation, the PAPRICA co-processor is integrated on a single VME board (6U) connected to a SPARC-based workstation. It comprises 5 major functional parts:

1. the Program Memory, storing up to 256k instructions;
2. the Image Memory (up to 8 MBytes of static RAM), whose 16-bit memory fetch time is 150 ns;
3. the Processor Array, whose PE cycle time is 250 ns;
4. the Frame Grabber device, able to grab and store into PAPRICA Image Memory 512×512 8 bit/pixel grey-tone images at video rate (25 frames/s);
5. the Control Unit, which manages the activity of the whole system.

1.2 The Evolution of the Project: PAPRICA-3

The intensive use [1, 2] of the first prototype as well as some theoretical studies [7, 3] have pointed out some architectural problems.

- The low amount of internal memory does not allow to store internally a large amount of intermediate results, as often needed by computation intensive problems, thus requiring I/O operations to store the intermediate data in the external memory.

- The data transfer between the IM and the PA takes place on a 16 bit data bus: in a single cycle 16 bits (belonging to the same image pixel) are transferred in parallel. Unfortunately in many cases (almost all the real ones) the word parallelism of the data to be transferred is not a multiple of 16. This means that a significant amount of time is wasted transferring unnecessary data back and forth from the IM to the PA.

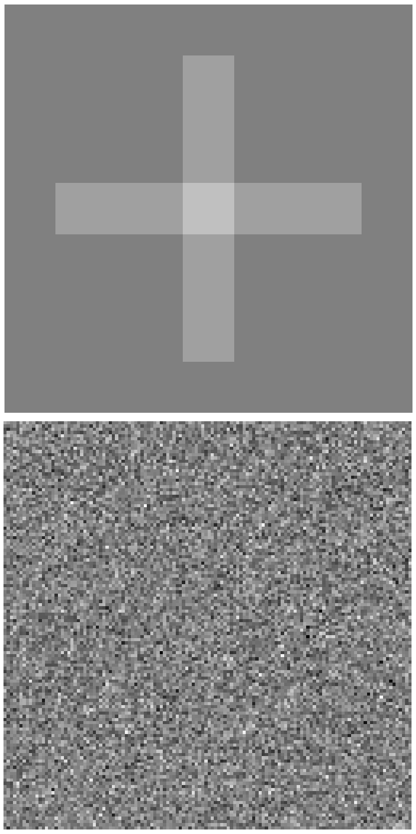
- The first prototype was conceived to run DRC algorithms and thus its instruction set follows a sort of CISC-oriented implementation, including also specific instructions needed to solve that kind of tasks. The evaluation of the performances has led to the consideration that the instruction set should evolve toward a RISC implementation. Moreover, since the complexity of the instruction set determines the PE size, the RISC choice allows the integration of a higher number of PEs on a single chip.
- Finally, the possibility to enlarge the neighborhood involved in morphological operations has been investigated.

The above considerations led to the design of a different architecture: as shown in figure 2.a, the PAPRICA-3 system features a *linear* array of Q single-bit PEs, connected to an external image memory via a bidirectional Q-bit bus. Therefore each memory access transfers a complete vector of Q pixels at a time, 1 bit per pixel. The rationale behind this system is that the size of the PA matches exactly the width of the input image, thus reducing the PE virtualization mechanism problem which has been proven to be a critical design issue in 2D arrays [7]. The foreseen values of Q for the first hardware implementation (now under development) will range from a minimum of 64 to 256.

Data are transferred into the PEs internal registers, processed, and explicitly stored back into memory according to a RISC-oriented processing paradigm. The PE neighborhood has been augmented to include also the 4 cells in the 4 main directions at a distance 2 from the central cell (reduced 5×5 neighborhood, as shown in figure 2.b). For this reason a number of registers (*Morphological Registers (MOR)*) have a structure which is more complex than that of a simple memory cell and are in reality composed of 5 1-bit cells with a `S`$\rightarrow$`N` shift register connection. Besides the MOR, each PE owns also a number of *Logical Registers (LOR)* which may be used for intermediate storage and for all operations which do not require the use of neighboring values.

2 The Abingdon Cross benchmark

The main target of the Abingdon Cross benchmark [10] is to compare the performance of different computer architectures devoted to image processing.



(a)

(b)

Figure 3: *a*) synthetic cross; *b*) white gaussian noise.

This benchmark has been defined to be implemented on any computer architecture, regardless of its computational paradigm. The target of this benchmark is the determination of the computational time required by the filtering of a 8 bit/pixel test image; the filter consists of the determination of the skeleton of a synthetic cross (figure 3.a) added with white gaussian noise (figure 3.b).

There are no constraints at all on the implementation of the Abingdon Cross benchmark, but obviously the best performances are achieved when the algorithm matches the computer architecture and the computational paradigm. Moreover the design of the algorithm can take advantage of the knowledge of the key parameters used in the synthesis of the test image.

1. Number of gray levels: 256 (8 bit/pixel);
2. Linear image size: N;
3. Mean value of the added gaussian noise: 0;
4. Signal to noise ratio: 0 dB.

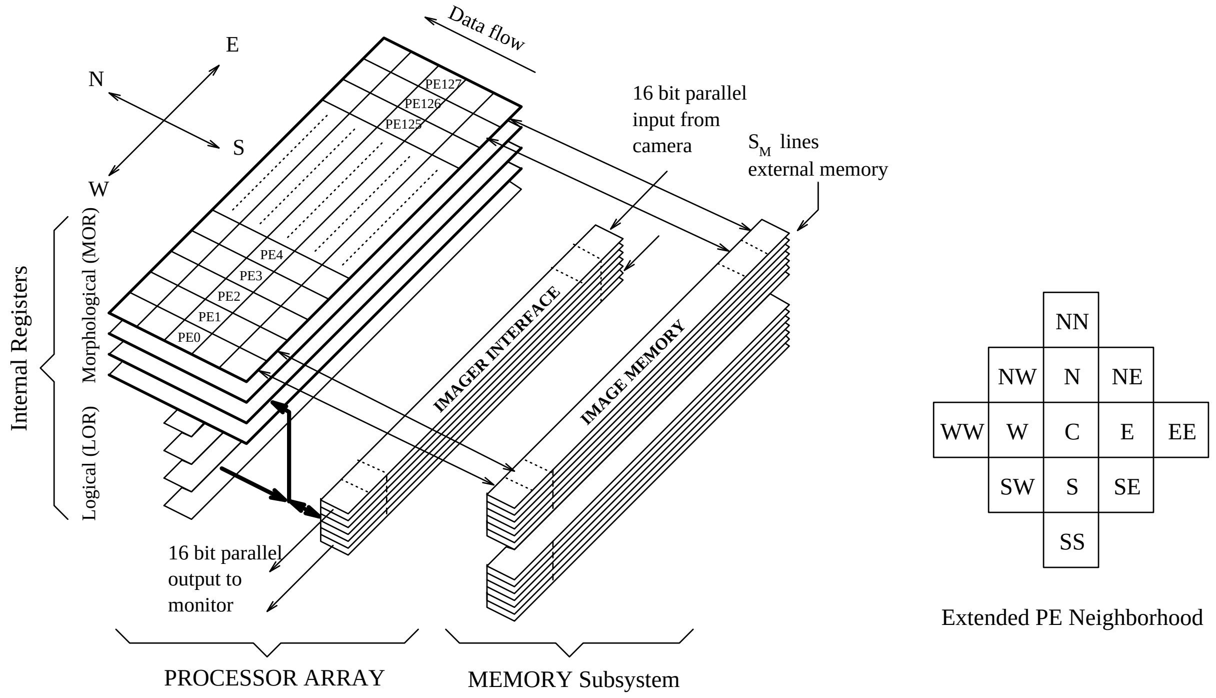


Figure 2: *(a)* structure of the Processor Array; *(b)* PE neighborhood.

5. Size of cross relative to the image size: arm length: $\frac{3}{4}N$; arm width: $\frac{N}{8}$;

An example of a test image with $N = 128$ generated according to the specifications described in [15] (256 gray levels; background level: 128; cross arms level: 160; arms intersection: 192; noise standard deviation: 32) is shown in figure 4.a. The final target of the Abingdon Cross benchmark is the determination of the *Quality Factor* performance index, which is defined as the ratio between the linear size of the test image (N, generally 128 or 512) and time required to filter it (T):

$$QF \triangleq \frac{N}{T} \ . \tag{1}$$

Moreover, for commercial systems the *Price-Performance Factor* is also defined, but since the two PAPRICA systems are non-commercial research prototypes, only the Quality Factor is considered.

3 The Algorithm

Since both PAPRICA-1 and PAPRICA-3 systems are composed of single bit PEs, the processing of binary images is preferred with respect to the handling of data structures with higher word parallelism. Moreover, due to the specific instruction set implemented on such systems, high performances are obtained when the algorithm is based on morphological [8, 14] operations, namely when only the configuration of a reduced neighborhood is considered in the processing of each image pixel.

The above considerations lead to the subdivision of the algorithm into two main steps:

(a) image binarization, and

(b) morphological processing.

3.1 Image Binarization

Concerning step *(a)*, different approaches have been considered, including the application of: (i) a fixed threshold, (ii) an adaptive one (based on a given neighborhood), (iii) a set of two thresholds together with a hysteresis process. However, since step *(b)* plays the most important role in the algorithm, the simplest implementation (fixed threshold) has been chosen for step *(a)*.

3.2 Morphological Processing

As already justified above, the second step of the filter consists of a set of morphological operations whose im-

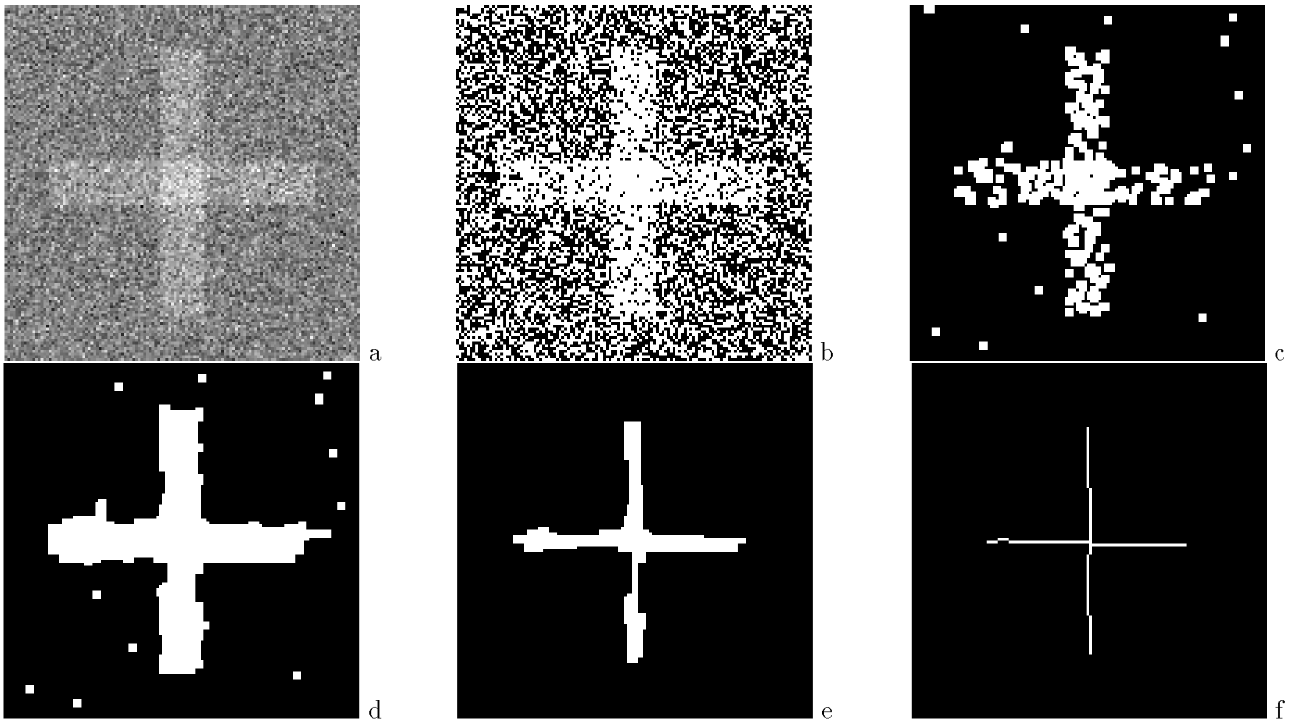


Figure 4: The Abingdon Cross benchmark on a 128×128 image: a) input image; b) thresholded image; c) after one morphological opening with the 3×3 anisotropic structuring element; d) after one morphological closing with a 7×7 anisotropic structuring element; e) after four morphological erosions with the 3×3 anisotropic structuring element; f) after three thinning iterations.

plementation on the PAPRICA systems is extremely efficient. The morphological step of the algorithm is based on:

- a morphological opening [8] (an erosion followed by a dilation) with the 3×3 anisotropic structuring element $\begin{array}{|c|c|c|}\hline \bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet\\\hline\end{array}$, aimed to the elimination of the noise in the background;
- a morphological closing (a dilation followed by an erosion) with the 7×7 anisotropic structuring element $\begin{array}{|c|c|c|c|c|c|c|}\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet\\\hline\end{array}$, aimed to the elimination of the noise within the cross;
- a fixed number of erosions (again with the 3×3 anisotropic structuring element) aimed to the reduction of the arms width. The number of iterations depends on the image linear size: in the case of $N = 128$, 4 erosions are performed.
- finally, a thinning filter [12] is iterated until the image consists of lines with single-pixel width. For $N = 128$, 3 to 4 iterations are performed, depending on the specific noise conditions.

The complete sequence of intermediate results of the algorithm for the Abingdon Cross Benchmark is depicted in figure 4.

4 Benchmarking PAPRICA-1 system

In the implementation of point *(b)*, an optimizing technique [3] has been exploited: it is based on the consideration that the data bus, which links the IM and the PA, has a 16 bit parallelism, thus during the transfer of a p-bit image (with $1 \leq p \leq 16$), the data bus efficiency is $\frac{p \times 100}{16}\%$. A low bus efficiency is measured for example during the execution of the thinning filter, since both the data loading and the data saving involve the transfer of a single-bit (binary) images over the 16-bit data bus; this produces the lowest bus efficiency (about 6%).

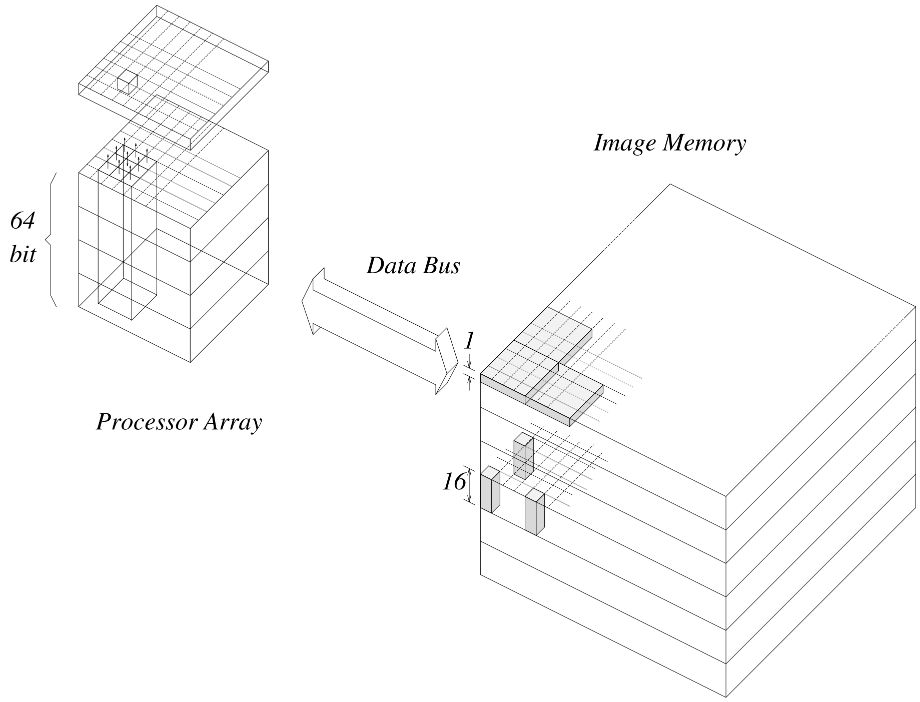


Figure 5: The result of the data packing procedure

The goal of the optimization technique is the improvement of the data bus efficiency, which is achieved by a data packing procedure [3]. The data packing, performed immediately after the thresholding step, reads the result of the binarization process (a $N \times N \times 1$ image) from the PA and writes a $\frac{N}{4} \times \frac{N}{4} \times 16$ image into the IM. Besides improving the bus efficiency (now during the saving of the binarized result into the image memory a single data transfer consists of 16 bit, and thus the bus efficiency reaches 100%), it also decreases the image linear dimensions from N to $\frac{N}{4}$, and increases the pixels' neighborhood accessed by an individual read instruction (for details see [3]). An example of the packing of 3 image subwindows is depicted in figure 5, which shows that the data of a single sub-window are stacked in 16-bit columns.

The total processing time for $N = 128$ is about 110 ms. The quality factor, as defined by equation (1), becomes $QF = 1.1\ 10^3$.

5 Benchmarking PAPRICA-3 system

The data packing optimization used for the implementation of the algorithm on PAPRICA-1 system was explicitly designed to overcome the low efficiency of the data bus during the transfer of data structures with a parallelism lower than 16. Conversely, the different logical organization of the IM in the PAPRICA-3 system allows to transfer on the data bus only the significant information, thus providing always the maximum bus efficiency. In this case no data packing is needed, and the image is processed in plain format.

Another advantage offered by the architecture of the PAPRICA-3 system is the enhanced neighborhood accessible with a single morphological operation. This fact allows to decrease the number of morphological instructions.

Furthermore, a specific hardware enhancement allows to decrease drastically the number of memory accesses. When a new image line is loaded from the IM to the PA, the morphological registers are shifted one position north; the processing begins and the result is saved back into the IM (see figure 6). When the processing is based on a neighborhood wider than 5 consecutive lines, the above process must be iterated: a filter based on a 9×9 neighborhood needs two iterations.

In order to reduce the number of accesses to the IM, the PA has been enhanced by the possibility to load a new image line also from the logical registers, as shown in figure 7. Thanks to this feature, the intermediate saving and successive reloading of temporary

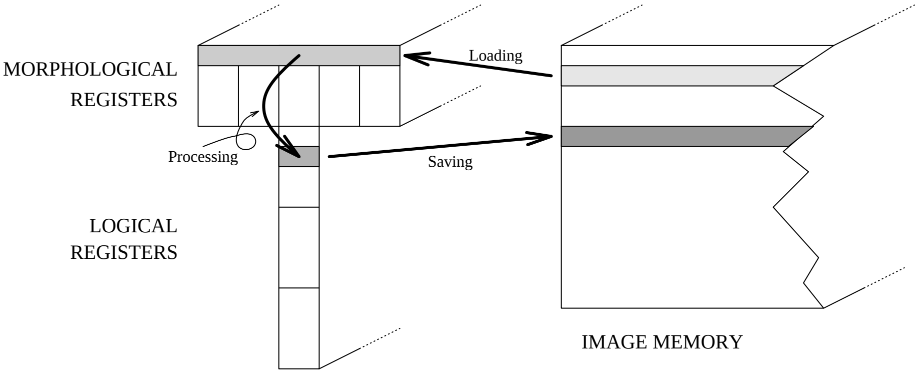


Figure 6: An image line is loaded from the IM into the PA morphological registers, which are shifted one position; the result of the processing is generally stored into the logical registers; and finally it is saved back again into the IM.

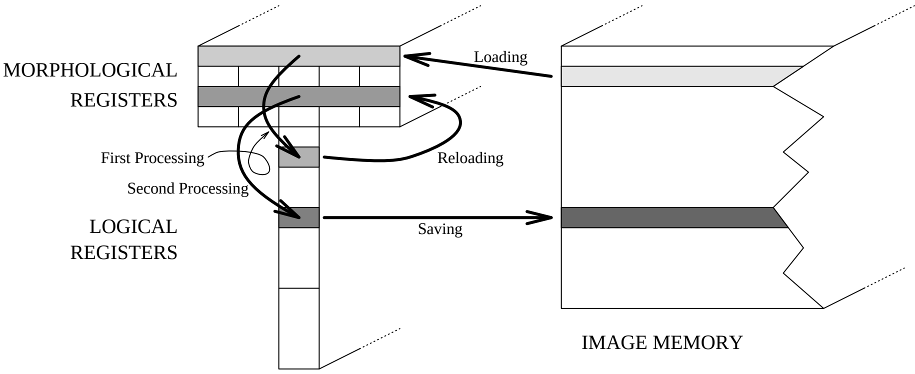


Figure 7: In this case, after the first processing, the result is no more saved back into the IM, but it is reloaded into the PA morphological registers for further processing. This facility emulates the presence of a wider neighborhood.

results to/from the IM are replaced with a faster and more efficient internal data transfer between the PA registers. In fact in this way, if the number of morphological registers is sufficiently high, data remain into the PA chip. Nevertheless, since only 16 morphological registers are available, in the implementation of the Abingdon Cross benchmark, one intermediate transfer to and from the IM is anyway required.

Since the PAPRICA-3 hardware system is currently still under development, a software simulator (at the system level) has been used to derive the Abingdon Cross Quality Factor. In order to produce results comparable to the ones derived for the first PAPRICA-1 system (with $16 \times 16 = 256$ PEs), in the simulator the number of PEs has been fixed to 256, thus the size of the test image is 256×256. Two different results have been obtained, assuming different values for the memory access time T_M and the instruction execution time T_C:

a) a conservative solution: $T_M = 100$ ns and $T_C = 20$ ns;

b) a desirable one: $T_M = 50$ ns and $T_C = 10$ ns.

In case a), the total time required to filter a 256×256 test image is 4.68 ms, thus providing a Quality Factor of about 5.47 10^4. In case b) the time is reduced to 2.34 ms, while the Quality Factor becomes 1.1 10^5.

6 Conclusions

Table 1 presents the results shown in [10] for non-commercial machines only, together with the results for the PAPRICA-1 and PAPRICA-3 systems.

This results are in accordance with the theoretical studies on the architectural solution (discussed in [5]), which forecast a speed-up of 2 orders of magnitude thanks to the different IM logical organization and to

Machine	Quality Factor
CLIP4 (96×96)	$7.3\ 10^3$
Cyto III	$1.0\ 10^2$
DIP	$4.7\ 10$
PHP	7.1
POP II	2.9
VAP	$1.8\ 10$
WARP	$5.0\ 10^2$
PAPRICA-1	$1.1\ 10^3$
PAPRICA-3	$5.47\ 10^4 - 1.1\ 10^5$

Table 1: The quality factor for some non-commercial machines

the possibility to decrease the computational overhead caused by the PA virtualization.

Acknowledgments

The author would like to thank Luca Giovanelli and Gian Luca Porcari for their useful help.

References

[1] A. Broggi. A Novel Approach to Lossy Real-Time Image Compression: Hierarchical Data Reorganization on a Low-Cost Massively Parallel System. *Real-Time Imaging Journal*, 1(2), 1995.

[2] A. Broggi. Parallel and Local Feature Extraction: a Real-Time Approach to Road Boundary Detection. *IEEE Transactions on Image Processing*, 4(2):217–223, February 1995.

[3] A. Broggi. Word Parallelism vs Spatial Parallelism: a performance optimization technique on the PAPRICA system. In *Proceedings 3rd EuroMicro Workshop on Parallel and Distributed Processing*, pages 236–243, Sanremo, Italy, January, 25-27 1995.

[4] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, and L. M. Reyneri. The PAPRICA Massively Parallel Processor. In *Proceedings MPCS - IEEE International Conference on Massively Parallel Computing Systems*, pages 16–30, Ischia, Italy, May 2-6 1994.

[5] A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, and L. M. Reyneri. The Evolution of the PAPRICA System. *Integrated Computer-Aided Engineering Journal - Special Issue on Massively Parallel Computing*, 1995. In press.

[6] T. Fountain. *Processor Arrays: Architectures and applications.* Academic-Press, London, 1987.

[7] F. Gregoretti, L. M. Reyneri, C. Sansoè, A. Broggi, and G. Conte. The PAPRICA SIMD array: critical reviews and perspectives. In L. Dadda and B. Wah, editors, *Proceedings ASAP'93 - IEEE Computer Society International Conference on Application Specific Array Processors*, pages 309–320, Venezia, Italy, October 25-27 1993.

[8] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(4):532–550, 1987.

[9] NCR Corporation, Dayton, Ohio. *Geometric Arithmetic Parallel Processor*, 1984.

[10] K. Preston. The Abingdon Cross Benchmark Survey. *Computer*, pages 9–18, July 1989.

[11] S. Reddaway. DAP - A Distributed Array Processor. In *1st Annual Symposium on Computer Architectures*, pages 61–65, Florida, 1973.

[12] A. Rosenfeld. A characterization of parallel thinning algorithms. *Information Control*, 29:286–291, 1975.

[13] L. A. Schmitt and S. S. Wilson. The AIS-5000 Parallel Processor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):320–330, May 1988.

[14] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982.

[15] W. B. Teeuw and R. P. W. Duin. An algorithm for benchmarking an SIMD pyramid with the Abingdon Cross. *Patt. Rec. Letters*, 11:501–506, 1990.