A MODIFIED SIMULATION ENVIRONMENT FOR RECONFIGURABLE MULTICOMPUTER SYSTEMS IN DIGITAL IMAGE PROCESSING APPLICATIONS

Francisco J. Quiles Flor and Antonio Garrido del Solo
Dpto. de Informática. University of Castilla - la Mancha

Campus Universitario
02002 Albacete (Spain)

ABSTRACT

In our work we improve the EPPI programming environment, which was made in the University of Castilla - la Mancha one year ago. EPPI is a tool for simulating parallel algorithms that runs on a monoprocessor standar computer under the Unix operating system, what makes it easily transportable. This environment provides a set of tools oriented to digital image processing, whereby it is very adequate for performance stidies of parallel architectures and algorithms. Although it was first built to sumulate NETRA architecture [1], it can now admit different topologies, shared/distributed memory configurations and a variable number of processors with floating point capacity and several simulation parameters, providing also information about execution of the algorithm, allowing capacities of debugging and optimization. In this paper, our shows the tool EPPI with a comparison module which allow to compare different architectures and algorithms.

INTRODUCTION

It is quite usual, at present the use of parallel architectures in digital image processing applications, because of the high computational capacity required by a great number of techniques [2]. Moreover, in most low intermediate level applications, the intrinsic parallelism of the operations procedures high speed-up. Since parallel computers are not always available, the simulation of their behaviour on standard hardware computers is very valuable, providing the user an optimous set of specific and general tools for digital image processing.

There are other simulators of parallel architectures [4] [5] [6] [7] [8] for applications of image digital processing which are generally characterized by their need to use computer networks or multicomputer systems as support. VISIX [9] and EPPI [10] [11] are examples of these cases. For the design of EPPI, we have used as support a UNIX (DG/UX) kernel an a computer AViiON 4120 and graphic monitor.

INTERNAL STRUCTURE

The environment presented consist of five modules perfectly integrated, which provide as a whole the simulation capacity above mentioned. Four of them correspond with the original design of EPPI, and the fifth one provides the ability to compare several configurations for a given algorithm. They are as follows:

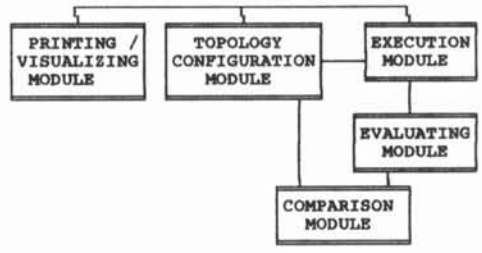

Fig 1: *Modules*

- Printing/visualizing module: It constrains image visualization tools and extraction by printer. It also incorporates specific functions related to image visualization. Such as false colour, palette management, etc. This is a distributed module, and it can be executed on a different computer connected by a local network. The visualization module operates in distributed mode, the exchange of information being carried out by means of its own protocol denominated PEV which, in turn, is supported by a standard RCP protocol. Figure 3 shows its basic structure. The user is provided of functions for image display with printing and other tools, on laser or matrix printer. Figure 2 shows the display system

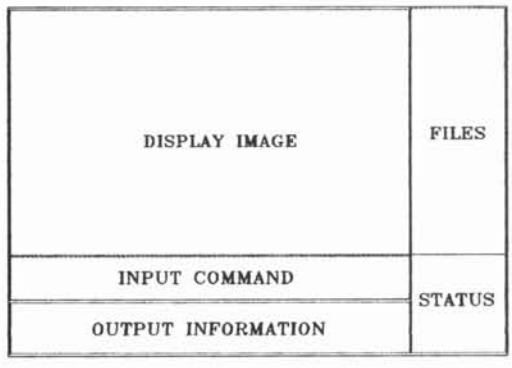

Fig 2: *Display of system*

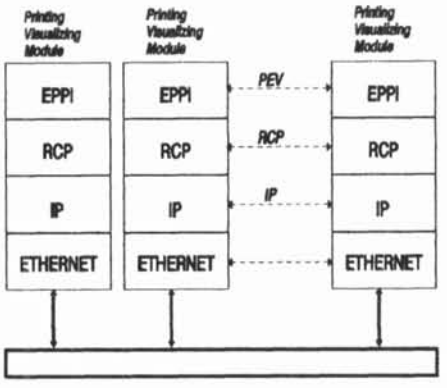

Fig 3: *Structucture of P/V module*

- Execution module: it is responsible for the simulated execution of the code. Using the user's program as starting point and the advanced functions in the library, it accomplishes the parallel execution. The programming model used is the SPMD [12], being also available discriminant functions of processors which allow the processing in SISD, SIMD and MIMD modes. Fig 4 show the internal process in digital image processing.

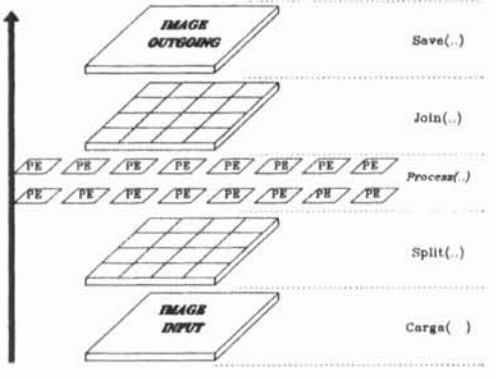

Fig 4: *Internal process*

- Evaluating module: It works in two stages. In the first stage, the information about the execution of a programme on a simulated architecture is registered. In the second stage, the information is analysed generating tables and graphics on different execution parameters. Figure 5 shows the two previous stages. While the simulated algorithm is being executed, the EPPI-ECD module captures the data stored in a file. In the second stage, the EPPI-EAD and EPPI-EGD submodules generate the different evaluation parameters.

- Topology configuration module: It allows the definition of topologies on which the simulations can be carried out. It has a comfortable user interface and generates a configuration file that can be stored in a topology library.

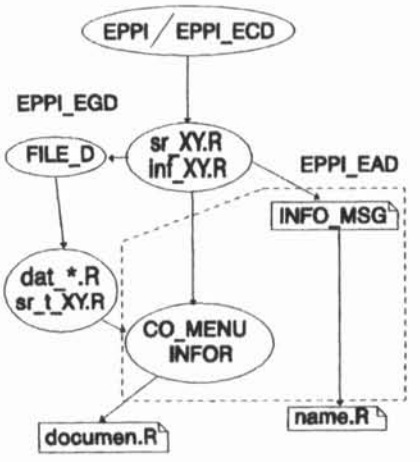

Fig 5: *Evaluating module*

The interface with the user is made up of three windows. The configuration window shows the data of the present configuration (version, topology denomination, number of neighbours per processor). The menu window presents the options available: create, erase, initialize, and see connexion. The introduce window is used to dialogue with the user. The error window is used to inform the user about errors in the introduction of data. Finally, the connexions window shows all the links defined between processors.

- Comparison module: This new module provides the ability of compare several configurations and architectures, giving also information relative to the adequation of the algorithm to the architecture. It is possible to save the results of other executions in a data base, with the aim of comparing the behaviour of a given architecture with different types of algorithms.

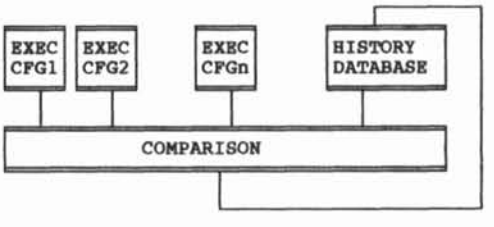

Fig 6: *Comparison module*

DESIGN

Unlike other parallel implementations, our work starts from the duality physical processor-logic processes, simulating the different processors though process of the UNIX operating system. This allows to use in a controlled way the internal resources of the operating system related to virtual memory management, communication between process, etc.

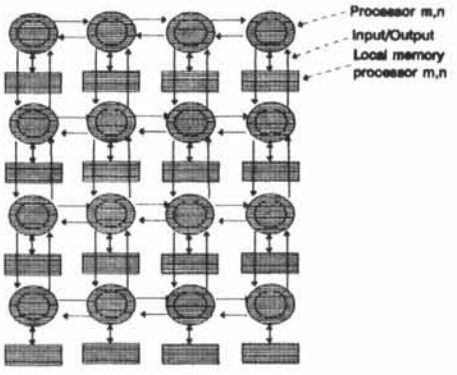

Fig 7: *Example of mesh*

Each process simulated evaluates in a controlled way, being available a global watch of the system and local watches that can be synchronized by means of primitives. The interconnection network is simulated by means of logical channels queues which make use at low levels of thereceiving and sending call of the operating system. In this way the different topologies are simulated by means of a central system of access to the tails from each processor. Is possible to implementation of deadlock detection procedures, based on a resident process which informs the user when and where it would be produced.

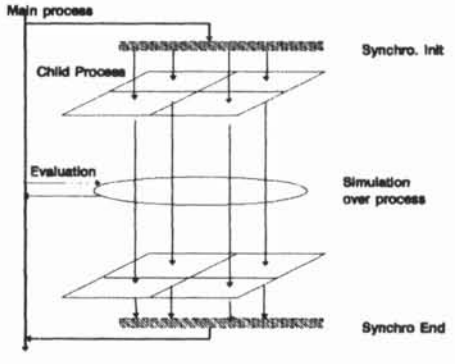

Fig 8: *Synchronism*

A relevant aspect is also that regarding the reconfiguration of the interconnection network. According to our design by only modifying the structure of access control in the network, any topology can be simulated (including total interconnection). The user generally defines (or chooses a topology library) a predefined control structure in accordance with the topology needed. The system includes a tool to make easier the definition of topology. This presents the advantage that by using a stopping function the reconfiguration of the network is feasible by reinitializating the processors according to the new communication pattern. Thus, our environment allows the application of simple reconfiguration algorithms.

The user can define the model of memory desired and its size. As far as the evaluation module is regarded we have chosen to integrate it within the environment, aiming at a greater execution speed, with the minimum influence on simulation. Although this module consumes an important part of the resources of the system (memory, space in disk, processor's time of use), this consumption is not 1 determinant: it is around 5% of total. It presents storing capacity for a great number of parameters relating message traffic, processors and general architecture parameters. It also provides information about traffic in the channels, real latency of origin-destination routes, speedup in function of the number of processors, evaluation of the execution of specific processors, etc.

UTILIZATION

EPPI users must simply introduce the code to be executed, indicating which processor has to be executed. Parameters must be previously established.

Next we will show the result of simulating an tresholding algorithm in a mesh topology with a variable number of processors [13]. Figure 9 shows the original image to which the algorithm is applied and figures 10 to 12 shows the result of the simulation of a variable number of processors. Figures 13 to 15 shows results of simulation.

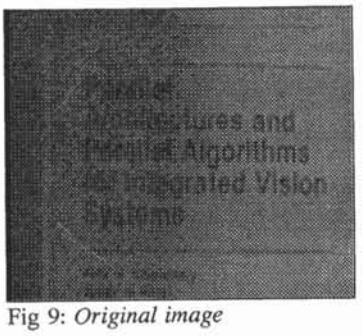

Fig 9: *Original image*

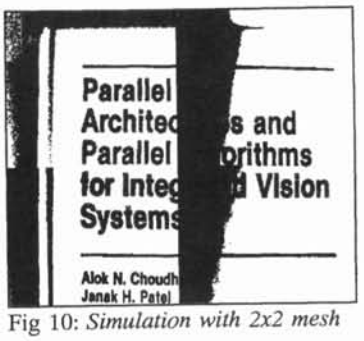

Fig 10: *Simulation with 2x2 mesh*

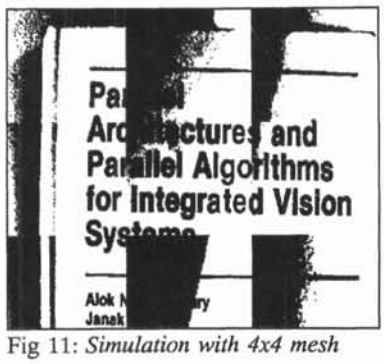

Fig 11: *Simulation with 4x4 mesh*

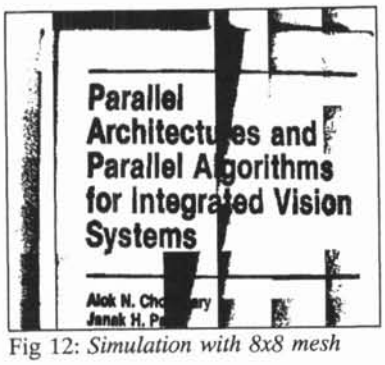

Fig 12: *Simulation with 8x8 mesh*

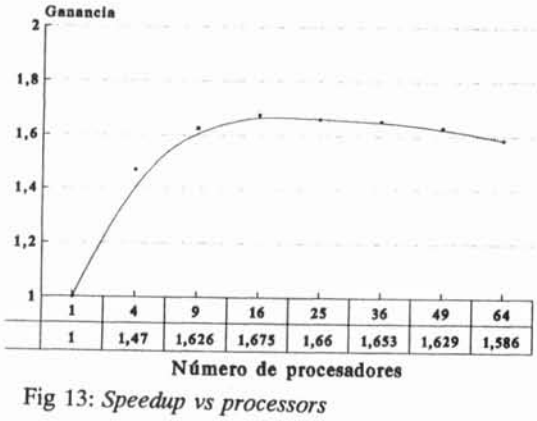

Fig 13: *Speedup vs processors*

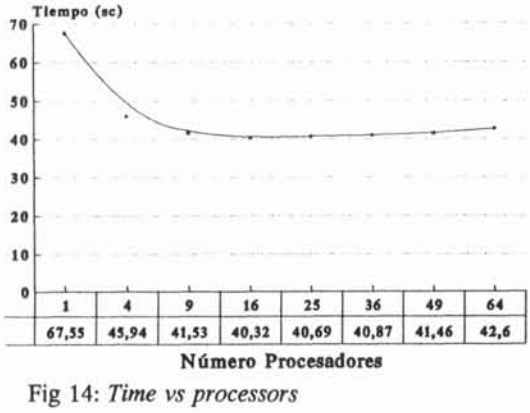

Fig 14: *Time vs processors*

ACKNOWLEDGMENTS

This work was supported by Comisión Interministerial de Ciencia y Tecnología (CICYT) under project PB092-511 and University of Castilla-La Mancha (Spain).

REFERENCES

[1] A.N. Choudhary, J.H. Patel "A Parallel Processing Architecture for Integrated Vision Systems", in 17th Annual International Conference on Parallel Prosessing, St. Charles, IL, pp. 383-388, August 1988.

[2] A.N.Chourdhary, J.N.Patel. "Parallel Architectures and Parallel Algorithmes for Integrated Vision Systems". Kluwer 1990.

[3] A. Choudary and S. Ranka, "Parallel processing for computer vision and image understanding", IEEE Computer, vol 25, nº 2, Feb-1992

[4] B. Bruegge, C. Chang, R. Cohn, T. Gross, M. Lam, P.Lieu, A. Noaman and D. Yam. "The Warp programming environment in Proc.". 1987 Nat. Comput. Conf., AFIPS, Chicago. Il, June 1987, pp 141-148.

[5] G. Marino, G. Succi, G. Levo, R. Pavesio and T. Vernazza "A system-independent tool for developing applications on MIMD architectures", PACTA September 1992, pp. 536-46.

[6] K.M. Nichols, J.T. Edmark. "Modeling Multicomputer Systems with PARET". Computer IEEE, May 1988.pp. 39-48.

[7] L. Snyder. "Parallel Programming and the Poker Programming Environment". Computer IEEE, July 1984, pp.27-36.

[8] Min-You Wu, D.D. Gajski. "Hypertool: A Programming Aid for Message-Passing Systems". IEEE Transactions on Parallel And Distributed Systems Vol. I, No. 3. July 1990, pp. 330-343.

[9] A. Reeves, "Software Computer Vision environments for parallel computers", Parallel Architectures and Algorithms for Image Understanding, Prasanna-Kumar 1991, pp 453-472

[10] F.J.Quiles. "Arquitecturas Paralelas para Visión Artificial". Departamento de Informática. Universidad de Castilla-La Mancha, 1992.

[11] F. J. Quiles, A. Garrido and M. Vicens, "EPPI: Entorno de programación paralelo orientado al tratamiento digital de imágenes", V Simposium Nacional de AERFAI. Valencia 1992, pp 326-336.

[12] A. H. Karp, "Programming for paralelism", IEEE Computer May 1987, pp 43 - 57

[13] D.V. Ramanamurthy, N.J. Dimopoulus ans K.F. Li. "Parallel Algorithms for Low Level Vision on the Homogeneous Multiprocessor". IEEE. Trans. on Computer.1986. pp .421- 423.