

A PC-BASED ENVIRONMENT FOR THE DEVELOPMENT OF SYSTOLIC APPLICATIONS

GHANEMI S.*
HOUCINE M.H.*
DIABI N.*
ZERARI F.*
ZERARI S.*

Abstract

Systolic architectures have been proven to be cost-effective solutions for a large variety of problems, often occurring in signal and image processing. Usually, the development of a systolic solution for a given problem follows the conventional problem solving methodology consisting of the problem specification, the design of a solution and its hard-implementation. It is important to note that the design process which is almost an ad-hoc choice from a family of possible solutions, can be structured into three major steps: the choice of an adequate architecture, its soft-description using the OCCAM language and its validation and performance evaluation on a computer system. Note that much of the work is done on a host computer before its execution on a transputer. In this paper, we present a programming environment based on PC system for the development of systolic solutions. It combines the advantages of two integrated tools: A Graphical Editor and a Syntactic Editeur.

Keywords: Parallel Programming, Systolic Systems, Transputer, OCCAM Language.

1. INTRODUCTION

Recent developments in circuit technology has led to the Very Large Scale Integration (VLSI) which enormously increased the number of electronic devices in a single chip, the size of postage stamp. It is foreseen that up to 1,000,000 components per chip are likely in the near future [7]. Until the advent of VLSI technology, the development of parallel computers with a large number of processors was limited by the unaffordable manufacturing costs. Kunt H.T was the first to realise that the VLSI technology together with the automata theory could be the key success to building high performance parallel computers at a very low cost [6].

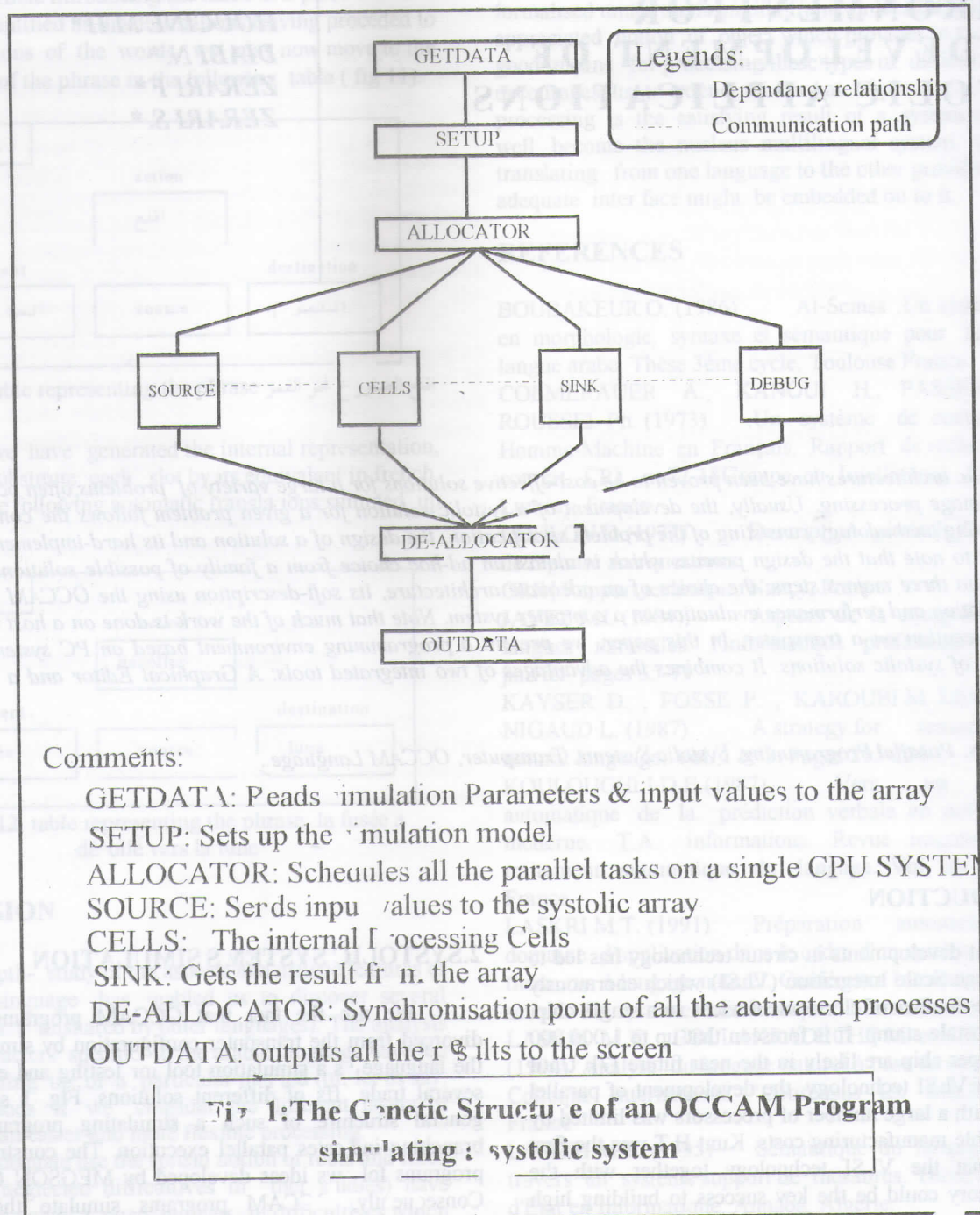
However, the growth in VLSI complexity motivated the development of CAD tools. Such approaches are extremely imperative due to the considerable large number of gates per chip in the current technology. While these CAD software take an algorithmic solution down to the actual hardware implementation, the designer still needs assistance at the design stage.

In this paper, an environment for the development of systolic applications is presented.

2. SYSTOLIC SYSTEMS SIMULATION

We use the fact that OCCAM programs can be divorced from the transputer configuration by simply using the language's simulation tool for testing and evaluation several trade offs of different solutions. Fig. 1 shows the general structure of such a simulating program where branching indicates parallel execution. The construction of programs follows ideas developed by MEGSON G.M. [8]. Consequently, OCCAM programs simulate the formal proofs by replacing I/O description by actual results.

Although the simulation does not guarantee program correctness, it is nevertheless a less consuming approach which does not result in unsolvable equations. Furthermore, a working program retains the possibility of actual implementation and so solves two problems in one attempt. The technique described in the above paper [8] have been used successfully to implement systolic designs in OCCAM [2] and [9].



3. PRESENTATION OF GEPASS

Systolic design which is an ad hoc process heavily relies on common sense and personal experience. Consequently, in order to derive the most efficient and economical solution it is necessary to study and evaluate different systolic architectures that are related to the same

problem. This rather unavoidable step which consumes much of the design costs can only be solved by mathematical equations [8]. Therefore, the solution should be found in the former approach if the involved design costs can be considerably reduced.

A quick look at the different solutions to a given problem reveals that while retaining the same structure of the architecture (i.e. linear, array or hexagonal), it is only the

way in which data and control are routed through the architecture that changes (i.e cell interconnection and data communication). This means that at any given time only a fraction of the implemented program is modified.

In a traditional programming environment, the entire program needs to be rewritten and recompiled several times before getting the final version. Consequently, if we were to follow the same approach we would lose more time than we can save. Therefore, in order to gain more time, it is only necessary to check the syntactic and semantic validation of the added instructions within the program. This has the advantage of considerably reducing the lengthy and repeatedly process of editing, compiling and executing. The approach to use is based on the concept of Syntactic Oriented Editing.

A second improvement which might be of great help to the systolic designer is the use of some dedicated software packages such as the design assisting tools. In particular, coding a given systolic solution into a simulating OCCAM program can be done automatically. Thus, leaving the designer to put more effort into the process of suggesting and deriving alternative solutions.

In this respect, GEPASS (stands for Graphical Editor & Programming Assistance for Systolic Systems) which is a computer Assisted programming tool for soft-systolic systems is developed [3]. It has the advantage of combining the merits of both the graphical and syntactic editors into one integrated software tool. The former, which is a natural way for system interfacing, is used in order to build graphically the architecture of the systolic solution. The latter is needed for eventually expanding internal details of some systolic entities such as the internal processing cells (IPCs).

3.1 GRAPHICAL EDITOR

The graphical editor is used to provide an easy and friendly way of specifying the structure of the proposed systolic solution. It is developed around the Object concept (not to be confused with the object oriented concept). An object which can be either simple or complex is a graphical representation of a systolic entity such as a cell, channel or a port. In order to efficiently manipulate these objects several functions were defined. Examples of such order functions include objects creation and deletion, moving objects around the structure, ...etc. A more comprehensive list of all functions with examples can be found in [4]. Fig 2 which is one of the menus of GEPASS shows some graphical functions.

Because of the simplicity of the entities in systolic systems, only a sufficient and limited number of basic objects are defined. These are geometric figures such as squares, circles and hexagons to represent different types of cells, lines or segments to represent channels and dots to

represent ports. From these simple objects it is possible to form any complex object. For example, a cell with two ports is represented by a square and two dots one on each side of the square. Similarly, complex objects can be combined to create even more complex objects such as a complete systolic architecture.

One major characteristic of graphically designing systolic systems is the problem of the size of the graph (i.e proportional to the number of the involved cells) which might increase considerably so that the screen would not be large enough to fit in the entire structure. Therefore, two additional functions are required: The ZOOM IN and ZOOM OUT functions. The first increases gradually the size of each object so that it becomes possible to see clearly any relevant detail in the architecture. The second function which is the inverse of the first one, allows the designer to see the entire structure while discarding some details.

During the editing stage, GEPASS generates interactively the simplified syntactic tree of the equivalent OCCAM program. Generating the tree instead of the OCCAM program has many advantages. For example, it is easy to update the tree without a complete rewrite of the entire program every time changes are made. However, at the request of the designer, the associated OCCAM program can be easily derived from the tree after unparsing it.

A typical syntactic tree might well consist of a set of nodes, one for every declared procedure, one for the system body and one for the body of the main program. A procedure declaration is represented in the tree by the procedure name and a list of some formal parameters which correspond to the cell's ports. The system is a set of parallel calls to the following procedure: sink, source and the internal processing cells.

3.2 SYNTACTIC EDITOR

Since it was proved that, in practice, it is not possible to design graphically some parts of the proposed systolic solution, for instance the internal definition of the cell, we developed a simplified syntactic Editor for the OCCAM language. It mostly takes in charge only those parts of the OCCAM program that were left unexpanded by the graphical editor. Hence, both the editors cooperate in deriving a complete OCCAM program that is syntactically and semantically correct.

Every entity or node in the syntactic tree is a process (i.e declaration or construction process such as `DEF` or `PAR`) and can be manipulated through a set of commands based on the structural movement of the cursor. Such commands can be issued, for example, to insert, delete, select, move, copy or duplicate an entity. Of course, the syntactic integrity

is always checked by the editor after the execution of any command.

While structural commands are needed to manipulate entities, textual commands are required to expand empty place holders found in entities. An example of a template and place holders is given below:

VAR *<variable_list>*:

PAR

<process>

WHILE *<condition>*

<process>

Where VAR, PAR and WHILE are keywords of the language and between brackets are places holders that need to be expanded.

4. VALIDATION AND EVALUATION

After generating an OCCAM program, it is necessary to validate its logical behaviour and evaluate its performance. Since OCCAM compilers on PCs are not yet available on the market, we developed our own OCCAM interpreter, called MICAM that is Micro based Interpreter for occAM programs [5]. It accepts an OCCAM program as input, generates its associated model based on the process oriented concept, and then interpretes it, giving as output all the necessary results such as performance figures.

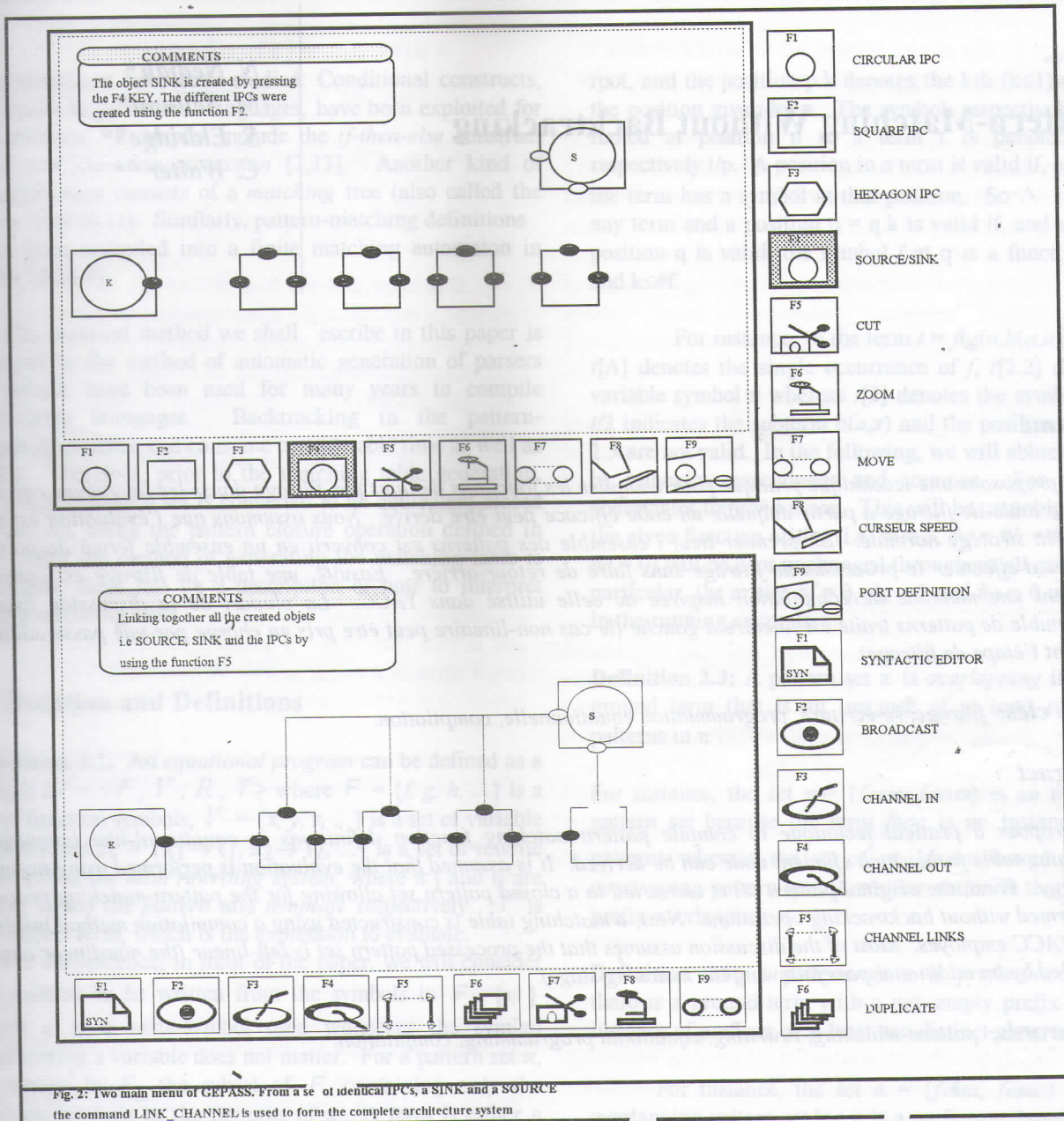
Because of the implementation language used for the interpreter, Turbo PASCAL, MICAM features an important characteristic of portability. Therefore, it could be easily used on any computer system. In addition, it could run independently from the GEPASS environment since it incorporates its own developed modules for syntactic and semantic analysis.

A second feature in MICAM, worth mentioning here, is the fact that it was designed not specifically for systolic systems. Hence, it could be used to simulate and interpret any other synchronous or asynchronous parallel architectures such as SIMD and MIMD architectures. Furthermore, it could be used by undergraduate and postgraduate students to design and evaluate their own synchronisation and communication mechanisms through the use of the OCCAM language.

5. CONCLUSION

The use of CAD tools is well developed in most industrial, manufacturing and engineering fields. However, little work has been done to assist the systolic designer, from the programming point of view. In this paper, we showed the need for sophisticated programming environment specially designed to program systolic systems and presented two software tools: GEPASS and MICAM that can be integrated in this type of programming environment.

GEPASS which is a graphical editor for programming soft-systolic systems is a friendly software package that automatically generates OCCAM programs from their graphical representation. It features several useful commands that can be used to update and modify both the architectural structure and the OCCAM program. On the other hand, MICAM which is an OCCAM interpreter that runs on PCs, allows the designer to validate and evaluate programs. Because of the interesting characteristics of the OCCAM language, MICAM can well be used to simulate and study other types of synchronous and asynchronous architectures. Furthermore, it could also be used for tutorial purposes at both undergraduate and postgraduate levels.



REFERENCES

- [1] FAGERSTROM J. and PATEL R.K.M
High level Simulation for Systolic Systems", Research report, CADLAB, Jr 1. 1986, Linköping University, Sweden.
- [2] GHANEMI S.
Non Numerical Parallel Algorithms for Asynchronous Parallel Computer Systems", PhD Thesis, Sept. 1987, The Computer Studies Dept.
Loughborough University of Technology, LUT, U.K.
- [3] GHANEMI S., HOUCINE M.H and DIABIN.
"Programmation Assistée par Ordinateur à Interface Graphique pour les Systèmes Systoliques", mémoire de fin d'études pour l'obtention du diplôme d'ingénieur d'état en Informatique, Sept. 1990, Institut d'Informatique, Université de Annaba, Algérie.
- [4] GHANEMI S., HOUCINE M.H and DIABIN N. "GEPASS: Guide de l'utilisateur", Institut d'informatique, Université de Annaba, Sept. 1990.
- [5] GHANEMI S., ZERARI F. and ZERARI S.
Simulation des Systèmes Systoliques: Génération de Modèles", Mémoire d'ingénieur, Institut d'informatique, Université de Annaba, Sept. 1990.
- [6] KUNG H.T.
Why Systolic Architectures", IEEE Computer, Jan. 1992.
- [7] KUNG S.Y
"VLSI Array Processors" Prentice Hall, Englewood Cliffs, New Jersey, USA.
- [8] MEGSON G.M
"Soft-systolic Simulation of unbounded Problems", Internal Report, N° 330, Computer Studies Dept., Loughborough University of Technology, U.K.
- [9] MEGSON G.M
Novel Algorithms for the Soft-Systolic Paradigm", PhD Thesis, the Computer Studies Dept., Loughborough University of technology, uk (1) Groupe de Recherche sur le Parallélisme (G.R.P)

* Groupe de Recherche sur le Parallélisme
(GRP°°Université Badji Mokhtar, Annaba