PROMPT: A Mapping Environment for Telecom Applications on "System-On-a-Chip"

Michel Barreteau,
Juliette Mattioli
Thomson-CSF LCR - A&TS Lab.
Domaine de Corbeville
91404 Orsay (France)
firstname.lastname@lcr.thomson-csf.com

Thierry Grandpierre,
Christophe Lavarenne,
Yves Sorel
INRIA Rocquencourt - BP 105
78153 Le Chesnay (France)
firstname.lastname@inria.fr

Philippe Bonnot,
Philippe Kajfasz
Thomson-CSF Communications
66, rue du Fossé Blanc - BP 156
92231 Gennevilliers (France)
firstname.lastname@tcc.thomson-csf.com

Corinne Ancourt, François Irigoin
Ecole des Mines de Paris - CRI
35 rue Saint Honoré
77305 Fontainebleau (France)
firstname.lastname@cri.ensmp.fr

Bernard Dion
SIMULOG
1 rue James Joule
78286 Guyancourt (France)
firstname.lastname@simulog.fr

ABSTRACT
Increasing of computation needs and improving of processor integration make the mapping of embedded real-time applications more and more expensive. PROMPT [1] provides a new approach which relies on the co-operation of two technologies whose main strength consists in simultaneously taking into account regular and irregular aspects of telecom applications on Systems-On-a-Chip.

Keywords
System-on-a-Chip, heterogeneous multiprocessor embedded system, constraint, mapping environment.

1. INTRODUCTION
In the next decade, "Systems-On-a-Chip" (SOC) will play a crucial role in telecom applications (UMTS, Wide Band, …). Deep sub-micron technologies are designed to give an answer to the high level of integration required by these applications. They will have to deal with all the various levels of digital processing, from Signal Processing (SP), in digital beam-forming applications for instance, to protocol stacks.

The architecture of these SOC will be fundamentally heterogeneous. It will integrate various computing engines devoted to specific functions like number crunching (filtering, FFT, …), Digital Signal Processing, data processing and/or decision and supervision. Number cruncher engines, which exploit data parallelism are optimally designed using homogeneous SIMD structures. They may be interconnected with VLIW, DSP, RISC cores or micro-controllers in order to enlarge the application spectrum. For instance, Mefisto [2] an example of advanced SOC architecture, is based on a SIMD computation unit (Marañon), a floating point unit (mAgicFPU) and a general-purpose processor (ARM).

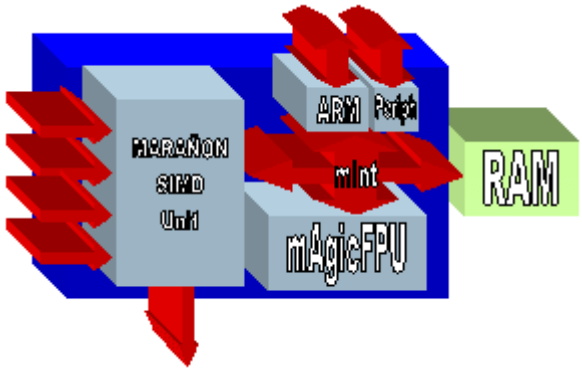

Figure 1. Global Architecture of the Mefisto SOC.

Unfortunately, programming of such SOC becomes harder and harder. Indeed, mapping and performance estimation of real-time applications on this kind of engines, closely interconnected on a single silicon die, is not trivial at all. Moreover, complexity of software development (processing and communication) is drastically increasing because each engine uses its own tools (Assembler, C/C++ compilers, optimizer, debugger, …) and furthermore because on-chip inter-engine communications are not supported by these tools.

This paper focuses on a new approach, and the environment which is based on, for helping users to map telecom applications onto SOC. It is currently validated through a prototype environment which is based on the co-operation between two complementary methodologies. One of them is optimized to handle SIMD and regular aspects of SP applications and SOC, whereas the other one takes into account irregular and MIMD aspects required by the SOC and such applications.

This article is articulated as follows. The next section gives an overview of some development tools for mapping and optimizing embedded real-time applications onto multiprocessor architectures to rapid prototyping. Section 3 describes AAA and PLC2 methodologies that compose the generic software design approach of the PROMPT project and why it makes sense to combine them. Before concluding, section 4 focuses on an example which illustrates our new global approach.

2. RELATED WORK

Several development tools that aim at mapping and optimizing applications onto multi-processors are presently available as commercial or research products. APOTRES97, CASCH, Fx, GEDAE, Ptolemy, SynDEx, TRAPPER are tools of this type. We do not intend here to provide a state-of-the-art but only give an overview of three of these tools which are close to our needs.

Ptolemy [3] (University of Berkeley) is a simulation, modeling and code generation environment for SP heterogeneous systems; it can use several models in the same simulation to take into account several aspects of an embedded system. It performs optimizations based on a very abstract model of architecture avoiding to take into account the SIMD and MIMD characteristics of the architecture.

SynDEx [4] (INRIA) is a system-level CAD for rapid prototyping and optimizing the implementation of real-time embedded applications on heterogeneous and homogeneous multiprocessor architectures. It generates dedicated (very low overhead) executives without deadlock for these architectures.

GEDAE [5] (Lockheed Martin ATL) is a commercial programming environment, whereas both previous ones are research tools. As SynDEx does, it allows rapid prototyping on heterogeneous and homogeneous multiprocessor machines; on the contrary, it is not possible to specify generic architectures not provided by the environment. It generates executives based on a real-time OS.

Each tool has its own features but none of them allows to simultaneously take into account SIMD and MIMD architectures in the optimization process, as it is necessary for the SOC we are aiming here. We choose to extend the SynDEx capabilities towards SIMD by combining it with the EPHORAT tool which is specially devoted to SIMD architectures.

3. METHODOLOGIES

Consequently, our approach is based on a co-operation between two methodologies and use of their associated tools:

- the Algorithm Architecture Adequation (AAA) [6,7] methodology for homogeneous and heterogeneous MIMD architectures, with the SynDEx software,
- the PLC2 [8,9] methodology, concurrent constraint programming model-based, optimized for homogeneous SIMD architectures, with the EPHORAT software.

After their own description, some comparisons will help to understand our motivations to couple them.

3.1 AAA

The goal of the AAA methodology is to find the best matching between an application algorithm and an MIMD architecture, while satisfying constraints. We will use further only the generic word "algorithm" in order to denote the set of all the algorithms used to specify the functionality of an application. Algorithms may be coded in any source languages. Note that "Adequation" is a French word meaning an efficient matching; it is different from the English word "adequacy" which involves a sufficient matching. AAA is based on graph models to exhibit both the potential parallelism of the algorithm and the available parallelism of the multiprocessor (Cf. bottom of Figure 2). The implementation consists in distributing and scheduling the algorithm data-flow graph on the multiprocessor hyper-graph while satisfying real-time constraints. Heuristics are used to optimize the real-time performances and resources allocation of real-time applications running on the multiprocessor. This optimized implementation process is formalized in terms of graphs transformations. The results are on the one hand a timing simulation of the behavior of the algorithm running on the architecture, and on the other hand an optimized Synchronized Distributed Executive, automatically built from a library of architecture-dependent executive primitives composing the executive kernel. There is one executive kernel for each supported processor. These primitives support boot-loading, memory allocation, inter-engine communications, sequentialization of user supplied computation functions and of intra and interprocessor communications, and finally synchronizations between sequences of computations and interprocessor communications.

3.2 PLC2

PLC2 methodology focuses on mapping of Systematic Signal Processing (SSP) applications on parallel computation units (SIMD or SPMD) using a concurrent constraints programming language. Systematic means that such application is based on patterns of computations repeated several times through loops on different data. SIMD architectures are dedicated to implement this kind of applications. PLC2 uses a global multi-model approach to describe the general mapping problem. It takes as input the specification of different models (Cf. bubbles in bottom of Figure 3) such as the target machine, the communication costs, the partitioning, the memory allocation and the scheduling models. These models are represented with mathematical variables and affine constraints. Non linear constraints (Cf. small arrows in bottom of Figure 3) such as the number of processors link the different models and generally are composed with complex and polynomial terms. The latency, resources and data-flow dependencies constraints are global constraints. While storing the different constraints, the constraints solver system builds a solution-space on a model-per-model basis. Each model solution-space is pruned when constraints are propagated from other models. Indeed the modelled symbolic constraints are propagated concurrently and the solutions satisfying the global problem are enumerated. The solution outlook depends on several criteria as memory allocation or latency. Solutions must be looked for in a resulting overall search space using a specific global search. This search relies, first on the semantics of the variables of each model and their importance w.r.t. other models and, second the goal to achieve (i.e. resource minimization under latency constraint, latency minimization under resource constraint). Each variable takes part in a global cross-model composite solving, such that only relevant information is exchanged between models. The result is a fine-grain schedule of computational blocks, their distribution onto processors, the latency and the memory allocation.

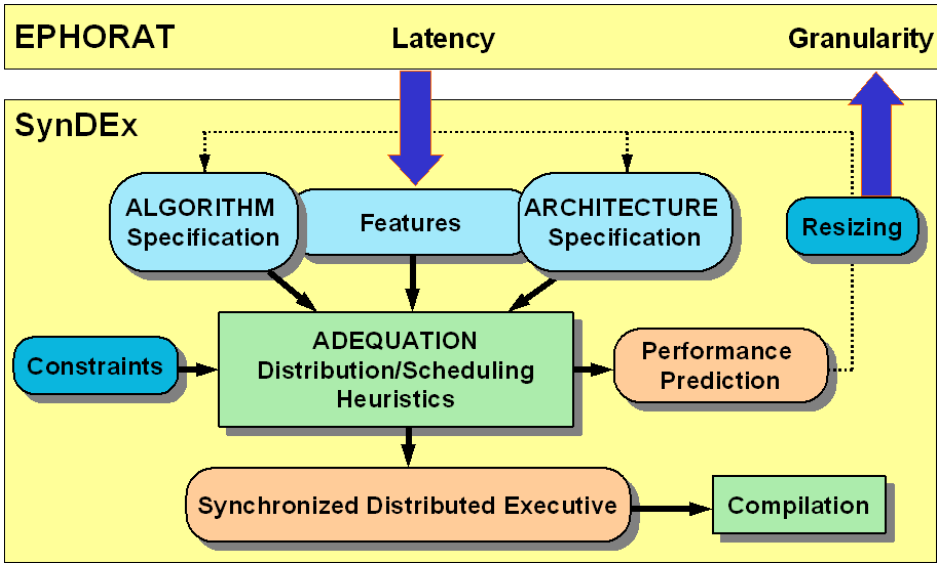

Figure 2. AAA Methodology within the PROMPT Environment.

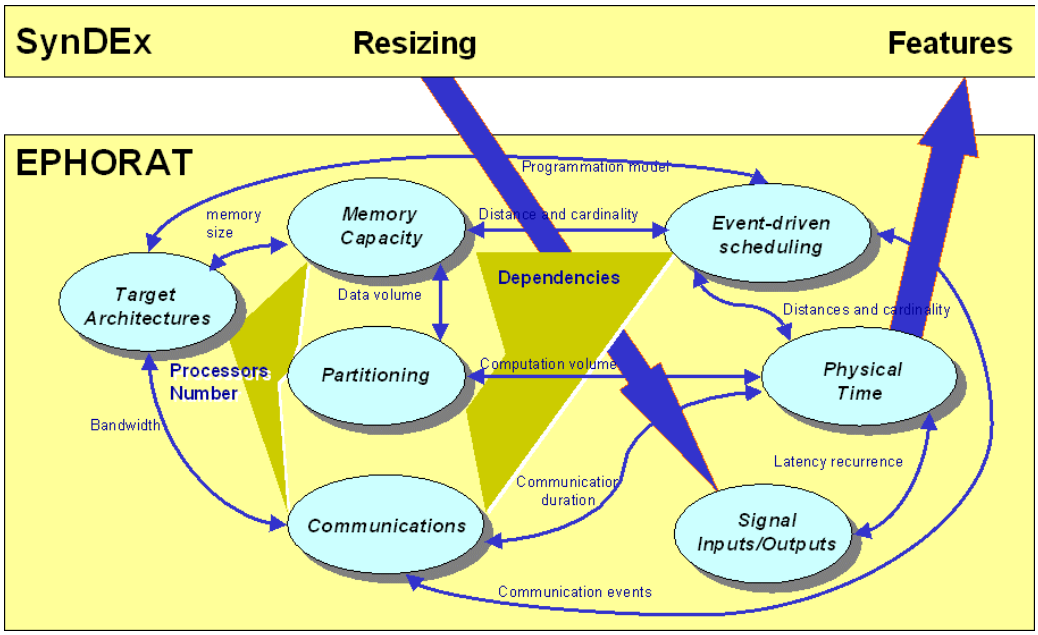

Figure 3. PLC2 Methodology within the PROMPT Environment.

3.3 Motivations for combining AAA and PLC2

In order to understand why it makes sense to combine AAA and PLC2 methodologies, we compare them according to three items: algorithm model, architecture model and features of the tools which support these methodologies.

3.3.1 Algorithm Model

AAA and PLC2 methodologies share the same kind of model to specify the algorithm: a data-flow graph based on data dependencies between computation operations. The algorithms may have regular and repetitive parts on one or several dimensions. Benefits of this common model are not only its declarative aspect (close to mathematical expressing of SP algorithms) but also the potential parallelism related to the partial execution order due to the data dependencies between operations. This potential parallelism may be directly exploited in order to implement the algorithm on the available parallel architectures. However, in PLC2 which is devoted to SSP, each operation (a vertex of the algorithm graph) is a nested loop executed without any condition, whereas in SynDEx which is devoted to complex applications with automatic control, SP and image processing that may include some SSP, an operation has a more general control

structure, including nested loop, and may be conditioned. Moreover, because PLC2 aims SIMD architectures it requires fine grain operations in order to specify data parallelism. AAA allows the user to specify the algorithm with different sizes of grain, which are usually bigger when it is not necessary to detail data parallelism.

3.3.2 Architecture Model

PLC2 implicitly considers fully-connected homogeneous architectures (SIMD or MIMD using SPMD programming model). "Homogeneous" means that processors of the architecture are similar to each other (same memory capacity, same duration of computation and communication tasks). AAA target machines are MIMD architectures where one or several processors may be a SIMD processor. The processor graph (described in extension) has to be connected but not necessarily fully. The main difference between PLC2 and AAA relies on the capability for PLC2 to support a unique sequencer controlling several computation units, whereas AAA deals with several sequencers, each of them controlling one computation unit.

3.3.3 Tool Features

About the Man-Machine Interface, EPHORAT inputs mainly are the architectural parameters (number of processors, latency, memories capacity, throughputs, …), the algorithm and a partial solution if necessary. SynDEx offers a graphical interface to specify both the algorithm and architecture graphs and the corresponding parameters (execution durations of computing and communication operations, distribution constraints).

SynDEx provides libraries of portable operations to specify the algorithm, EPHORAT provides libraries through a specification language of SP applications: Array-OL [10].

Satisfying some constraints such as real-time or resources ones is difficult not only due to the number of constraints but also to the type of constraints: global or elementary constraints. The former are complicated to manage because they all have to be respected (PLC2 models). The latter are only intricate to handle because of their number ; such constraints correspond to the operation durations used by SynDEx. PLC2 automatically manages global constraints (e.g. latency) thanks to a constraints solver.

SynDEx and EPHORAT are designed for validation, not for simulation. As a matter of fact they only provide a schedule that can be viewed as a prediction of the real-time behavior without simulating the actual computations. SynDEx performs off line distribution and scheduling of computations and communication operations onto the processors composing the architecture. EPHORAT computes a schedule of partitioned nested loops that are interleaved to maximize parallelism. Moreover, EPHORAT is not restricted to the validation of a given mapping since it is ensured by the constraints solver itself. It can enumerate multiple solutions without violating the whole constraints.

SynDEx uses optimization heuristics mainly based on critical path computed from the execution durations of the operations in order to minimize the response time of the application. EPHORAT offers several optimization criteria (e.g. memory, latency). Minimizing search space of solutions is one of the main strengths of EPHORAT.

The optimized application has to be executed on a realistic machine. Then code generation is a major issue. Whereas SynDEx generates automatically a dedicated executive with very low overhead, EPHORAT only generates placement directives.

3.3.4 Summary

Comparisons above showed two complementary methodologies. PLC2 handles fine-grain, strongly repetitive (regular) and unconditional algorithms on homogeneous SIMD (or MIMD using SPMD programming model) architectures, whereas AAA focuses on irregular, medium-grain and conditional algorithms on heterogeneous MIMD architectures.

4. AN INNOVATIVE GLOBAL APPROACH

Our new approach consists in simultaneously taking into account regular and irregular aspects of telecom algorithms as well as the SIMD and MIMD aspects required by the SOC. This approach relies on the AAA methodology with the assistance of PLC2 for the more intensive and data-parallel parts of the algorithm implemented on the SIMD part of the architecture. The way both tools communicate is shown through figures 2 and 3 and further illustrated. The benefit of this approach is that it provides a global optimization improved by local optimizations for regular parts of the algorithm. As a result, the user may obtain on the one hand a more accurate timing simulation of the behavior of the algorithm running on the architecture, and on the other hand a dedicated distributed real-time executive automatically generated.

Our innovative global approach would improve productivity through rapid prototyping using executives generation. Moreover, debugging of distributed executives is no more a costly task; based on timing simulations this approach enables evaluation and performance optimization of algorithms without any physical available architecture, and easy re-targeting of algorithms on other architectures.

Combining AAA and PLC2 methodologies actually leads to coupled their respective tools SynDEx and EPHORAT. In order to figure out the principles of this coupling we give below a realistic example of a SSP (radio telecom) application optimized with SynDEx/EPHORAT.

First the algorithm and the architecture must be specified on both tools. The algorithm (Cf. top of Figure 4) is made of computing operations *"filtPuiCor, detect, max, filt_final, Corr_*and *pond"* (each of them is based on nested loops since it is a SSP application). There is a sensor operation *"antennes"* and an actuator operation *"modem"*. The sensor produces data for *filtPuiCor* and *Corr*, which in turn produces data to *detect* and *pond* and so on, until the actuator operation. The architecture (Cf. top left of Figure 4) is the *Mefisto SOC*, made of two processors the *mAgicFPU* and the *Marañon*. The *mAgicFPU* a floating point DSP processor can execute all the operations of the algorithm whatever their data types are, but regular non floating point computing may be slower than if executed by the *Marañon*. This latter a SIMD processor (called "root" on the figure 4) can only deal with non floating point data. *Corr*, *filtPuiCor* and *filt_final* operate on non floating point data and require high computational power; then they are best suited to be executed by the SIMD unit of the SOC. All other operations operate on floating point data, so they must be executed by the mAgicFPU.

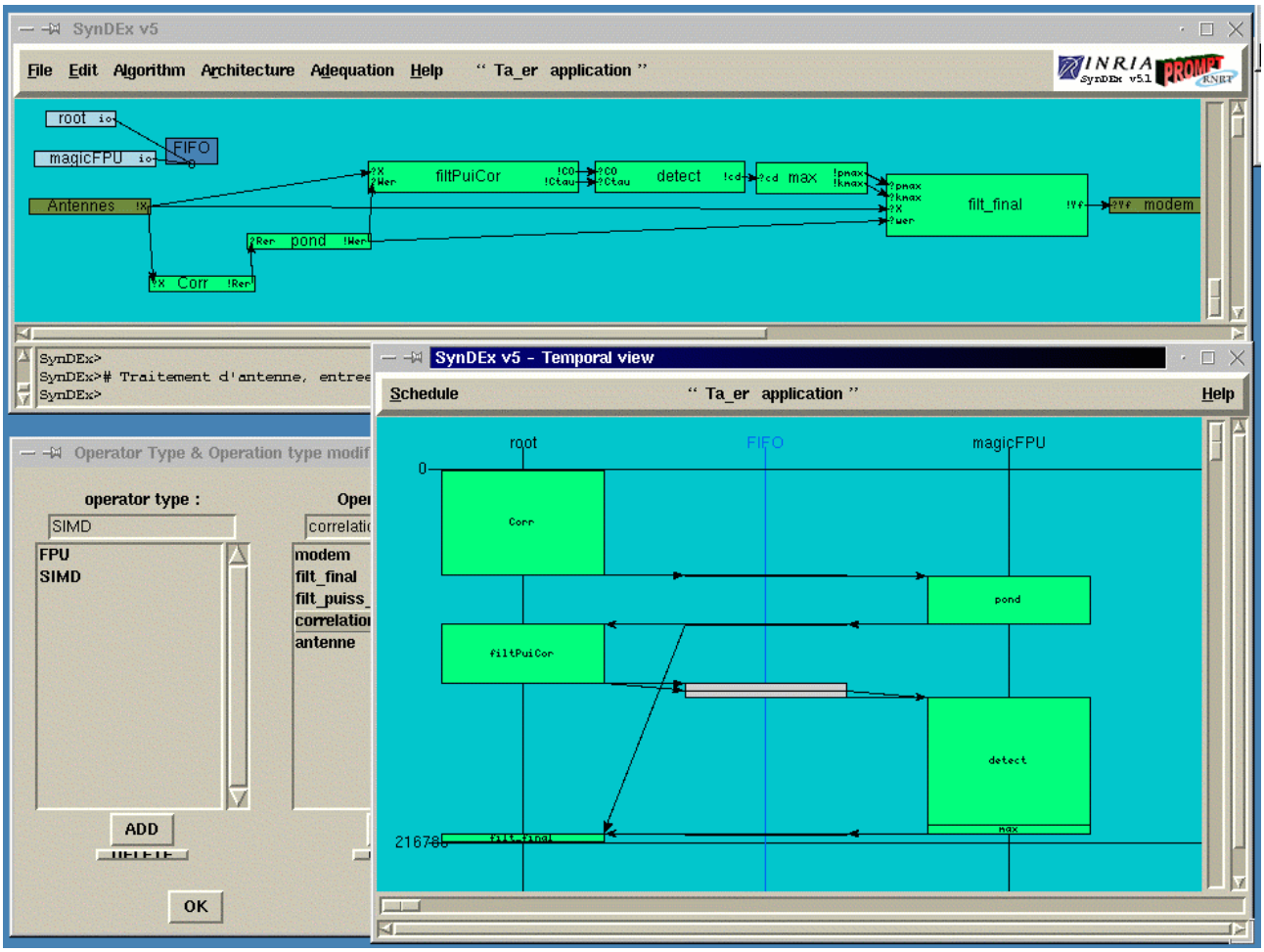

Figure 4. A basic SSP application specified with SynDEx and the resulting timing simulation.

In order to perform optimizations, SynDEx's heuristics need the execution duration of each operation of the algorithm on each processor able to execute it. The execution durations of each operation executed on the *mAgicFPU* is obtained by measuring its in real-time. EPHORAT is used to optimize and determine the accurate value of the execution duration of each operation executed on the SIMD *Marañon*; it also provides the optimized data mapping exploiting the data-parallelism of the SIMD architecture. Then SynDEx uses these results for communication and code generation.

For instance, Figure 5 shows the mapping results of three SIMD tasks (reduced to *filtPuiCor* in Figure 4) that have to be executed by the *Marañon*. From the Man-Machine Interface where the characteristics of the target architecture *Marañon* and the (radio telecom) algorithm are described (Cf. top left of Figure 5), a solution is computed by EPHORAT according to the optimization criterion (e.g. memory, latency) then printed on screen (Cf. top right of Figure 5) and in a report file. Bottom of Figure 5 illustrates the partitioning (how nested loops are split and on which processor) and the scheduling (how computation blocks of which period are interleaved) of the last solution found.

When an operation is not able to be executed by a processor its execution duration on that processor gets an infinite value. In our example *pond*, *max* and *detect* operations get an infinite value for the SIMD *Marañon* and finite values for the *mAgicFPU*. The timing simulation diagram produced by SynDEx (Cf. bottom right of Figure 4), shows that there is no parallelism in the algorithm specified as presented previously in the top of Figure 4. It points out clearly that hardware resources are poorly exploited since the FPU processor has to wait the end of *Corr* execution before starting to execute *pond*. Symmetrically the SIMD processor has to wait the end of *pond*, before starting to execute *filtPuiCor*. Moreover, the communications between the SIMD and the FPU induce execution delays and, on an implementation point of view, these communications may require large data buffer (to store data in order to transmit them between processors) which may not be available on the SIMD.

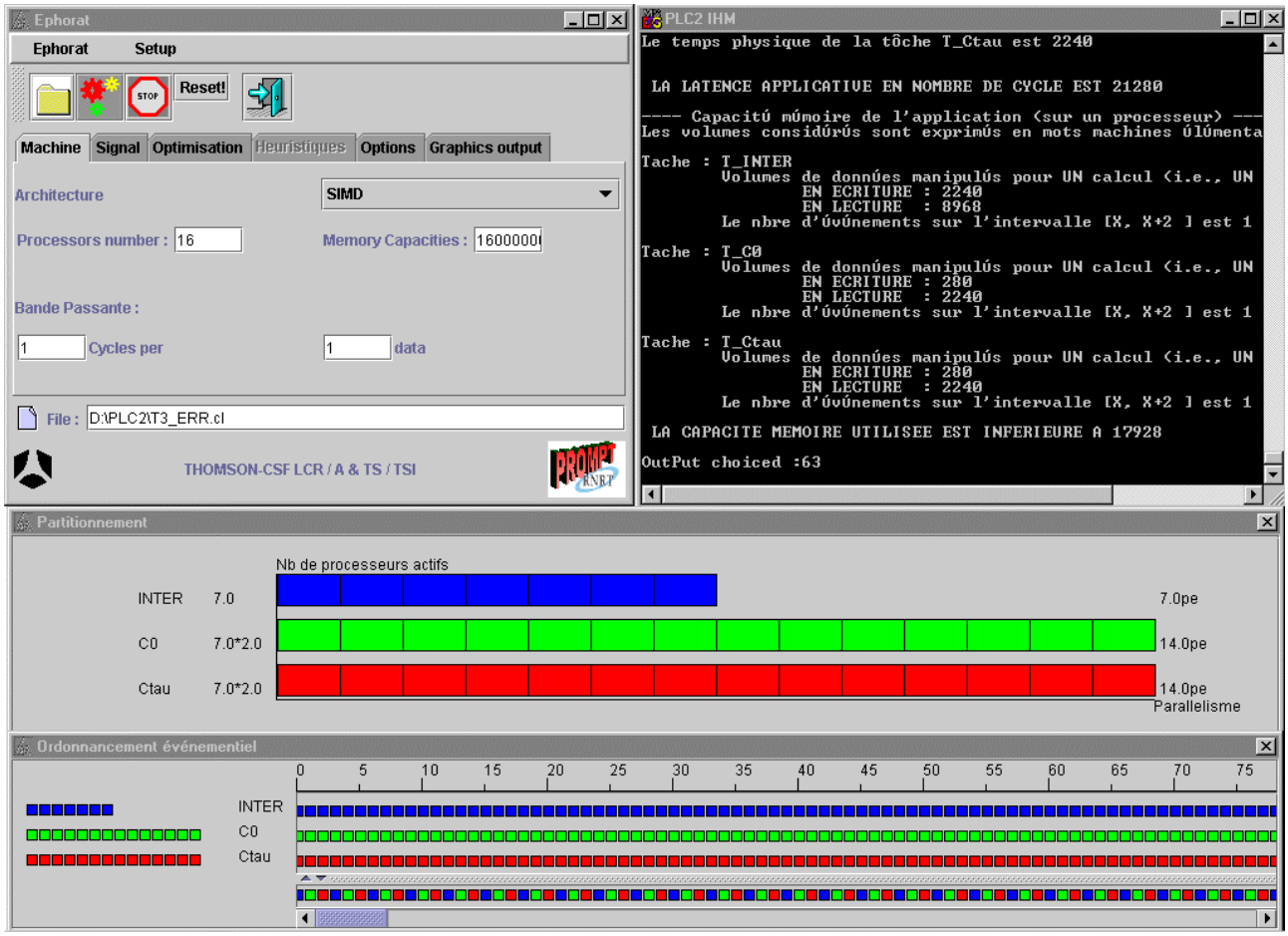

Figure 5. EPHORAT partitioning and scheduling results.

SynDEx/EPHORAT may help the user to solve both problems by modifying the original algorithm specification. Since all the operations are each based on nested loops, it is possible to split up them into two smaller operations: *Corr* become *Corr1* and *Corr2*, *pond* become *pond1* and *pond2* and so on (Cf. top of Figure 6). This transformation of the algorithm specification introduces potential parallelism which will be exploited in order to optimize the execution from a response time and load balancing point of views. The new timing simulation made with SynDEx/EPHORAT shows (Cf. right of Figure 6) that the operations now overlap their executions. The overall execution (response time) of the algorithm is reduced, and the data buffer required to store data in order to transmit them between the processors, is smaller.

Thanks to SynDEx/EPHORAT it is easy to try several modifications of this type in order to determine the correct splitting of the operations. Big arrows of Figure 2 shows that EPHORAT provides the latency (optimized execution duration of SIMD blocks of operations) to SynDEx. The latter exploits it and sends its results to EPHORAT back to resize the grain until a correct splitting is found (see big arrows on Figure 3). Currently both tools can only interact thanks to the exchange of data by hand.

In this example only one split is necessary since more splits would lead to increase the communication cost too much. Moreover, since the FPU processor is able to execute all the operations of the algorithm (but sometimes slower than the SIMD processor), another solution to solve the problems mentioned before could be to split some loops assigned to the SIMD in order to execute some of them on the FPU according to the aimed response time.

5. CONCLUSION

PROMPT, a development environment to map telecom applications on SOC is based on the co-operation between two complementary methodologies AAA and PLC2. It consists in simultaneously taking into account regular and irregular aspects of SP applications as well as the SIMD and MIMD aspects required by the SOC. The first SOC using this approach is Mefisto.

The kind of model used to specify the algorithm is shared by their associated tools SynDEx and EPHORAT. Hence they can interact and yield an integrated development environment for SOC from specification to optimized code generation.

PROMPT provides the ability of sizing the grain. So results about communications in terms of response time and memory allocation can be studied.

Improving algorithm specifications (nested loops splitting and distribution) enables to better benefit by SOC (SIMD-MIMD tradeoff). This process is currently done by hand. Its automation is planned into the optimization heuristics.

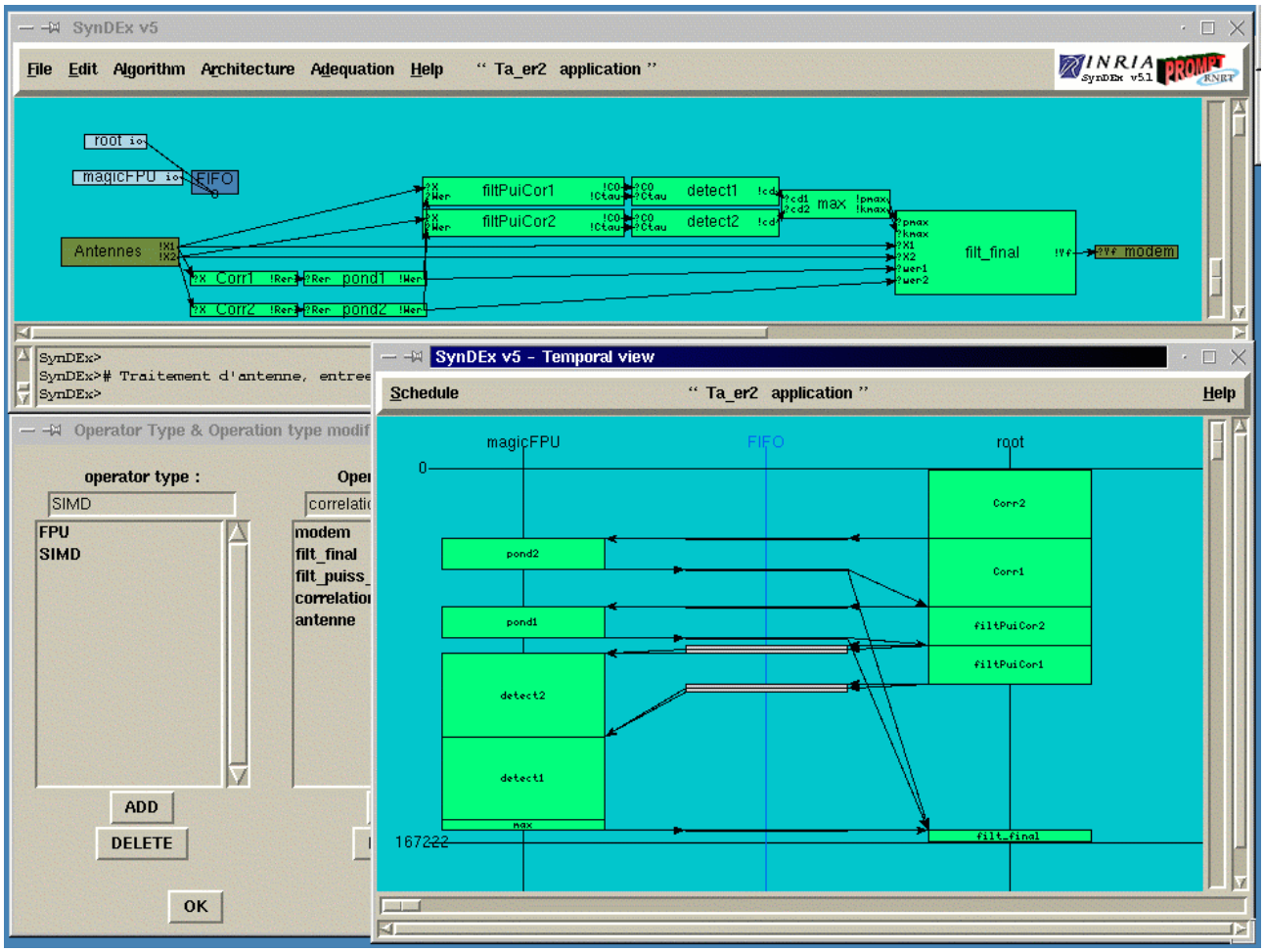

Figure 6. Algorithm specification modification for implementation improvement

6. REFERENCES

[1] "Placement Rapide Optimisé sur Machines Parallèles pour applications Télécoms", Project supported by the French Ministries of Industry and Research through the RNRT'98 (Réseau National de Recherche en Télécommunications) program, http://www.tcc.thomson-csf.com/prompt

[2] D. Maufroid, P. S. Paolucci, P. Kajfasz, A. Bertini, "mAgic FPU: VLIW Floating Point Engines for System-On-Chip Applications". *Proc. EMMSEC'99*, Stockholm, Sweden, (June, 1999).

[3] http://ptolemy.eecs.berkeley.edu/

[4] http://www-rocq.inria.fr/syndex/

[5] http://www.gedae.com

[6] T. Grandpierre, C. Lavarenne, Y. Sorel, "Optimized Rapid Prototyping for Real Time Embedded Heterogeneous Multi-Processors", *CODES'99 : 7th International Workshop on Hardware / Software Co-Design*, Rome, Italia (May, 1999).

[7] Y. Sorel, "Massively parallel systems with real-time constraints, the Algorithm Architecture Adequation Methodology", *Proc. Massively Parallel Computing Systems*, Italy (May, 1994).

[8] C. Ancourt, D. Barthou, C. Guettier, F. Irigoin, B. Jeannet, J. Jourdan, J. Mattioli, "Automatic Data Mapping of Signal Processing Applications", *IEEE International Conference on Application Specific Systems, Architectures and Processors*, p. 350-362, Zurich, Switzerland (July, 1997).

[9] C. Guettier, "Optimisation globale du placement d'applications de traitement du signal sur architectures parallèles utilisant la programmation logique avec contraintes", PhD Thesis, Ecole des Mines de Paris, (December, 1997).

[10] A. Demeure, A. Lafage, E. Boutillon, D. Rozzonelli, J-C. Dufourd, J-L. Marro, "Array-OL : proposition d'un formalisme Tableau pour le Traitement de Signal multi-dimensionnel", *GRETSI'95*, Juan-les-Pins, (France).