

Model Driven Scheduling Framework for Multiprocessor SoC Design

Ashish Meena and Pierre Boulet
e-mail: Ashish.Meena@lifl.fr, Pierre.Boulet@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille
Cit e scientifique,59 655 Villeneuve d'Ascq, Cedex, France

Abstract. The evolution of technologies is enabling the integration of complex platforms in a single chip, called a System-on-Chip (SoC). Modern SoCs may include several CPU subsystems to execute software and sophisticated interconnect in addition to specific hardware subsystems. Designing such mixed hardware and software systems requires new methodologies and tools or to enhance old tools. These design tools must be able to satisfy many relative trade-offs (real-time, performance, low power consumption, time to market, re-usability, cost, area, etc).

It is recognized that the decisions taken for *scheduling and mapping* at a high level of abstraction have a major impact on the global design flow. They can help in satisfying different trade-offs before proceeding to lower level refinements.

To provide good potential to scheduling and mapping decisions we propose in this paper a *static scheduling framework for MpSoC design*. We will show why it is necessary to and how to integrate different scheduling techniques in such a framework in order to compare and to combine them. This framework is integrated in a model driven approach in order to keep it open and extensible.

1 Introduction

As compared to previous small embedded controllers designing, the designing complexity of today's embedded system (SoC, MpSoC, etc) has grown exponentially. This complexity rise has been seen due to complex integration of multiple resources (DSP, GPP, FPGA, etc) on a single die, the miniaturization of electronic and mechanical devices, the changes in the approaches to design as determined by the cognitive complexity of the task, the massive use of test automation tools, and the pervasive use of embedded system in all kinds of applications. Real-time systems, beside high performance computing, are about strict timing constraints (deadline, periodicity), application-specific design, and high reliability (fault tolerance). The current trend of using multiple processor (SoC, MpSoC etc) has been promoted to satisfying the real-time systems needs. This trend has made real-time systems a special case of chip level high performance distributed computing. Using multiple processors is an attractive solution because of its price-performance ratio, locality constrains (data processing must

take place close to sensors or actuators), reliability (replicating of computing resources provides fault-tolerance), high throughput (parallel task execution) and high schedulability (many scheduling possibilities).

Scheduling and mapping optimization is at the core of the design process of such systems. It is recognized that scheduling and mapping at the highest abstraction level has the most important optimization potential compared to low-level optimizations. Furthermore, as time-to-market constraints become tighter and tighter, one should take these decisions as early as possible in the design flow in order to start software and hardware development concurrently and the earliest. These decisions should be definitive as a late modification would induce a too long re-engineering process that would cause to miss time-to-market constraints.

These scheduling and mapping optimizations have to take into account many factors such as real-time constraints, power consumption, various costs (area, fabrication, verification, etc), re-usability, adaptability, etc.

As stated scheduling and mapping optimizations are one of the key factors of the whole co-designing process. Keeping all the above mentioned problems in mind we propose here a framework for scheduling and mapping optimization. This framework will be used and integrated within our project Gaspard2 [10], a model driven co-design environment for computation intensive embedded systems. As we focus on intensive applications, we limit ourselves to static scheduling. Dynamic scheduling is another class of problems that we do not handle for the moment.

In the past many attempts have been made to devise scheduling and mapping frameworks or tools for real-time embedded system. As stated about the current and future trends of embedded systems, among them many of the tools were not designed to cope with the rising complexities. For instance, tools such as [11, 15, 1, 16, 13, 12] etc, implement a collection of common scheduling algorithms and protocols for communication and resource access. Among them tools like [1, 16, 13] provides a GUI with their own modeling language and one has used UML base modeling [15]. Here some have focused on desired behavior of the system rather than implementation choices, such as which scheduling policy to use. Further, most of the existing frameworks are not considering multi-criteria constraints, flexibility and scalability to apply different newly or existing scheduling and mapping techniques [11, 17, 7, 14, 12] etc.

In contrast to tools or framework listed above, our framework in addition, is targeted to certain required futuristic features, which makes it unique to the best of our knowledge. It is applying scheduling and mapping algorithms on a global task graph with three approaches first simultaneously, second iteratively, third with user concerns. It evaluates, compare and merges their relative results while respecting different constraints (multi-objective).

It will provide the flexibility of plug-ins and interoperability with legacy tools and with new methods/formats as and when they appear for scheduling and mapping related optimization in MpSoC research domain.

The rest of this paper is structured as follows: in section 2, we discuss the need for a model driven scheduling and mapping framework; in section 3, we present the common extensible graph structure that will allow cooperation between several scheduling and mapping heuristics; then in section 4, we will give heuristic integration examples; we will finally conclude in section 5.

2 Why a Framework?

2.1 Variety of Problems and Solutions

Some proposed scheduling and mapping algorithms are mono-objective, most of them minimize latency; some others take into account several parameters such as computation time, communication time and power consumption. Others give answer to only part of the problem such as mappability for core selection. Another class of algorithms targets some restricted application domain, such as loop nests, for which they are more efficient than generic ones. In order to benefit from the strength of all these heuristics, we need to integrate them in a single framework.

To be able to combine them and to compare their solutions, we need a common application and hardware platform description, based on which we can feed the optimization algorithms their inputs and integrate their results. We propose such a graph based description in section 3.

Furthermore, in order to use different integrated scheduling and mapping algorithms, the problem of algorithm selection, for which part of the application (data or task parallelism...) and for what objective (fast execution, low power...) is a highly critical issue. We propose different approaches to this problem in section 4, the goal being to give to the designer of a complex heterogeneous SoC some help in making one of the most crucial decisions of its design that no current tool provides.

2.2 Model Driven Engineering

What are the benefits of using a model driven approach for building this kind of framework? One is purely technical, the other is user friendliness. Indeed, model driven engineering (MDE) is really about interoperability and abstraction.

Interoperability. MDE is becoming a more and more used approach to develop software. Thus many tools are becoming available to handle models, metamodels and model transformations. These tools often rely on standards that allow interoperability between them. Using model transformation tools, it is relatively easy to extract part of a common model to generate the input file format of a given scheduling tool. Many software libraries exist to manipulate models; we leverage all these to build our framework.

Abstraction. The other strong point of MDE is abstraction. Indeed, models are built to be more readily understandable to humans than programs. The user can design its application, hardware and their association in a piecewise way,

using a top-down or bottom-up approach or a combination thereof, as he wishes and reuse part of previous designs easily.

Obviously, to be able to integrate different scheduling and mapping heuristics, one has to share common models for the application, the hardware architecture and their association. These models have to be extensible to allow for adaptation of future techniques. MDE helps in this aspect by allowing very easily to extend models. We describe in the next section what should these models be like.

3 Graph Based Input Models

3.1 Hierarchical Graphs

Most scheduling and mapping heuristics work on graph based description of the application and the architecture. In order to be able to handle large and complex systems, we should have a hierarchical description of, at least, the application. It is also important to have a hierarchical description of the hardware architecture in order to accommodate different granularity views. One can thus start with a coarse grain optimization and then refine it by going deeper in the hierarchy to reveal more details.

Another required structure of these models is some kind of compact way of representing repetition, such as loops in the application or SIMD units in the hardware. Indeed such a compact representation helps a lot in generating efficient code for these repetitive structures. In this way, we are able to benefit from both task parallelism and data parallelism. Depending on the optimization algorithm used to map and schedule these repetitive structures they may be seen as an indivisible entity or a structured (regular) repetition or even an unstructured (irregular) set of tasks.

We have proposed such repetitive, hierarchical graph models to describe computation intensive embedded applications and SoC hardware platforms in [8, 9, 3]. These models are based on UML2 profiles and are incorporated in the Gaspard2 co-modeling tool [10].

3.2 Characterization

In order to completely define the application and hardware architecture parameters, the graph models need to be characterized. These characteristics are attributes of the nodes and edges of the graphs.

Some of these characteristics are application specific, as real-time constraints of needed computing power. Some are architecture dependent, as instruction set of processors, operating frequency range, power consumption, memory size, communication bandwidth, etc. And some characterize the association of application components to architecture components such as execution time estimations (worst-case or average)¹ or mapping constraints. These are represented as links

¹ Finding good approximations is beyond the scope of this paper and is a research problem in itself.

between application graph elements (nodes or edges) and architecture graph elements.

3.3 Handling Optimization Results

As the optimization results have to be combined, we need a repository to store them. The model driven approach helps us in this prospect as a model is in itself a repository of concepts and links between concepts. We keep a set of models representing partial optimization results. These results are represented as mapping constraints, for the mapping, and as artificial dependencies (links between nodes of the application) for the scheduling. Some specific constructs are used to keep a compact representation of repetitive (or cyclic) schedules and mappings of data-parallel application components onto repetitive hardware (grids of processors, SIMD units or FPGAs).

4 Integration of Techniques

4.1 Framework Usage Schemes

Our framework is environment-sensitive and configurable for applying and selecting scheduling and mapping algorithms. It means, after analyzing application and platform all together the user can follow either of approaches (A),(B) or (C) below.

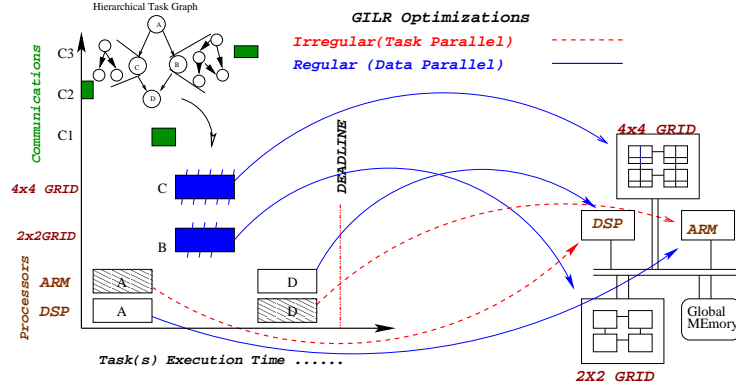


Fig. 1. Simultaneously scheduling and mapping approach (GILR)

(A) Combine different scheduling and mapping algorithms on different hierarchical (see figure 1) nodes of the global application model. Here the decision drawn for selecting specific scheduling and mapping algorithms is based on

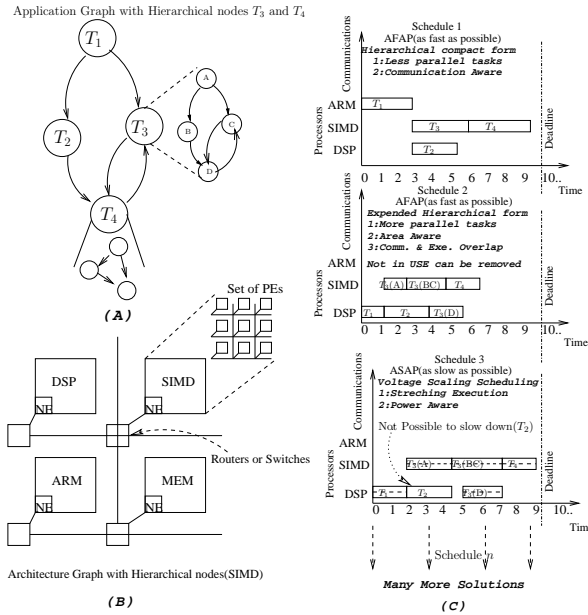


Fig. 2. Iterative scheduling and mapping approach

past stored simulation results or on the structure of the application and architecture models or even on results provided by some existing research work. We have already presented this approach as the GILR (Globally Irregular and Locally Regular) heuristic [4]. Here we have shown the combined use of a list heuristic on a global task graph, and if we find loops on any node, we applied data parallel scheduling and mapping algorithms for loops and map them on the available repetitive hardware components.

- (B) *Compare different scheduling and mapping algorithms* on the same hierarchical nodes of the application model. Consider text written in figure 2 as an explanation of techniques (Area, Communication & Power Aware etc). Here we are not sure of the quality of results, and we do not have related information in database. Here we spend more time but comparison supports us to find best scheduling and mapping algorithm. The results of scheduling and mapping will be saved in a repository for future considerations.
- (C) *User specific*, In this approach the user is fully allowed to take all decisions to select among different scheduling and mapping algorithms, iteratively or simultaneously.

Our repository based approach will allow us to choose the most adapted algorithm for each level of the hierarchy in function of the application structure, available hardware, available characteristics or even optimization criteria.

As a global solution we are expecting to help satisfying multi-criteria problems. This framework will also provide built-in decision capabilities to select one or more scheduling and mapping techniques at a time.

4.2 Implementation

The framework is implemented as a set of common metamodels and APIs. The metamodels define the graph based models described in section 3. The APIs facilitate the manipulation of the models and the use of existing scheduling and mapping algorithms and the implementation of user specific required scheduling and mapping optimizations. The framework will be flexible and extensible enough to implement newly available scheduling and mapping algorithms in the research domain of MpSoC.

Frameworks like Simgrid [6] and GridSim [5] exist in the field of distributed GRID computing. Such frameworks shares some common basic aspects for scheduling and mapping compared to on-chip distributed computing (MpSoC). Scheduling and mapping algorithms such as list scheduling, critical path scheduling, genetic algorithms, communication scheduling, etc, are common. Where as comparing the architecture platform, MpSoC typically consist of heterogeneous processors among them some nodes can be homogeneous (SIMD) array of processors. Communication networks (NoC) [2] consist of Switches and Routers. But in this communication platform nodes of SoCs are physically close to each other, have high link reliability and low latency.

5 Conclusions

We have sketched² in this paper a scheduling and mapping framework for multiprocessor system-on-chip design. This framework aims at integrating various scheduling and mapping heuristics handling part of the optimization problem, either part of the objectives or part of the application domain. In this quest, a recurring issue has been how to apply multiple optimization concepts to the specific problem at hand. Moreover, optimization theory often neglects that aspect which is crucial for the success, namely the models. Therefore, the first step, before any algorithmic properties can be considered, is to establish a model capable of expressing the problem accurately enough.

We have presented common graph based models which are at the heart of the framework. This framework is based on model driven engineering, allowing it to be user friendly and expandable. Several usage schemes of the framework have been proposed.

More precise experiments on the integration of different specific heuristics is underway. In the end, for real-time system design more cost-effective solutions can be found and developed. Furthermore, the design process will be shorten which results in reduced development cost.

² More details will be available in a research report.

References

1. T. Amnell, E. Fersman, L. Mokrushin, P. Petterson, and W. Yi. TIMES: a tool for schedulability analysis and code generation of real-time systems. In *1st International Workshop on Formal Modeling and Analysis of Timed Systems, FORMATS 2003*, Marseille, France, September 6-7 2003.
2. L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, 2002.
3. P. Boulet, J.-L. Dekeyser, C. Dumoulin, and P. Marquet. MDA for System-on-Chip design, intensive signal processing experiment. In *FDL'03*, Fankfurt, Germany, Sept. 2003.
4. P. Boulet and A. Meena. The case for globally irregular locally regular algorithm architecture adequation. In *Journées Francophones sur l'Adéquation Algorithmique Architecture (JFAAA'05)*, Dijon, France, Jan. 2005.
5. R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. In *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, pages Volume 14,13–15., Wiley Press, Nov.-Dec 2002.
6. H. Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, page 430, Washington, DC, USA, 2001. IEEE Computer Society.
7. J. Chin and M. Nourani. SoC test scheduling with power-time tradeoff and hot spot avoidance. In *DATE*, pages 710–711, 2004.
8. A. Cuccuru, P. Marquet, and J.-L. Dekeyser. Uml 2 as an adl : Hierarchical hardware modeling. In *WADL'04*, Toulouse, France, august 2004.
9. C. Dumoulin, P. Boulet, J.-L. Dekeyser, and P. Marquet. UML 2.0 structure diagram for intensive signal processing application specification. Research Report RR-4766, INRIA, Mar. 2003.
10. Gaspard2: Graphical array specification for parallel and distributed computing. <http://www.lifl.fr/west/gaspard/>.
11. A. J. Framework for task scheduling in heterogeneous distributed computing using genetic algorithms. In *15th Artificial Intelligence and Cognitive Science Conference*, Castlebar, Mayo, Ireland., 2004.
12. J. Jonsson and J. Vasell. Evaluation and comparison of task allocation and scheduling methods for distributed real-time systems. In *ICECCS*, pages 226–229, 1996.
13. J. W. S. Liu, J.-L. Redondo, Z. Deng, T.-S. Tia, R. Bettati, A. Silberman, M. Storch, R. Ha, and W.-K. Shih. PERTS: A prototyping environment for real-time systems. Technical Report UIUCDCS-R-93-1802, University of Illinois at Urbana-Champaign, 1993.
14. J. Luo and N. K. Jha. Battery-aware static scheduling for distributed real-time embedded systems. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 444–449, New York, NY, USA, 2001. ACM Press.
15. J. L. M. Pasaje, M. G. Harbour, and J. M. Drake. MAST real-time view: A graphic UML tool for modeling object-oriented real-time systems, March 2001.
16. J. A. Stankovic, R. Zhu, R. Poornalingam, C. Lu, Z. Yu, and M. Humphrey. VEST: An aspect-based composition tool for real-time systems. In *The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canada, May 2003.
17. C.-P. Su and C.-W. Wu. A graph-based approach to power-constrained SOC test scheduling. *J. Electron. Test*, 20(1):45–60, 2004.