MULTILEVEL MPSOC SIMULATION USING AN MDE APPROACH

Rabie Ben Atitallah Éric Piel Smail Niar Philippe Marquet Jean-Luc Dekeyser
INRIA FUTURS – LIFL – University of Lille, France
{benatita,piel,niar,marquet,dekeyser}@lifl.fr

ABSTRACT

In this paper, we first present an efficient Multi-Processor Systems-on-Chip design methodology based on Model-Driven Engineering. Later, a *deployment* profile is introduced to allow IP reuse and to carry multilevel implementation details. With this methodology, simulations at different levels are automatically generated, reducing the cost of targeting several levels. A compilation chain has been developed to transform the high abstraction level into both CABA and PVT simulation levels. The effectiveness of the methodology is illustrated by the development of an H.263 encoder.

I. INTRODUCTION

Multi-Processor System-on-Chip (MPSoC) architectures are becoming an incontrovertible solution for the embedded systems designed for applications that require intensive parallel computations. With new sub-micron technologies, more and more transistors will be integrated on a single chip. Such huge transistor budgets stress designers' capacity to design and verify the resulting complex chips. Thus the gap between chip complexity and the productivity of the logic design is getting wider and wider. Furthermore, MPSoC design is facing several additional challenges. The most important ones are the definition of a reliable Design Space Exploration (DSE) strategy and the implementation of this strategy into an efficient design methodology.

In this paper, we focus on the use of an Model-Driven Engineering (MDE) approach to reduce the complexity of MPSoC design. Transformation from model to model is considered the main concept of the MDE. We propose to use this concept to compile high level MPSoC models into simulations at distinct accuracy levels. The main part of the compilation chain can be reused, thus reducing the cost of targeting several simulation levels. As a second contribution, we introduce a *deployment* profile which permits to link the MPSoC model with implementation details for several target platforms. Furthermore, we propose a DSE based on SystemC simulation levels. The objective is to refine the architectural space from higher to lower level until converging to the adequate solution. To achieve this aim, we focus on the two SystemC simulation levels: Cycle Accurate Bit Accurate (CABA) and Timed Programmer View (PVT). At each level, the exploration is based on developed performance and power estimation tools.

The rest of this paper is organized as follows. An overview of related work on existing MPSoC design for DSE is provided in Section II. Section III introduces the MDE approach and then details our methodology to apply it to SoC compilation. An overview of deployment phase and the targeted SystemC simulation levels are given in Section IV. Section V presents the experimental results obtained when applying the proposed framework for multilevel MPSoC simulations of an H.263 encoder.

II. RELATED WORK

In recent years, a lot of researches on DSE for MPSoC architectures have been conducted. As a result of these researches, several academic and commercial exploration environments are proposed, such as MILAN [5], GRACE++ [4], Artemis [6] and Metropolis [1]. They distinguish from each other on several aspects. One of them is the application model description, for instance Khan Process Networks (KPN) are used by Artemis, while Synchronous Data Flow (SDF) is used by MILAN. Applications in our framework are described with the ARRAY-OL model of computation [2]. This data flow language expresses both the data parallelism and the task parallelism. An other varying aspect is the architecture model description, it goes from a functional level to a Register Transfer Level (RTL). Like MILAN and GRACE++, our framework gives the possibility of multilevel architecture description.

Our framework shares more common points with GRACE++, both rely on SystemC to obtain a homogeneous co-simulation (software and hardware). Compared to traditional heterogeneous co-simulation tools, SystemC (especially at the TLM level) accelerates simulation, as the abstraction level has been elevated from RTL to transaction level. Moreover, GRACE++ and our proposed framework target timed TLM simulation for contention monitoring in the interconnection network. Nonetheless, we obtain more interesting simulation speedup compared to the CABA level. For a reliable DSE, we define a

set of tools for performance and power estimation at each level. Our environment Gaspard [9], thanks to the MDE approach, provides great flexibility on the compilation chain. Thus, designers can couple additional tools, or target different platforms. For instance we have also carried out the generation of VHDL implementation on FPGA and of synchronous language code from the same chain.

III. MODEL-DRIVEN ENGINEERING APPROACH

Our work is based on *Model-Driven Engineering* [7] (MDE). This methodology is centered around two concepts: model and transformation. Data and their structures are represented in models, while the computation is done by transformations. Models contain information structured according to the metamodel they conform to. In our framework, models are used to represent the system (application, architecture, and allocation). Transformations are employed to move from an abstract model to a more detailed model. The set of transformations forms the compilation chain. In our case, this chain converts the platform-independent MPSoC model into a SystemC simulation code.

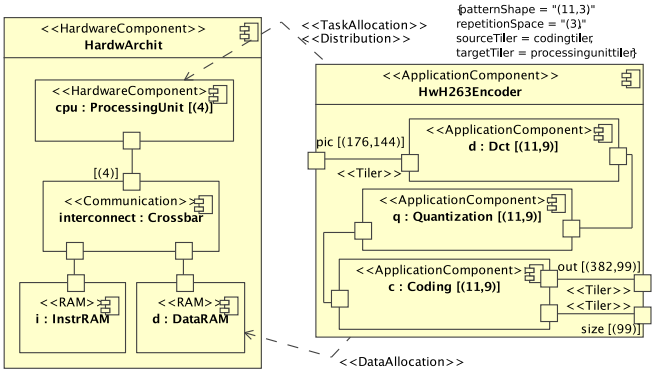

Figure 1: Global view of the H.263 application on a 4-processors MPSoC.

A. MPSoC modeling

At a high abstraction level, we use the MARTE profile standard [8] to represent an MPSoC. The representation separates the application, the hardware architecture and the corresponding allocation. Allocation allows to map tasks onto processing units and to map data arrays onto memory banks. Compared to Domain-Specific Languages (DSL), the MARTE profile uses a graphical representation, exhibiting the same advantages as UML. It is possible to separate the system into several aspects (several views at different levels of detail, or showing different domains...), thus the representation gains clarity. A strong advantage of using the MARTE profile for MPSoC modeling is the particular concept of *factorization*, both for hardware architecture and application. With the semantic introduced by the ARRAY-OL model of computation, factorization provides a mechanism that expresses the parallelism of the system in a compact way. As shown on Fig. 1, the multiplicity syntax specifies repetitions in the hardware architecture (e.g., `ProcessingUnits[(4)]`) as well as in the application (e.g., `Dct[(11,9)]`). In the same way, *allocation* syntax expresses the distribution of tasks over the processing units.

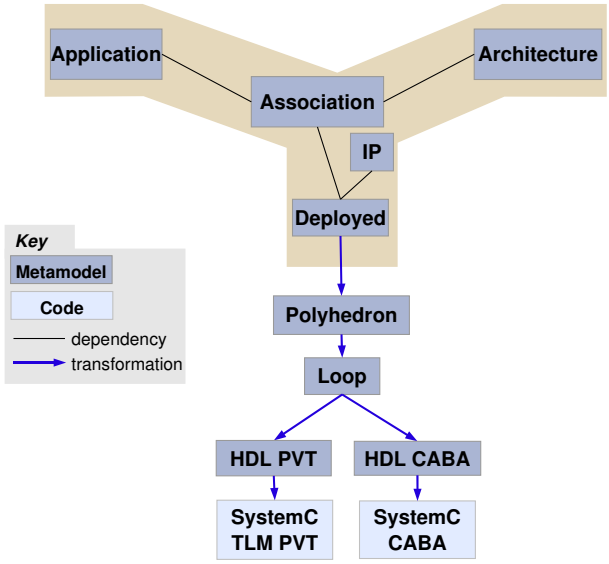

Figure 2: The part of our compilation chain for SystemC generation.

B. MPSoC model transformation

We propose to use MDE transformations in order to implement the MPSoC compilation flow. Transformations are probably the most interesting and specific aspects of MDE. In particular, transformation inputs and outputs are *formally* described by metamodels. Two chained transformations, one having its output being the input of the other one, are linked by a model conforming to a specific metamodel. This provides strongly documented "synchronization points" in the compilation flow. Consequently, each transformation is independent from the others (they only depend on the pre-defined metamodel). By following a methodology which separates the compilation into a sequence of small transformations, passing through several intermediate representations, it is possible to maximize the reuse of transformations while compiling toward different simulation levels. An overview of our compilation chain is available in Fig. 2. The models in the Y shape compose the high level MPSoC model. It is first transformed into the *polyhedron* model, where the application is reshaped and split on the different processing units. This derived model

is then transformed into the *loop* model, where the application is represented according to a traditional nested loop schema. While those transformations are shared, the later transformations are specific to the simulation level targeted. They generate simulation code with details corresponding to the SystemC simulation level selected by the user.

In SoC design the hardware is as versatile as the software. Likewise, the employed technologies evolve very quickly between the development of two products. The compilation flow must adapt easily to those evolutions. Being able to write transformations with a declarative language increases their maintainability and simplifies their modification. The declarative approach specifies that each transformation is separated into a set of rules. Each rule is associated to an explicit set of input and output patterns. Here, the engine we used to execute the transformations is a framework which permits to have a declarative structure of the transformations using Java.

IV. MULTILEVEL SIMULATION

A. Deployment

To transform the high abstraction level models into simulation code, very detailed deployment information must be provided. In particular, each elementary component must be linked to an existing code. For this purpose, a *deployment* profile is introduced. A key point in our methodology is to facilitate Intellectual Property block (IP) reuse. Therefore great care was taken to allow usage of IP libraries (models which contain a set of pre-defined IPs) and to keep the MPSoC model independent from the compilation target. In this profile, we introduce the concept of *AbstractImplementation* which expresses one hardware or software functionality, independently of the compilation target. It contains one or several *Implementations*, each one being used to define a specific implementation at a given simulation level and programming language. Fig. 3 shows an example of an *AbstractImplementation* of a MIPS processor which contains two *Implementations* at the CABA and PVT levels written in SystemC.

Using the *ImplementedBy* dependency, the designer can select the adequate IP for each hardware and software component. The *Implementation* which fits best the target will be used to reify the component during the compilation phase. This automatic selection allows to generate the exact same SoC model at different simulation levels. For automatic code generation, we use the concept of *CodeFile* to specify the code and compilation options required. *Specializations* and *Characteristics* can be added by the designer in order to pass information respectively to the generated code (e.g., to specify the port interfaces of a SystemC component) and to the transformation (e.g., timing data, implementation details...). Moreover, in order to generate simulation code with precise power estimation, the deployment model can be enriched with *PowerModels* which associate a power consumption model to a hardware component.

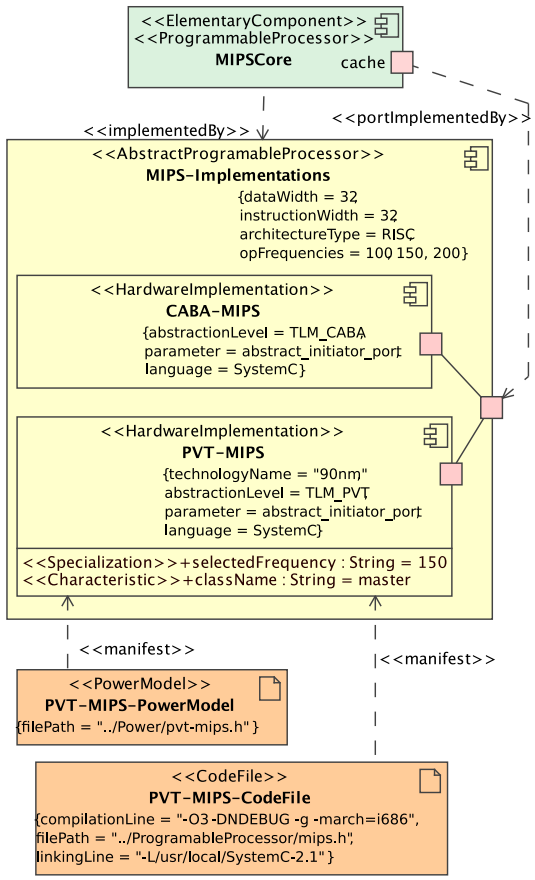

Figure 3: Deployment example of a MIPS processor.

Using this methodology, we were able to target the PVT and CABA levels in SystemC from the same design. Based on SystemC version 2.1 and the TLM library, hardware components (processors, caches, interconnection networks...) were specified. Similarly, software components (FFT, FIR, DCT...) were specified. Several implementations can also be provided for each software component, this allows to target functional simulation levels or to generate a hardware accelerator.

B. Simulation levels

The main objectives of the PVT level are fast verification of system functionalities and monitoring of the contentions in the interconnection network. Complementary to this level, the CABA level is used to

accurately estimate the execution time and power consumption. At the PVT level, details related to the computation and communication resources are omitted. The software application is executed by an instruction-accurate Instruction Set Simulator. Transactions are performed through channels instead of signals used at the CABA level. At this second level, hardware components are implemented at the cycle accurate level for both processing and communication parts. Communication protocol and arbitration strategy are specified as well.

Simulation at the PVT level permits a rapid exploration of a large solution space by eliminating non-interesting regions from the DSE process. The solutions selected at this level are then forwarded to a new exploration at the CABA level. At each level, the exploration is based on developed performance and power estimation tools. Code generation at both of those levels needs parameter specifications for execution time, power estimation, and platform configurations. These parameters are specified at the deployment phase.

V. CASE STUDY

In order to demonstrate the effectiveness of the proposed design methodology, we used our MPSoC design tool with the illustrated compilation chain to generate SystemC simulation code both at the PVT and CABA levels. The selected application is an H.263 video encoder [3], represented in Fig. 1. The application, composed of three tasks (DCT, Quantization and Coding), is parallelized on an MPSoC architecture with 4 to 16 processors. In the model, at a high abstraction level, varying the number of processors is very easily done. Fig. 4 shows comparisons between the two levels of abstraction. The PVT simulation time speedup over CABA time varies between 7.2 and 12.1. This is mainly due to the fact that the high number of processors increases the communications, which are very costly to simulate at the CABA level. The difference of execution time was

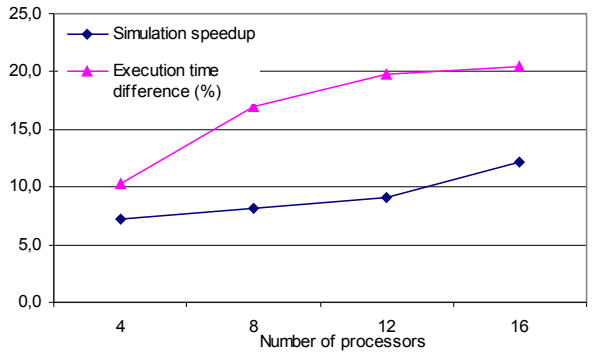

Figure 4: Comparison of results between SystemC simulations at PVT level and at CABA level.

up to 20% (PVT simulation tended to estimate more cycles). We should say that a much more in-depth study to compare the accuracy between CABA and PVT could be done, in particular with respect to the power estimation.

VI. CONCLUSION

We have presented a Model-Driven Engineering approach to generate several simulation levels from the same high level MPSoC model. In particular, MDE transformations help to factorize the work for several simulation levels and also have the advantage of being very flexible to adapt the compilation to new technologies. Moreover, we have introduced a *deployment* profile which ease the IP reuse and allows the high level system model to be platform independent. This high level also empowers the designer to manage the complexity of parallel systems.

Our framework is able to generate SystemC at the CABA and PVT levels. Future works in this domain will encompass interoperability between different simulation levels, allowing simulation at a given level even if not all the IPs have an implementation at this level. We will also be working on the automatization of Design Space Exploration, with automatic variation of the model and control of the simulation level.

REFERENCES

[1] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *IEEE Computer*, 36(4), Apr. 2003.

[2] P. Boulet. Array-OL revisited, multidimensional intensive signal processing specification. Research Report RR-6113, INRIA, Feb. 2007.

[3] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: video coding at low bit rates. *IEEE Trans. On Circuits And Systems For Video Technology*, Nov. 1998.

[4] GRACE++. *System Level Design Environment for Network-on-Chip (NoC) and Multi-Processor platform (MP-SoC) exploration*.

[5] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis. Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation. In *Conference on Languages, compilers and tools for embedded systems*, Berlin, Germany, 2002.

[6] A. D. Pimentel, P. Lieverse, P. van der Wolf, L. Hertzberger, and E. Deprettere. Exploring embedded-systems architectures with artemis. *IEEE Computer*, 34, Nov. 2001.

[7] Planet MDE. Model Driven Engineering, 2007. `http://planetmde.org`.

[8] L. Rioux, T. Saunier, S. Gerard, A. Radermacher, R. de Simone, T. Gautier, Y. Sorel, J. Forget, J.-L. Dekeyser, A. Cuccuru, C. Dumoulin, and C. Andre. MARTE: A new profile RFP for the modeling and analysis of real-time embedded systems. In *UML-SoC'05, DAC 2005 Workshop UML for SoC Design*, Anaheim, CA, June 2005.

[9] WEST Team LIFL, Lille, France. Graphical array specification for parallel and distributed computing (GASPARD-2). `http://www.lifl.fr/west/gaspard/`, 2005.