MARTE based design approach for targeting Reconfigurable Architectures

Imran Rafiq Quadri, Samy Meftali, Jean-Luc Dekeyser

MARTE based design approach for targeting Reconfigurable Architectures

Imran Rafiq Quadri, Samy Meftali and Jean-Luc Dekeyser
INRIA Lille Nord – Europe, USTL / LIFL / CNRS , Lille – France
{Imran.Quadri, Samy.Meftali, Jean-Luc.Dekeyser}@lifl.fr

Abstract—*This paper demonstrates the use of a model driven design flow for Multiprocessor System on chips (MPSoCs) such as those dedicated to intensive signal processing applications. Due to the continuous exponential rise in SoC's design complexity, there is a critical need to find new seamless methodologies and tools to handle the SoC co-design aspects. This paper addresses this issue and proposes a novel SoC co-design methodology based on Model Driven Engineering (MDE) and the MARTE (Modeling and Analysis of Real-Time and Embedded Systems) standard proposed by OMG (Object Management Group), in order to raise the design abstraction levels. Extensions of this standard have enabled us to move from high level specifications to execution platforms such as reconfigurable FPGAs.*

Key Words - Real-Time and Embedded Systems, SoC Co-design, FPGAs, Partial Dynamic Reconfiguration, ISP, Control, MDE, MARTE, UML

I.INTRODUCTION

The computing power requirements of intensive signal processing applications such as video processing, voice recognition, telecommunications, radar or sonar are steadily increasing (several hundreds of GOPS (Giga Operations per second) for low power embedded systems in a few years). If the design productivity does not increase dramatically, the limiting factor of the growth of the semiconductor industry will not be the physical limitations due to the thinness of the fabrication process but the economy. Indeed, we ask the system design teams to build more complex systems faster, cheaper, bug free and to decrease the power consumption.

Model Driven Engineering [1] (MDE) is an emerging domain and can be seen as a High Level Design Flow and an effective solution for resolving the above mentioned problems. The advantage of MDE is that the complete system (both application and architecture) is modeled at a high specification level allowing several abstraction levels. A designer thus can focus on a particular domain space related to an abstraction level. The UML (Unified Modeling Language) graphical language allows to increase comprehensibility of the system and permits relations between concepts defined at different abstraction levels. High abstraction level descriptions of systems can be provided by the users and they can identify the internal concepts (task/data parallelism, data dependencies and hierarchy). The graphical nature of these specifications allows for their reuse, modification, maintenance and extension.

MARTE [1] (Modeling and Analysis of Real-Time and Embedded Systems) is an industry standard proposal of the Object Management Group (OMG) for model-driven development of embedded systems. It add capabilities to UML allowing to model software, hardware and their relations, along with added extensions (for e.g. performance and scheduling analysis). Although rich in concepts, MARTE lacks a design flow to move from high level modeling to execution platforms.

Gaspard [2],[3] is a MDE oriented MARTE compliant SoC co-design environment dedicated specially towards parallel hardware and software co-design allowing to move from high level MARTE specifications to an executable platform. It exploits the parallelism included in repetitive constructions of hardware elements or regular constructions such as application loops. Gaspard proposes an environment starting at the highest level of abstraction, namely the system modeling level. Automate code production by the use of (semi)-automatic model transformations is possible in our environment. The environment currently focuses on a limited application domain, that of intensive signal processing applications.

Our contribution is related to the RTL chain in Gaspard which allows to convert the modeled application at the high abstraction level, as a hardware functionality which can be then implemented in a targeted FPGA. The produced code is generated automatically using model to model transformations.

The plan of this paper is as follows: section 2 gives a brief overview of MDE, while section 3 relates to Gaspard2 in general and the MARTE extensions. Section 4 briefly describes the RTL chain while section 5 presents a case study. Finally we finish with a conclusion.

II. MODEL DRIVEN ENGINEERING

MDE is centered around three focal concepts. *Models*, *Metamodels* and *Transformations*. A model is an abstraction of reality and composed of concepts and relations. Concepts are "things" and relations are the "links" between these things in reality. A model can be observed from different point of views (views in MDE). A metamodel is in fact a collection of concepts and relations for describing a model. It defines the syntax of a model as a language defines its grammar. Each model is then said to conform to its metamodel.

Finally using model transformations, it is possible to move from higher abstraction level models to low level technology modes in order to generate executable models (or source code). A model transformation is a compilation process that transforms a source model into a target model and allows to move from an abstract model to a more detailed model. The source and target models each conform to their respective metamodels. A model transformation is based on a set of rules that help to identify concepts in a source metamodel in order to create enriched concepts in the target metamodel. This separation allows to easily extend and maintain the compilation process. New rules extend the compilation process and each rule can be independently modified. Model transformations carry out refinements moving from high abstraction levels to low levels for code generation. At each intermediate level, implementation details are added to the compilation process. The advantage of this approach is that it allows to define several model transformations from the same abstraction level but targeted to different lower levels, offering opportunities to generate several implementations from a specification.

III. GASPARD CO-DESIGN ENVIRONMENT

Gaspard is a MDE oriented SoC co-design environment [2],[3]; which is compliant with the latest OMG industry standard, MARTE [1]. MARTE allows modeling of real-time embedded systems: the application, architecture and the allocation between application and architecture in the UML graphical language. Gaspard has significantly contributed in the definition of the MARTE standard, such as the RSM package which allows the expression of repetitive structures of systems (application loops and hardware repetitions) in a compact manner. The RSM concepts were originally used in Gaspard to implement intensive signal processing applications.

In the Gaspard design approach illustrated in fig.1, the application and architecture are modeled using the MARTE concepts. Afterwards, an allocation is carried out: the application part is mapped onto the available hardware resources, such as tasks on processing units and data onto memory. Although MARTE is suitable for modeling purposes, it lacks the means to move from high level modeling specifications to execution platforms. Gaspard2 bridges this gap by introducing the concept of deployment.

In the deployment level, all the elementary components or ECs, either of the application or the architecture, are linked with an implementation facilitating IP (Intellectual Property) reuse. An EC can have different implementations, which depend on the abstraction levels and execution platforms. The designer can choose an implementation among the possible choices. Hence, deployment allows one to move from platform independent models to platform dependent ones. Currently Gaspard targets different execution platforms such as formal verification [5], high performance computing [6], simulation [7],[8] and finally synthesis as illustrated in figure 1.

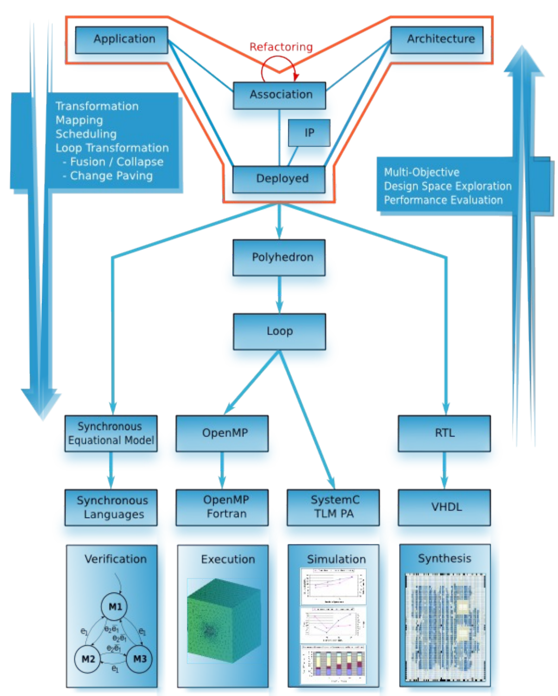

Fig.1 : The Gaspard co-design environment

IV. THE RTL TRANSFORMATION CHAIN

An application modeled at the MARTE specification level using the underlying Gaspard semantics exhibits parallelism, mainly task and data parallelism along with data dependencies. The application can be converted into a hardware functionality, i.e., a hardware accelerator, however care should be taken to ensure that the inherit parallelism remains true to its form after its conversion into a hardware design.

The RTL (Register Transfer Level) model to model transformation chain and its corresponding metamodel have been constructed to remain true to the application semantics modeled at the UML level, while providing additional low level enriched concepts in order to generate synthesizable HDL code for final FPGA implementation. The RTL level is independent from any HDL language (VHDL or Verilog for instance). The RTL metamodel relies on a factorized expression of the parallelism included in hardware accelerators. The RTL metamodel also enables the description of FPGA according to different views such as dedicated description of resources contained in an FPGA (storage, computing, etc.); another one focuses on FPGA topology (cell organization) and the last one defines the FPGA configuration zones.

Some concepts of the RTL metamodel are dedicated to the mapping of a hardware accelerator onto FPGA. These provide implementation characteristics of a hardware accelerator for a given FPGA. Such information will allow a fine topological placement of the hardware accelerator onto the FPGA.

Using model to model transformations, automatic Design Space Exploration (DSE) for an hardware accelerator can also be performed. In our design flow, the ECs can be synthesized independently to calculate the consumed FPGA resources. This information can be then incorporated into the model transformations, making it possible to calculate the approximate number of consumed FPGA resources of the overall application (at the RTL model) before final code generation and eventual synthesis.

Thus the designer is able to compare the resources consumed by the modeled application and the total resources available on the targeted FPGA resulting in an effective DSE strategy. If the application is too big to be placed on the FPGA, the designer can carry out a *refactoring* of the application. It should be noted that a refactored Gaspard application remains a Gaspard application in terms of its functionality and safeguards the parallelism.

In Figure 2 we present our MARTE based RTL design flow. Initially the application is modeled via UML and MARTE concepts; and is independent from any implementation details. Afterwards, the UML2MARTE model transformation allows to transform the UML model into a MARTE model.

This model corresponds to the MARTE metamodel. Afterwards this MARTE model is transformed into a Gaspard model by the MARTE2GASPARD transformation. Via GASPARD2RTL transformation, the RTL model is created which corresponds to a low abstraction level of an hardware accelerator (or several accelerators in the case of PDR) able to execute the initial modeled ISP application.

The RTL model provides a nearly accurate estimation of the resources required for the resulting design implementation. An exploration process (not illustrated in the figure) is performed according to these estimations. Finally, it is possible to convert the models to source code.

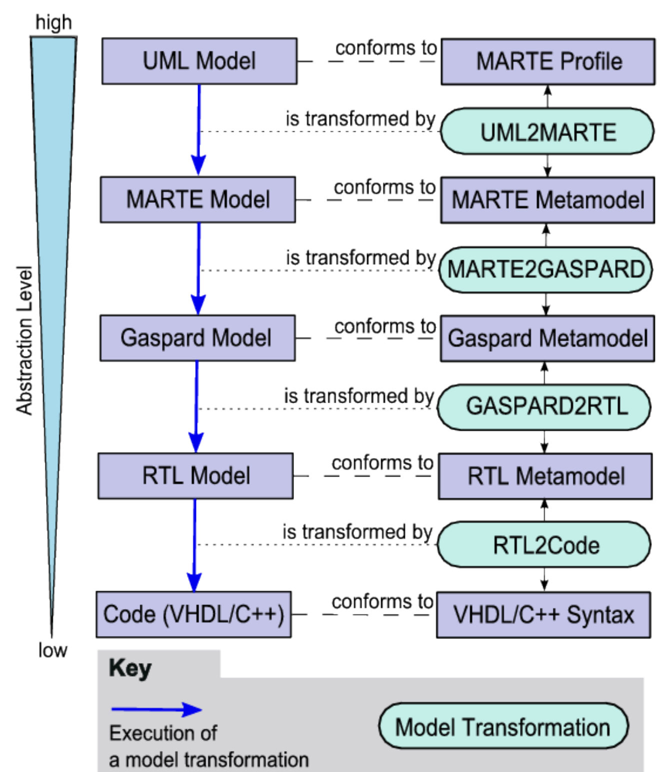

Figure 2 : An overview of the design flow related to the RTL chain

V. CASE STUDY

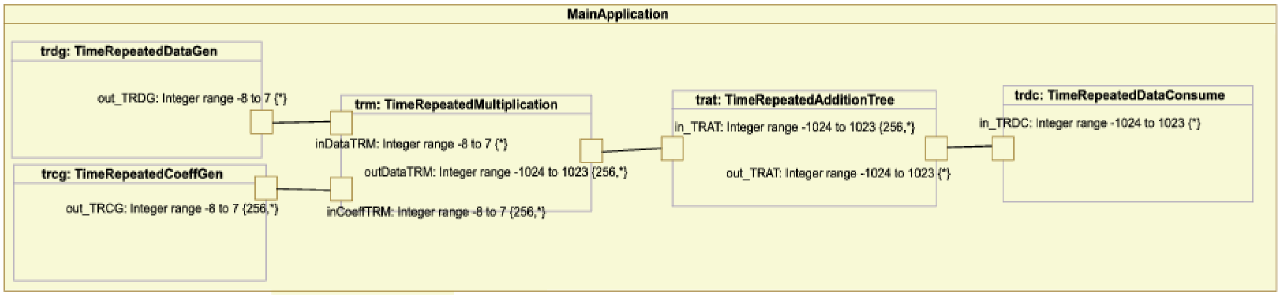

Figure 3 : Modeling of the FIR filter

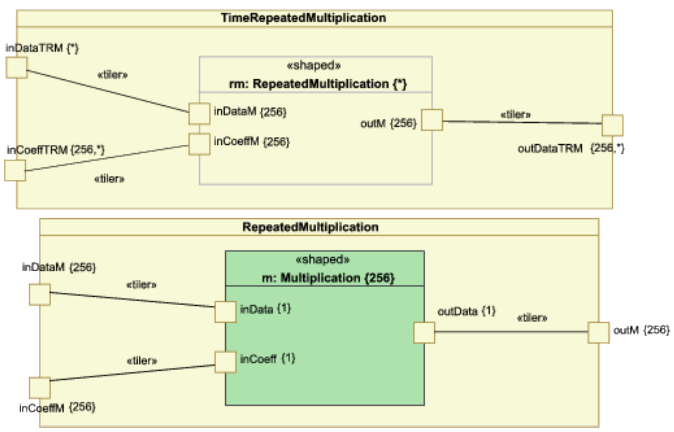

Figure 4 : Modeling of the Multiplication step

In order to illustrate the capability of our design flow, we present here a case study of a Finite Impulse Response (FIR) digital filter modeled using the MARTE concepts in the Gasapard Environment. These filters are largely employed in DSP (digital signal processing) based systems and offer a large applicability range such as linear phase and stability. The disadvantage related to these filters is that a high computational power is required and may need a large number of coefficients to reach the desired functionality. This could result in utilization of a large number of slices in a targeted FPGA. A FIR filter normally takes some input data values and compute an output which is then multiplied by by a set of coefficients. Afterwards the result of this multiplication is added together to produce the final output. While a software implementation can be utilized for implementing the FIR functionality, the filter functions will be sequentially executed. Where as a hardware implementation allows the filter functions to be executed in a parallel manner and thus increases the filter processing speed.

Figure 3 illustrates the global model of the FIR filter application. The two computing parts of the application, mainly the multiplication and the addition parts are further elaborated in detail subsequently.

The modeling of the modeling step is shown in figure 4 by means of two components. The component *TimeRepeatedMultiplication* expresses the repetition in time while *RepeatedMultiplication* expresses the repetition in space

Figure 5 represents the component which realizes the addition of 256 data elements. The input port *inAdditionTree* of this tree has a dimension of 256 while the output port *outAdditionTree* is a scalar: shape of {1}. The addition computation has been decomposed in a tree, with each stage of this tree carrying out partial additions. The dimensions of the ports between each stage in this pipeline of tasks reduce by a factor of 2 (256, 128 , ... 2, 1).

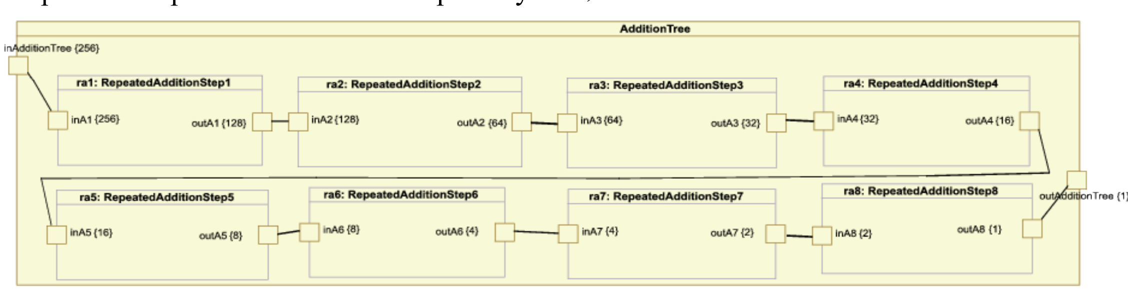

Figure 5 : Modeling of Addition Tree

Figure 6 represents the seventh stage, which realizes a partial addition of four elements on an input port and produces two elements on its output port. The computation task *Addition* is repeated 2 times and is elementary in nature.

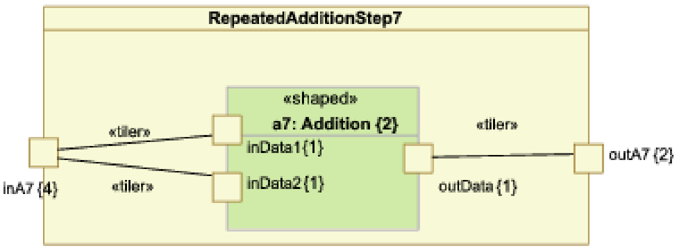

Figure 6 : Modeling of an addition step

Once the entire application is modeled, it is deployed. Figure 7 shows the deployment of one of the elementary components *CoeffGen* of the application.

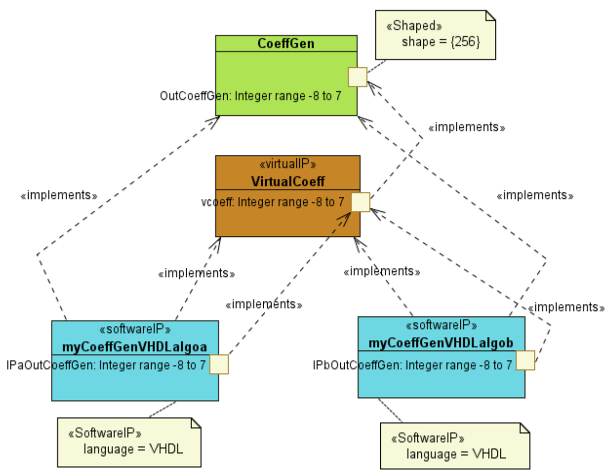

Figure 7 : Deployment of an elementary component

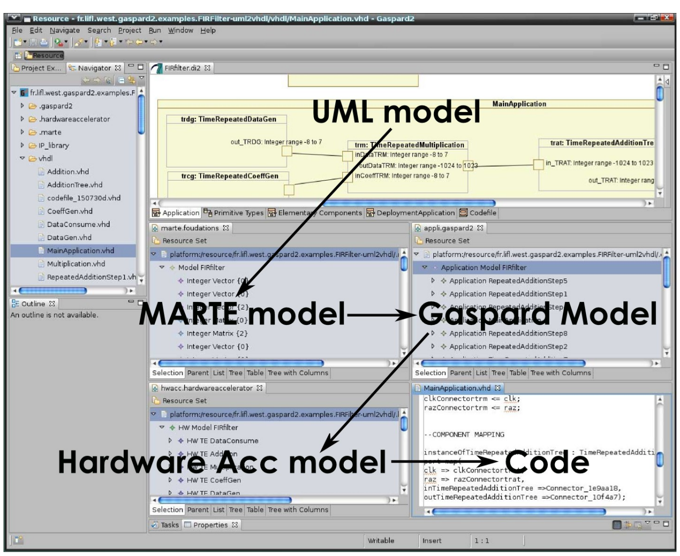

Figure 8 : Overall model transformation flow

Once deployment is carried out, it is possible to create the code via model to model transformations as shown in figure 8. The produced code can then be synthesized and implemented on a target FPGA.

VI. CONCLUSION

This paper presents the overall Gaspard environment that is now compliant to the OMG standard MARTE profile, which is dedicated to the design of embedded and real-time systems. Our contribution allow to specify complex intensive signal processing applications, which via the RTL model to model transformation chain, allows to implement the applications as hardware accelerators in a targeted FPGA. The Gaspard environment is also available as a *Rich Client Platform* (RCP) [4].

References

[1] OMG. *OMG MARTE Standard*. http://www.omgmarte.org/ . 2007

[2] A. Gamatié et al. *A Model Driven Design Framework for High Performance Embedded Systems*. Research Report INRIA, No 6614, August 2008.

[3] DaRT Team. Activity Report 2008. available at http://www.lifl.fr/DaRT/pub/public/dart.pdf . 2009

[4] DaRT Team. *Gaspard2 SoC Framework*. http://www.gaspard2.org/. 2009

[5] H. Yu. *A MARTE-based Reactive Model for Data-Parallel Intensive Processing: Transformations towards the Synchronous Model*. PhD Thesis, LIFL - USTL, France, November 2008.

[6] J. Taillard. *Une approche orientée modèle pour la parallélisation d'un code de calcul éléments finis*. PhD Thesis , Lille, France, February 2009.

[7] R.B. Atitallah. *Modèles et simulation des systèmes sur puce multiprocesseurs - Estimation des performances et de la consommation d'énergie*. PhD Thesis , LIFL - USTL, France, March 2008.

[8] E. Piel. *Ordonnancement de systèmes parallèles temps-réel, De la modélisation à la mise en oeuvre par l'ingénierie dirigée par les modèles*. PhD Thesis, LIFL - USTL, France, December 2007.