Extending IP-XACT to support an MDE based approach for SoC design

Amin El Mrabti
TIMA Laboratory
46 Ave Felix Viallet
38031 Grenoble CEDEX, FRANCE
amin.elmrabti@imag.fr

Frédéric Pétrot
TIMA Laboratory
46 Ave Felix Viallet
38031 Grenoble CEDEX, FRANCE
frederic.petrot@imag.fr

Aimen Bouchhima
TIMA Laboratory
46 Ave Felix Viallet
38031 Grenoble CEDEX, FRANCE
aimen.bouchhima@imag.fr

***Abstract*— We are interested in the problem of improving ip-reuse in SoC design. This paper presents an MDE based approach based on a proposed IP-XACT standard extension. This approach combines the benefits of using MDE techniques in SoC design such as abstraction levels definition and model transformation for code generation, and the benefits of the IP-XACT standard such as a unique exchange format of packaged IPs (Intellectual Property) with reuse capabilities.**

I. INTRODUCTION

Interoperability of SoC models from multiple vendors is a key issue in SoC design. To guarantee this objective, there is an increased use of standards such as SystemC [1] and UML profiles for SoC design (MARTE [2], Sysml [3], UML profile for SystemC [4], UML profile for IP-XACT [5]). The IP-XACT specification provided by the SPIRIT consortium represents one of the important standards used to facilitate the integration of various IP coming from different sources. It offers a standard way of describing IPs for integration purpose. IP-XACT proposes a set of XML schema to describe the components of a system. The standard targets hardware modeling at a low level of abstraction and particularly the RTL level. Although the latest versions offer support for TLM though, for example the transactional port component, the formalism is still based towards RTL modeling. The TLM level is not a model and represents a library implementing a high level approach where the details of communication are abstracted by transaction requests. We believe that IP-XACT could be a good environment for modeling HW/SW systems at higher abstraction levels. In this work, we try to improve ip-reuse in SoC design by taking advantage of the Model Driven Engineering techniques in an IP-XACT environment. We show how to extend IP-XACT to allow more formalized IP modeling at different abstraction levels with hardware dependent software representation, and how to generate SystemC code for simulation.

II. RELATED WORKS

There are several frameworks and environments for modeling embedded systems using an MDE approach. UML profiles definition for SoC design represents an important MDE activity. The SysML (Systems Modeling Language) [3] is a profile which takes a part from UML, extends the other part and defines new types of diagrams. Sequence, use case and state machine diagrams were taken by SysML. The profile defines a parametric and a requirement diagrams to structure requirements. Others profiles exist such as the UML profile for SystemC [4] and MARTE profile [2]. The MARTE profile is used to model IP-XACT designs [6]. Gaspard [7] is a tool implementing an MDA based design flow. The design flow is based on a "Y" chart design. The application and the hardware architecture are two models which represent the two sides of the Y design. Models for application and hardware architecture are done separately and each of them is conform to a corresponding meta-model. There is a third meta-model, named association meta-model, which defines relation between the functional components and the hardware components. The Gaspard flow focuses on intensive signal processing applications. Metropolis [8] is a system modeling environment based on a meta-model of computation. This meta-model provides the possibility to design various model of computation with various semantics. The meta-model specification is used to model the functionality, the architecture and the mapping. The main elements are "Process" and "Media" which represent respectively computation and communication semantics. MoDES [9] is an embedded systems design methodology which defines meta-models for application, platform and the mapping of application onto the platform. The methodology uses model transformation to define the possible mappings.

III. ABSTRACTION LEVELS, IP-XACT AND MDE BASICS

A. Abstraction levels in SoC modeling

Abstraction Levels in SoC design are used in a multilevel SoC design flow context. Intermediate abstraction levels are used to allow software generation, simulation and debug. The multilevel design flow proposed in [10] defines four abstraction levels starting from the system level and getting to the Cycle accurate level. We will focus in the third abstraction level called "Transaction Accurate" for three main reasons: (1) this abstraction level is the one that immediately precedes the RTL level (2) it represent the easiest level to integrate within the current IP-XACT specification (3) and it is interesting for

978-3-9810801-5-5/DATE09 © 2009 EDAA

the design because of the simulation results obtained in [10]. The Transaction Accurate level abstracts hardware architecture details. This specification requires the definition of software services offered by the hardware components.

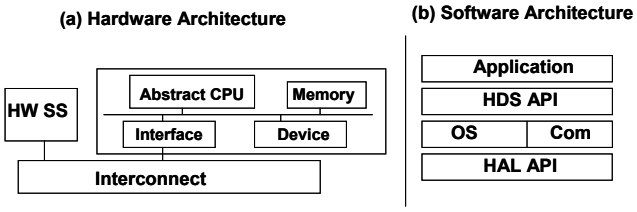

Figure 1. Transaction Accurate abstraction level

Those services are specified in the hardware abstraction layer (HAL API) of the software side (Fig 1.b). The software is linked with an explicit OS and specific communication primitives. The resulting software makes use of hardware-abstraction layer primitives (HAL API). The software is executed over an abstract model of the platform (Fig 1.a) that represents the explicit devices used by the HAL API, and the abstract processor. The simulation at this level allows validating the integration of the application with the OS and the communication layer. It may also provide precise information about the communication performances [10].

B. IP-XACT overview

IP-XACT is an independent front-end standard that allows IP packaging. It uses its own XML syntax to describe structure. The objective of SPIRIT is to facilitate the integration of various IP coming from different sources through the use of the IP-XACT specification [11]. The IP-XACT specification is provided by the SPIRIT consortium and offers a standard way of describing IP for integration purpose. SPIRIT proposes a set of XML schema to describe the components of a system. The IP-XACT model is an XML file that should be validated by an IP-XACT XML schema which represents the meta-model. IP-XACT XML schema enable hardware component modeling at the RTL and TLM levels. The IP-XACT Component [11] is the central concept. It is used to describe cores (processors, DSPs…), Peripherals (Memories, DMA controllers, Timers …) and Busses. A Design describes the component instances and the interconnection between these instances. IP-XACT represents a standard meta-modeling environment for hardware modeling.

C. MDE basic definitions

The basic entity in Model Driven Engineering is the Model. A Model is an abstract description of a system. A meta-model is a model that defines the language to create a model. A meta-model can be written in different ways (XML schema, UML diagram, etc…). The relation between a model and a Meta-model is a relation of conformance. The main MDE techniques are Meta-modeling and models transformations. Models transformations are sets of rules which describe how a model in a source language can be transformed into a model of a target language using the source and the destination meta-models. Model Transformations ensure automatic transition from a meta-model to another, by generating an output model from an input model. There are some languages which are used to implement model transformations such as ATL (ATLAS Transformation Language) [12]. The ATL language provides "ATLModule" to define transformation between source and destination models and "ATLQuery" to define transformation from a source model to text.

IV. IP-XACT EXTENSION TO SUPPORT MDE BASED APPROACH

A. MDE approach for IP-XACT SoC design

MDE techniques may improve the SoC design with IP-XACT in different ways: (1) the meta-modeling techniques can be used to define concepts to use at a defined abstraction level (2) the model transformation technique to generate simulation models in SystemC. (3) The introduction of different modeling abstraction levels may facilitate the reuse of existing components defined in different libraries. The modeling approach we propose allows modeling by using a meta-model of the desired abstraction level (Abstraction Level Definition in Fig 2.b). This meta-model will be written in XML schema and takes part of an IP-XACT extension that we call IP-XACT++. Theses meta-models will be considered as the semantic rules for each appropriate abstraction level, because the IP-XACT specification presents only the syntax side. Indeed, IP-XACT allows modeling at two levels: TLM and RTL. The concepts corresponding to those levels are gathered in the same XML schema. We want the designer to distinguish the concepts of each supported level of abstraction. This will be guaranteed by the definition of meta-models for each level of abstraction in the IP-XACT environment. This ensures a clear separation of concerns. This also ensures that the introduction of a new abstraction level, like the Transaction Accurate level seen in III.A, does not break the existing specification.

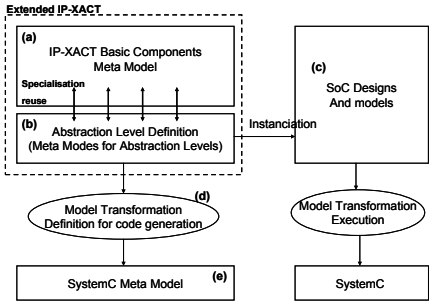

Figure 2. MDE approach for SoC design with extended IP-XACT

Figure 2 shows the MDE approach for SoC modeling. The relation between components from the abstraction level meta-models (fig 2.b) and the IP-XACT meta-models (fig 2.a) is specialization and reuse. It means that a concept corresponding to a specific abstraction level is a specialization or a reuse of a component from the IP-XACT meta-model. The SoC models will be extracted from this intermediate layer and will be conform to the IP-XACT specification (Fig 2.c). For generating SystemC code, model transformation (Fig 2.d) will be defined

between meta-models of the intermediate layer and the SystemC meta-model (Fig 2.e) inspired from [4].

B. *IP-XACT Extension: Meta-models for abstraction levels*

The meta-model for describing a level of abstraction allows the definition of constructs and rules between the different types of components for creating lacking semantics in IP-XACT. These components have a relation of specialization and reuse with the concepts defined by IP-XACT. This section will focus in the definition of the Transaction Accurate meta-model. It is written as an XML schema and represents the IP-XACT extension to allow modeling at this level. The "Execution Unit" abstracts computation in the SoC model. It is composed of one or more processing elements which are implemented with independent threads. The "Execution Unit" represents a task level view of the hardware parallelism. The "Address Space Unit" is an abstraction of the communication and the data transfer. It abstracts an addressable domain in the architecture which can be a bus, memory …. The "Synchronization Unit" abstracts the interruption process, which is a distributed process over many hardware components (interrupt controller …). The "Device Unit" in our meta-model is a functional view of the physical device represented by a set of HAL services.

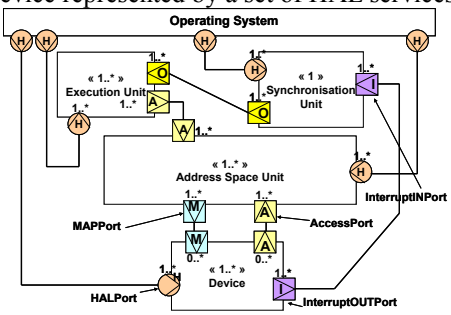

Figure 3. Transaction Accurate Meta-model

This meta-model level distinguishes four types of hardware ports and one type of software port connecting the logical units. The "Access Port" abstracts memories accesses between logic units. Executing a task by the processor requires writing data in address spaces. The Access Port is used to abstract this operation from "Execution Unit" to "Address Space Unit". The "Map Port" abstracts mapped memories. Some devices have a local memory mapped with other memory in the address space. The operation realized in the address space is automatically realized in the corresponding mapped memory of the device. The "Interrupt In Port" abstracts interruption trigger. For example, the device triggers an interruption and the synchronization unit receives this request. The "Interrupt Out Port" abstracts the process of sending the interruption from the "Synchronization Unit" to the "Execution Unit". This interruption will be handled by the abstract CPU. The "HAL Port" is a software port which abstracts the software communication. The logic units provide "HAL services" to the operating system. An initiator hardware port requires a hardware service, which is provided by a target hardware port of the same type. (The same applies for software port). The complete Meta-model is shown in Fig 3. It covers the needs for the expression of a TA model used in [10]. An IP-XACT extension for modeling at the TA Level is composed of two files; the fisrt one is an XML schema called "ta_library.xsd" which contains the definition of specific components to be used at this level. By analogy to "spirit:component" of IP-XACT, the central element is "spirit:TAComponent" which reuse the "spirit:model" and "spirit:view" elements of IP-XACT. The specialization and reuse of IP-XACT concepts are realized at this level. For example, the "spirit:HALPort" of the Transaction accurate level will reuse the "spirit:portTransactionType" concept of IP-XACT. If we were to use directly IP-XACT, we would be obliged to create a port which type is "spirit:port". Thus, we are able to create a new element "spirit:HALPort" which is a specialization of the "spirit:port" element and a reuse of the "spirit:portTransactionType". The second one is an XML schema called "ta_level.xsd" which define the interaction of component defined in "ta_library.xsd" to design a coherent model at the TA level. For example, we define at this level which type of ports are needed to define an "Execution Unit" component. With a "ExecutionUnit" component, we can define an unbounded "spirit:HALPort" ports which are in reality a reuse of "spirit:portTransactionnalPort" and a specialization of "spirit:port".

C. *IP-XACT ++ proposition*

The need to model SoC at different abstraction levels, and especially high levels, requires developing concepts able to express component at all these levels. In order to enhance the expressivity in IP-XACT, we introduce an intermediate layer which contains abstract concepts specific to the desired abstraction levels (Fig 4.b).

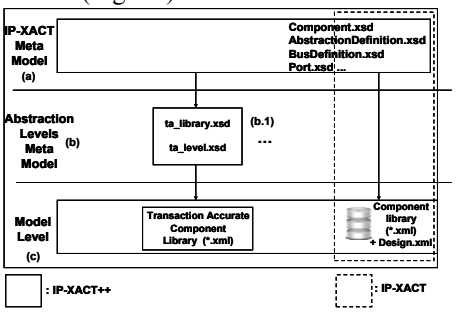

Figure 4. IP-XACT++ structure

The goal of adding this intermediate layer (Fig 4.b) is to restore the IP-XACT structure to be modular which allows the definition of specific models related to each modeling abstraction level. The Model level (Fig 4.c) in IP-XACT++ is the same as in the IP-XACT specification and different component libraries are developed with XML files. The modular structure of IP-XACT++ grants easy evolution and easy component reuse. Introducing this intermediate layer allow developing libraries of components specific to abstraction levels and thus eases IP reuse in SoC design.

V. EXPERIMENT AND INTEGRATION

An example of instance corresponding to the Transaction

accurate Meta-model is shown in Fig 5 with IP-XACT++. In this model, the "Execution Unit" element is the ARM processor. It is described as an abstract processor which provides HAL services via "spirit:HALPort" and request for data access services via "AccessPort". The HAL services offered by "Execution Unit" elements are basically context services (load_context, switch_context …).

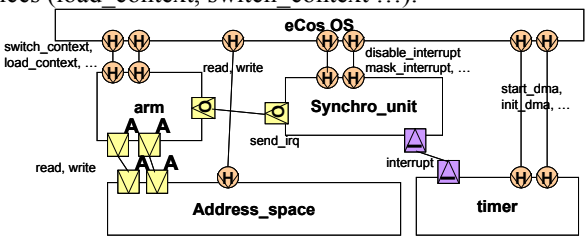

Figure 5. IP-XACT++ transaction accurate model

The data access services are access services (Read, Write). The model uses a TIMER device which provides services such as "init_timer", "start_timer". It uses a synchronization unit and address space elements too. The transformation process for SystemC code generation is detailed in Fig 6. We implement model transformations to define relations between the TA elements and the SystemC concepts with the ATL language. We use "ATL Module" to define mapping between concepts (Fig 6.a) and generate a SystemC model in a XML format. Transformations over SystemC XML model is performed with "ATL Query" which generate SystemC code (Fig 6.b).

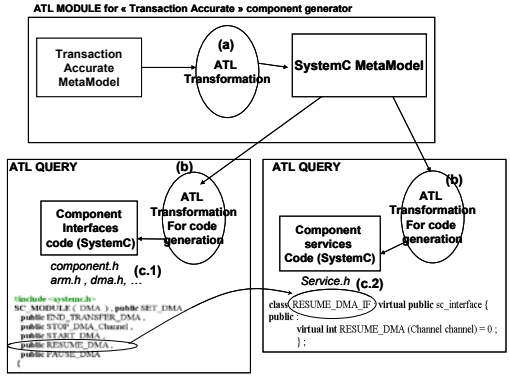

Figure 6. Transaction Accurate to SystemC Transformation for code generation.

Transformations in Fig 6.a consists of transforming a component at the TA level into a SystemC module. An initiator port is transformed into a "sc_port" and a target port into a "sc_export". The initiator port requires services while a target port provides services and has to implement them in the local SystemC module. The services described in the TA architecture are transformed into "sc_interface". Thus, we generate for each component an interface file (arm.h, mem.h, dma.h… Fig 6.c.1) and the file "services.h" (Fig 6.c.2) which contains the services interfaces. We intend to generate those files basically from the "HALservices" description of our model. The implementation of those interfaces needs the use of some predefined libraries. Limitations of this work are at:

(1) Code generation of the behavior of some components such as Execution Unit or DMA device (2) Modeling with different abstraction levels at the same time represents a point that our approach didn't take in consideration (3) Refinement of model from an abstraction level to another is still a complex task because it requires some designer decisions like the choices in mapping and in the libraries to be used in the communication and in the operating system.

VI. CONCLUSION

This paper presents an MDE approach in IP-XACT context to well formalize abstraction level models and to generate SystemC. We propose an extension of IP-XACT to support this approach. This extension consists of a set of meta-models implemented as XML schemas and used to formalize abstraction levels concepts. By means of our proposition of IP-XACT extension (IP-XACT++), we show how to integrate an abstraction level called Transaction Accurate. The integration of different abstraction levels may improve IP reuse in IP-XACT. Our future works will focus on modeling at other abstraction levels. We will focus on modeling application and software with IP-XACT++. This will offer software component reuse. The refinement between different abstraction levels will be an important activity we will focus on. We will work also in representing our proposed meta-models with UML profiles such as MARTE.

REFERENCES

[1] SystemC web site: http://www.systemc.org

[2] MARTE UML profile voted at OMG: http://www.omgmarte.org

[3] W. Mueller, Y. Vanderperren: "UML and model-driven development for SoC design". CODES+ISSS 2006

[4] E.Riccobene, P.Scandurra, A.Rossi, S.Bocchio, "A SoC Design Methodology Involving a UML 2.0 Profile for SystemC". In Design Automation and Test in Europe Conference (DATE 2005), pages 704-709.

[5] S. Révol. UML profile for TLM: contribution to formalize and automates SoC design flow and verification. Thesis document.

[6] C.André, F.Mallet, A. M. Kahn, R. de Simone. « Modeling SPIRIT IP-XACT in UML MARTE". DATE Workshop on MARTE, pp. 35-40, Mars 2008.

[7] P. Boulet, J.L. Dekeyser, C. Dumoulin, P. Marquet, "MDA for SoC Embedded Systems Design, Intensive Signal Processing Experiment", SIVOES-MDA workshop at UML 2003 San Francisco., pages 20-24

[8] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, Q. Zhu. "A Next-Generation Design Framework for Platform-Based Design," Conference on Using Hardware Design and Verification Languages (DVCon), Feb 2007

[9] F. A. M. do Nascimento, M. F. S. Oliveira, F. Rech Wagner, "ModES: Embedded Systems Design Methodology and Tools based on MDE," mompes, Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07), pages. 67-76.

[10] K.Popovici, X.Guerin, F.Rousseau, P.S.Paolucci, A.A.Jerraya: Platform-based software design flow for heterogeneous MPSoC. ACM Trans. Embedded Comput. Syst. Volume 7 Issue 4 - Article No. 39– 2008

[11] SPIRIT consortium November 2006, IP-XACT specification with ESL extensions -User Guide v1.4

[12] F. Jouault et I. Kurtev. Transforming Models with ATL. Proceedings of the Model Transformations in Practice (MTiP) Workshop at MoDELS 2005,pages128-138