Unifying HW Analysis and SoC Design Flows by Bridging Two Key Standards: UML and IP-XACT

Sebastien Revol, Safouan Taha, François Terrier, Alain Clouard, Sébastien Gerard, Ansgar Radermacher, and Jean-Luc Dekeyser

Abstract In order to save time and improve efficiency, all SoC development processes are separated into many parallel flows. These flows should keep a strong communication to avoid redundancy and incoherency. We distinguish two main trends. One aims at designing and implementing hardware when the other focuses on its functional description that may serve to software architecturing, analysis and allocation. Even if both are newly using UML, no connections have been made to synchronize them. The goal of this work is then to bridge permanently the gap between those two hardware design trends by unifying their corresponding model-based standards: UML and IP-XACT.

1 Introduction

Many initiatives are working on adapting the Unified Modelling Language (UML), for Hardware design in order to benefit from model driven development, reuse, refinement and complexity management. In electronics system design, depending on the modelling purpose, we can distinguish two main trends. One aims to implement hardware, describing circuits (structure and behaviour) using UML techniques [8][9][10][11]. The other trend focuses on functional description of hardware for analysis and allocation purposes [1][2]. These two approaches have never been efficiently unified, keeping the two modelling flows separated.

The number of the various UML diagrams enables to address many different aspects of a system. Moreover UML offers specialization mechanism for specific

Sebastien Revol · Alain Clouard
STMicroelectronics, e-mail: firstname.lastname@st.com

Safouan Taha · François Terrier · Sébastien Gerard · Ansgar Radermacher
CEA LIST, e-mail: firstname.lastname@cea.fr

Jean-Luc Dekeyser
INRIA-DaRT, e-mail: dekeyser@lifl.fr

domains, namely the UML profile capability. An UML profile is a set of stereotypes that extend UML concepts, and bring them a specialized semantics. It is the standard way to tune this general purpose language for a particular domain.

In the domain of System on Chip (SoC) design, different initiatives worked on defining profiles, having in mind a code generation purpose and using UML as a hardware design language. The UML SoC profile [10], standardized by the OMG (Object Management Group), proposes a graphical description of the SoC structure and permits SystemC[1] code generation. Likewise, the UML profile for SystemC [12] is a one-to-one transcription in UML context of all SystemC concepts including behaviour aspects. However, those profiles are often too close to the implementation languages and this has for effect to extend UML with implementation semantics. To resolve this point, our strategy was to get inspired from the IPXACT[2] concepts. This standard, widespread in the electronics community, is defined by the SPIRIT consortium with the objective to factorize in an XML grammar the hardware concepts, and to clearly dissociate the structural characteristics of a component (interfaces, registers etc.) from the way they are implemented. At STMicroelectronics we developed the ESL (Electronic System Level) profile that extends UML with IP-XACT concepts, allowing interoperability between them as well as the derivation of these formalisms into specific implementation languages.

In parallel to the hardware implementation flow, it is a common practice to specify functional, abstracted and understandable hardware models in order to communicate design intends and study interdependencies between hardware and software. Software design, allocation and analysis (e.g schedulability) require a high level description model of the hardware architecture in terms of number of processors, amount of memory Several profiles were also developed to define functional models of hardware like SPT [1] and AADL [2]. They classify resources whether if they are for computing, storage, communication and so on. These profiles are only introducing generic concepts and they are really lacking details and specific embedded systems properties. As a part of the new OMG standard MARTE [3] (Modelling and Analysis of Real-Time and Embedded systems), we developed in CEA LIST the HRM [4][5] (Hardware Resource Model) profile that is an open framework for UML-based hardware modelling. It provides many functional views and covers many detail levels.

The reason behind the separation between implementation models and functional ones, is the inadequacy between their levels and nature of details. This separation leads to redundancy and incoherency between these parallel flows. Implementation models are considered as very low level specification that cant serve for functional description. In this paper, we demonstrate that HRM profile is enough detailed, and ESL profile is enough abstracted to be able to define bridges between a functional description and an implementation one. Relying on the UML capability to provide different views of the same model, we succeeded to unify both profiles, so that they

[1] Open SystemC Initiative, www.systemc.org

[2] The Spirit Consortium: www.spiritconsortium.org

can be applied on an unique model, serving both for analysis and implementation concerns.

In the following sections, we will first present the HRM profile, its concepts and the way they can be used. Then we will introduce the ESL profile and its interoperability with IP-XACT. Last we will describe the unification process and illustrate it within a small example.

2 MARTE standard: Hardware Resource Model

The new OMG standard MARTE is proposed to replace the UML profile for Schedulability, Performance and Time (SPT). It handles the heterogeneity of embedded systems by adopting the Y-model [6] which consists of three models represented by different colours within Figure 1:

- Application model of the system tasks.
- Resource model of the execution platform, which is, in turn, composed of:
  - Software Resource Model describes the software execution platform (e.g. an operating system, drivers).
  - Hardware Resource Model.
- Allocation model that maps the application onto resources.

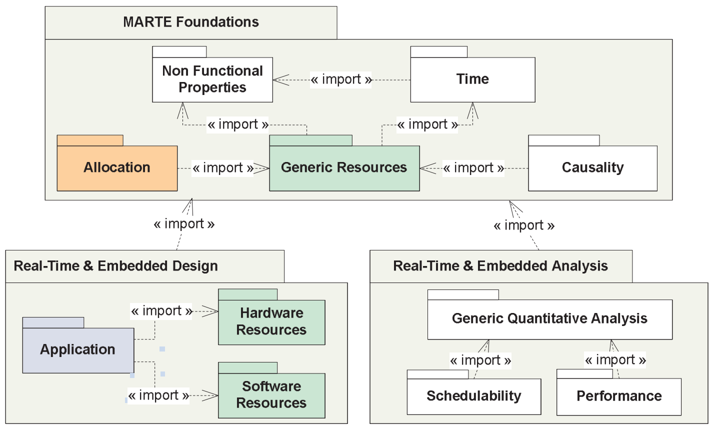


Fig. 1 MARTE structure

MARTE extends UML with a detailed Hardware Resource Model (HRM). This latter is intended to serve for functional description of hardware platforms, through different views and detail levels [4]. The Hardware Resource Model is grouping most of hardware concepts under a hierarchical taxonomy with several categories depending on their nature, functionality, technology and form. It is composed of two subprofiles, a logical one that classifies hardware resources depending on their functional properties, and a physical profile that focuses on their physical properties. HRM exploits particularly the Non-Functional Properties (NFP) package of MARTE [7] that allows quantitative annotations with measurement units and provides a rich UML library of basic types like Duration, Data Transmission Rate, Data Size and Power.

In this paper we will focus on the logical part of HRM that classifies hardware resources depending on their functional properties. The objective is then to provide a functional taxonomy of hardware resources, whether if they are computing, storage, communication, timing or auxiliary devices. This classification is mainly based on services that each resource offers. A big amount of stereotypes are introduced within HRM, they are rigorously specified and organized under a tree of successive inheritances from generic stereotypes to specific ones, no stereotype is orphan because a particular care has been made to explicit semantic relations and links among all the needed concepts. This is the reason behind the ability of the HRM profile to cover many abstraction levels. Another feature of the HRM is support of most hardware concepts thanks to a big range of stereotypes and once more its layered architecture. If no specific stereotype corresponds to a particular hardware component, a generic stereotype may match. This is appropriate to support new hardware components and new technologies. Finally, HRM includes many notations, and there is an appropriate icon for each logical stereotype.

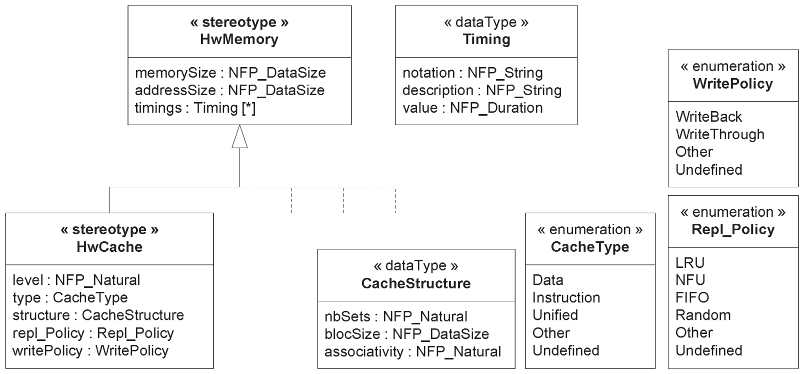


Fig. 2 HwMemory and HwCache stereotypes

In Figure 2 we extracted a part of the memory package from the logical subprofile of HRM. HwCache is a processing memory where frequently used data can be stored for rapid access. HwCache may vary depending on its level, type and structure. The cache structure is organized under sets of blocks, where associativity value is the number of blocks within one set.

In order to maximize flexibility, HRM stereotypes extend most UML structural concepts, allowing the use of the profile within any structural UML diagram. However, we provide in [5], a specific methodology to guide the hardware designers within an incremental process of successive compositions. It helps to resourcefully use HRM and benefit from its features. Finally, notice that we provide the XMI of the profile. This enables using XML-based technologies like model transformation and code generation for analysis, allocation or simulation of hardware models.

3 Electronic System Level Profile

The objective of the ESL profile is to provide a first view of the hardware architecture as a starting point of the refinement flow toward implementation, just after hardware software partitioning. Since this partitioning often leads to the reuse of existing components as well as the definition of new components, the goal of our profile is to provide both a strong IP interconnection mechanism and a way to ex-press the specifications permitting to quickly derive the implementation of new components. The following figure illustrates the role of ESL in the workflow.

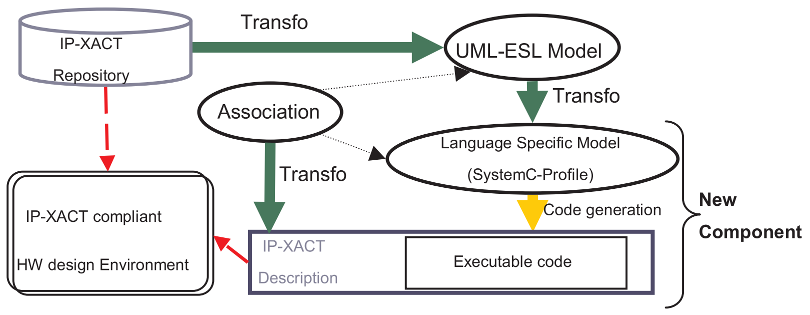


Fig. 3 Transformations workflow around the ESL profile

3.1 Positioning the profile

Regarding other initiatives, our objectives may seem similar to the OMGs SoC profile. However, the analysis of this profile led us to conclude that its semantics was very close to the old 2.0 version of SystemC. Particularly, the interconnection semantics, based on *soc_port* that can be *in*, *out* or *inout* and must be connected to *soc_interface* implementations, really constrains the SoC description to the SystemC coding-style (with *sc_in*, *sc_out*, *sc_inout* ports and *sc_interface*). This way to proceed does not provide an efficient way to describe a connection such as a master bus interface, which may be later implemented with a set of *in* and *out* ports. Moreover, the register memory map description of a component is an important concept when describing an IP, being at the frontier of the structure and the functionality of the component (since a register is a structural feature that may influence the way a component will work). This notion unfortunately does not appear in the SoC profile (neither in SystemC).

On the other side, the goal of IP-XACT is to provide a standard XML abstraction of HW components implementation files, whatever the language is (VHDL, Verilog, SystemC, etc.). Hence, they can be handled with standard compliant EDA tools, to favor the reuse of IPs. To do so, the members of the SPIRIT consortium realized a big effort to identify the concepts that represent the characteristics of a component from those that are specific to a particular implementation. Our approach was to select in the IP-XACT grammar the concepts that could be useful in an UML flow, not in order to replace IP-XACT with UML, but to provide a way to use them complementarily. Indeed, UML better fits for the definition of new components, whereas IP-XACT provides specific mechanisms for their instantiation.

The introduction of IP-XACT concepts into UML positions the ESL profile as a pivot language. As illustrated in figure 3, it enables the translation between a IPX-ACT models an UML ones. Moreover, the structural information contained in an ESL model can then be used to transform this model into a specific implementation, either directly to code, either to intermediate language specific profiles, such as the SystemC profile. Indeed the interest of relying on this intermediate model is then that it permits to complete the model with language specific concepts (including behavior) and to connect this implementation to the its ESL specification in order to generate a full coherent IP-XACT description.

3.2 Main profile concepts

Providing a behavioural description of hardware components independently from the abstraction level and the language they are implemented appeared for us a real challenge (that is not addressed by IP-XACT). Consequently, we had to focus and started by the structural description. The concepts we defined can be grouped into three main categories: the identification, the interconnections mechanisms and the register memory map.

The reuse of existing components implies to identify them clearly. To do this, IP-XACT provides the HW component with a unique identifier, based on the four attributes that are: the Vendor name, the Library to which it belongs, the Name of the component, and its last Version (VLNV). It is translated by extending the UML StructuredClass metaclass with a HWComponent stereotype, owning three tagged values (Vendor, Library and Version, the name of the component being mapped on the name attribute of the Class).

IP-XACT interconnection mechanisms is translated to UML using the port and provided/required interface UML concepts. However, IP-XACT introduces the Bus-Definition principle, which defines compatibility rules to connect together master and slave BusInterfaces. Instead of dealing only with in and out ports, the BusInterface represents a connection point of the component defined by a protocol (BusDefinition). We mainly distinguish two types of connection points: a Master- BusIf which initiates communication transactions and a SlaveBusIf that only answers them. The protocol type was expressed with the UML Interface concept, which also has to be uniquely identified with a VLNV. So, we used the provided and required interface mechanisms to express that a MasterBusIf requires the interface and must be only connected to a SlaveBusIf providing it.

The register map description relies on the Definition/Instantiation mechanism provided by the Class/Property couple. As illustrated in Figure 4, a component can instantiate several register maps that are defined by the RegisterMapDef. The latter can instantiate, in turn, several registers, characterized by several attributes such as their address offset, bit-width, multiplicity, access type (read-only, readwrite, write-only) and so on. By the same way, a register definition instantiate fields (set of bits in a register), also characterized by the same kind of attributes. Each definition concept is then mapped on a Class stereotype with the tagged values corresponding to its respective attributes, whereas each instance concept is mapped on a property stereotype, also accompanied of its tagged values.

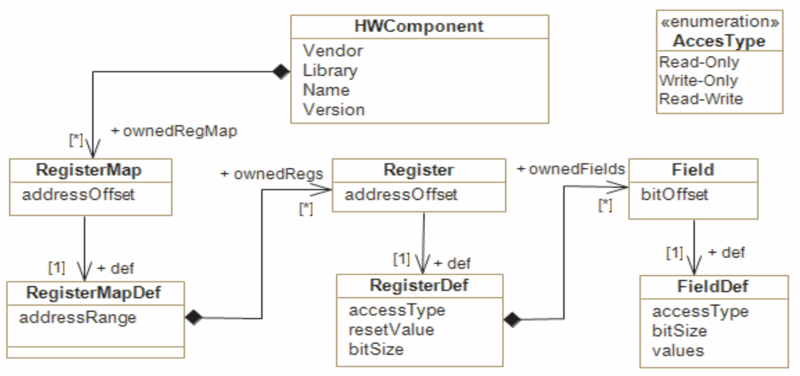


Fig. 4 Register map model

4 Unification of both approaches

Both HRM and ESL profiles permit to describe the structure of a hardware platform. In practice, they are used on the same kind of diagrams: the class diagram for the definition of components, and the composite structure diagram to describe module interconnections, and the hierarchical structure of the IP. However the concepts added to UML via the stereotypes of each profiles are not conflicting, but rather complementary. Whereas HRM brings to the model some information about the functionality of the IP, the ESL profile focuses on the way it will be implemented. On one hand, HRM introduces many stereotypes for each hardware function when ESL profile has a unique HWComponent concept permitting to identify components. On the other hand, HRM does not provide a strong interconnection semantics, with only a single stereotype to describe a connection point (HwEndPoint) when the ESL profile provides stronger connection rules distinguishing different kinds of connection points. The ESL profile also provides a fine grain description of the IP internal structure (e.g. registers) that is not addressed at all by HRM.

HRM will be useful for platform architects who want to analyse the characteristics of the system under construction, and study the mapping of an application on this platform. The ESL profile will then be used to specify and realize this platform, containing enough information to generate a big part of its implementation.

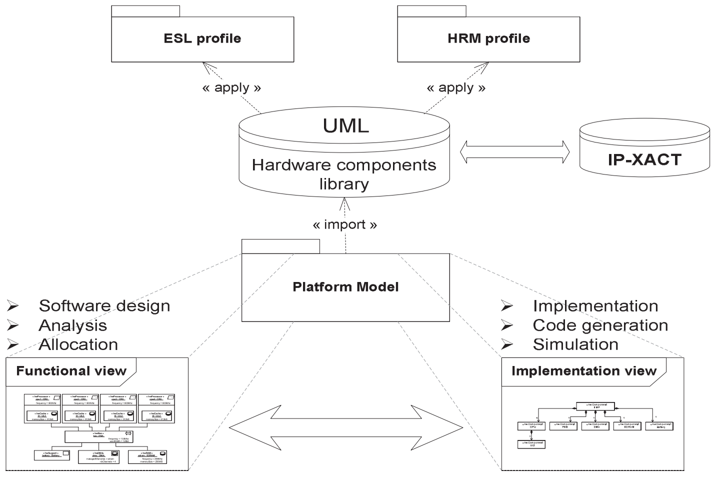


Fig. 5 Design flows unification process

UML allows the application of many stereotypes onto the same element, these stereotypes could come from the same profile as they could belong to different ones. In the first case, it means that the resource is playing many roles in the domain specified with the corresponding profile. While in the second case, it is an adequate way to merge concepts coming from different domains in the same model. In fact, we will unify different concerns that are defined in unconnected profiles into one complete hardware model, by means of multi-profile application.

The Figure 5 illustrates the development process we propose to manage the unification of design flows. First, we defined an UML library of hardware components on which we applied both ESL and HRM profiles. Each component is annotated with many stereotypes (as shown on Figure 7), there is at least one stereotype from ESL for implementation semantics and one stereotype from HRM for functional ones. This way we are filling a library of models that is conform to IP-XACT. Then, importing this library, the hardware designer may build its hardware platform by arranging and connecting components in an adequate way thanks to ESL stereotypes. Once the platform model is built and thanks to UML, we automatically provide two projections of the platform, one for implementation that only extracts the ESL annotations from the library, and one for functional purposes (e.g. software architecturing) that is HRM-based.

Therefore, two development flows are separated but keep sharing the same model, which means that they keep a strong communication between each-other. Suppose that in an incremental or refinement process, one of the design flows changed the hardware platform model, it will be automatically mirrored on the other flows view.

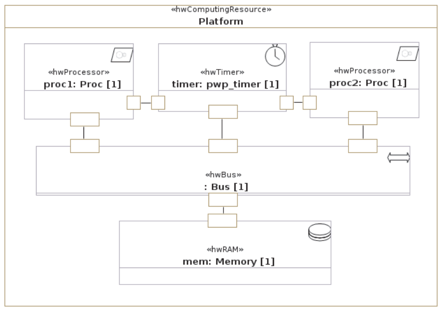
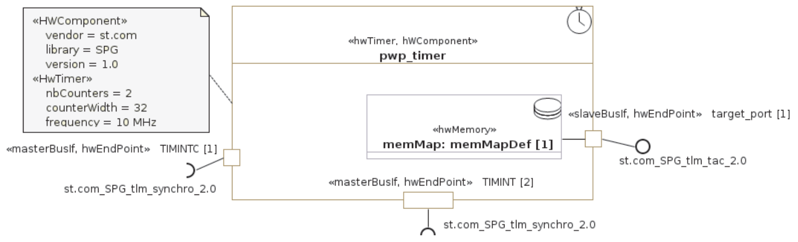


Fig. 6 Hardware platform functional view

Lets do a simple example, we create an SMP hardware platform where two processors proc1 and proc2 are sharing one system bus and the same main RAM memory mem. Figure 6 is a typical functional view of such platform model. It is used by software developers to take into account the multiprocessing aspect by designing a multi-tasks application. This view is also used by system designers for allocation or schedulability analysis, who may map each application task on one of the two processors depending on their strategy criteria and then test the adequation.


«HWComponent»
vendor = st.com
library = SPG
version = 1.0
«HwTimer»
nbCounters = 2
counterWidth = 32
frequency = 10 MHz

«hwTimer, hWComponent»
pwp_timer
«hwMemory»
memMap: memMapDef [1]
«slaveBusIf, hwEndPoint» target_port [1]
st.com_SPG_tlm_tac_2.0
«masterBusIf, hwEndPoint» TIMINTC [1]
st.com_SPG_tlm_synchro_2.0
«masterBusIf, hwEndPoint» TIMINT [2]
st.com_SPG_tlm_synchro_2.0

Fig. 7 The hardware component *pwp_timer*

Lets assume that the architectural study led to define a new timer component. The ESL profile will permit to specify its interfaces as well as its register map (Figure 7). The model transformation we developed enables the generation of more than 80% of its UML-SystemC implementation model, including base class inheritance, ports and registers declarations. It also generates the address decoding algorithms in the read/write communication API, containing meaningful debugging messages and taking into account the access type of each registers. After this step, the designer can complete its model, adding the missing behavioural features with for instance the state machines of the SystemC-profile, and generate both the full executable code and the coherent IP-XACT description. The latter allows handling this new IP in any IP-XACT compliant CAD tools.

5 Conclusion

We have presented a way to efficiently join different flows of the SoC design for which model-based approaches present interesting benefits. The ESL profile, introduced for the first time in this paper, acts as a pivot between three key aspects: the functional analysis provided by the MARTE profile, the design approaches with language-specific UML profiles and the IP-XACT industrial standard. Its level of details, compatible with the MARTE-HRM profile, enables to use both of them on a single model. This unification permits to work on one central model where three were needed before, avoiding not only a duplication of modelling efforts, but also the risk of inconsistency between the different models. Although we believe that the automation possibilities can still be improved by connecting our approach with higher level specifications processes, the efficient integration of different industrial standards we have presented in this paper let us foresee a soon adoption of this approach in a real industrial context.

References

1. Object Management Group, UML profile for Schedulability, Performance and Time (SPT), Version 1.1. OMG Document, 05-01-02.
2. Avionics Architecture Description Language Standards Document (AADL), http://www.aadl.info.
3. Object Management Group, UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), http://www.omgmarte.com.
4. S. Taha, A. Radermacher, S. Gerard and J-L. Dekeyser. An Open Framwork For Detailed Hardware Modeling In IEEE proceedings SIES2007, pages 118-125, Lisboa, July 2007.
5. S. Taha, A. Radermacher, S. Gerard and J-L. Dekeyser. MARTE: UML-based Hardware Design from Modeling to Simulation. In proceedings FDL07, Barcelona, September 2007.
6. L. Bonde, P. Boulet, A. Cucurru, J-L. Dekeyser, C. Dumoulin, P. Marquet, S. Meftaly and M. Samyn, Model Driven Engineering for Distributed Embedded Real-Time Systems, chapter Model Driven Architecture for Intensive Embedded Systems, ISTE, August 2005.
7. H.Espinoza, H.Dubois, S.Gerard, J.Medina, D.C.Petriu. Annotating UML Models with Non-Functional Properties for Quantitative Analysis, Proc of MODELS2005 Sattelite Events, Lecture Notes in Computer Science, Springer, 2006.
8. Y. Wang, X.G. Zhou, B. Zhou, L. Liang and C.-L. Peng. A MDA based SoC Modeling Approach using UML and SystemC. Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06)
9. T. Schattkowsky, J. Hendrik Hausmann, G. Engels. Using UML Activities for System-on-Chip Design and Synthesis, In proceedings of MoDELS 2006, Genova, Italy October 2006
10. Q. Zhu, R. Oishi and T. Hasegawa, T. Nakata, Integrating UML into SoC Design Process, DATE '05: Proceedings of the conference on Design, Automation and Test in Europe
11. W. Mueller, A. Rosti, S. Bocchio, E. Riccobene, P. Scandurra, W. Dehaene, Y. Vanderperren, UML for ESL design: basic principles, tools, and applications, ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design
12. Riccobene, E; Scandurra, P.; Rosti, A.; Bocchio, S., A model-driven design environment for embedded systems, Design Automation Conference, 2006