UML/MARTE methodology for high-level system estimation and optimal synthesis

Héctor Posadas, Pablo Peñil, Alejandro Nicolás, Eugenio Villar[1]

Microelectronics Engineering Group, Universidad de Cantabria,
39005-Santander, SPAIN
{posadash, pablop, nicolasa, villar}@teisa.unican.es

Abstract. Design of embedded systems is facing the challenge of their growing complexity and strict performance requirements. Model-driven design solutions are very common in this context, where the UML/MARTE profile is a well-known solution for real-time, embedded system modeling. During the design process, several specification alternatives can be considered; specifically, the HW/SW platform, concurrent application structure, application allocation into HW/SW platform resources, etc. The exploration of these design alternatives enables a set of performance estimations to be obtained in order to choose the optimal specification, facilitating system implementation and minimizing designer effort. The paper proposes an UML/MARTE methodology the enables automatic estimation of the system to be implemented. Once the optimal system specification has been defined, the proposed UML/MARTE methodology enables the final system to be implemented through an automatic synthesis process.

1 Introduction

The growing SW complexity of current embedded systems has led designers to increase the abstraction level of the first stages of the design process. Designing at higher levels of abstraction provides an effective way to deal with the complexity of large systems. The effects of creating different execution flows and deciding on different HW resource allocations have to be evaluated early in the design process. Thus, system-modeling methodologies and implementation flows have to be flexible enough to enable optimizing performance by evaluating multiple design decisions with minimal designer effort.

In this context, model-driven design methodologies based on UML are commonly adopted to handle the early design of embedded systems [1,2]. The models enable easy and fast description of the entire system, which is then used as input for the steps of code generation and integration [3]. However, pure UML usually lacks the semantics required to adequately model all the characteristics of embedded systems. As a consequence, these models are commonly developed following different profiles

[1] *This work has been founded by the PHARAON FP7-288307 and the Spanish TEC2011-28666-C04-02 MCI projects.*

which add additional semantics to the original basic UML components. Among these profiles, MARTE is gaining increasing interest for the development of real-time, embedded systems.

Taking MARTE-based models as input, several synthesis approaches have been proposed. Gaspard2 [4] is a design environment for data-intensive applications which enables MARTE description of the application and the hardware platform, generating an executable TLM SystemC platform at the timed programmers view (PVT) level. In [5] a design flow based on high-level languages (SysML, MARTE, SystemC, etc) enables the generation of the deterministic multi-threaded code for parallel implementations. In [6], a component-based modeling methodology based on UML/MARTE and explicitly designed for supporting DSE is presented. In 0[7] a semi-automatic solution for generation of HW/SW infrastructure from UML models is presented.

However, all these solutions are oriented to generating completely fixed models, especially in their concurrent structure, which limits their applicability when the system must fulfill different constraints. In that context, design space exploration of concurrency and allocation must be considered in order to find the solutions accomplishing all the requirements. In this way, the Zeligsoft infrastructure [3] supports the generation of codes for different resource allocations, but without providing simulation services that enable system constraints to be considered early in the design process. Other solutions [8,9] enable different system-level simulations, but with limited exploration capabilities. However they do not provide enough capability to explore the optimal system's concurrent architecture

To solve this problem, this paper presents a methodology to help designers to explore and automatically implement different design system concurrency architecture alternatives in the POSIX domain. The approach enables the optimization of the concurrent structure in the UML/MARTE model by easily modifying the communication semantics and interfaces used as communication mechanisms. Using this infrastructure, the models are automatically implemented, generating all the files required to simulate the architectures in a fast native co-simulation tool. Then, the designer can easily modify the model based on the performance evaluations resulting from the modifications proposed.

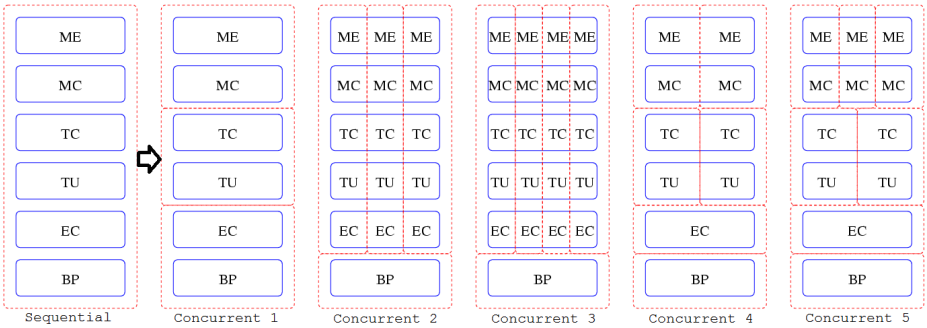

Fig. 1. Original sequential architecture of the MPEG4 application used as an example and later concurrent architectures evaluated modifying the communication semantics.

These model modifications basically focus on concurrent SW architecture and resource allocation. It is possible to modify the communication mechanisms among application components to force them to be sequential, concurrent, to create multiples copies of a component, etc. (Figure 1). Then, the communications are automatically implemented and the result evaluated using the synthesis tool developed. Thus, at the end of the exploration process, optimized synthesized codes are automatically generated for direct integration in the physical platform.

2 UML/MARTE design flow

The exploration process that can be performed by the proposed infrastructure is based on a five-step flow (Figure 2). First, the system is modeled following the proposed UML/MARTE modeling methodology, providing the C/C++ files containing the components functionality. Then, the model is automatically transformed into an executable code and, the fast, native simulation tool SCoPE [10] is called, in order to obtain performed metrics characterizing the UML model. Next, the performance metrics are analyzed by the user, obtaining conclusions about which changes in the model channels can optimize the system.

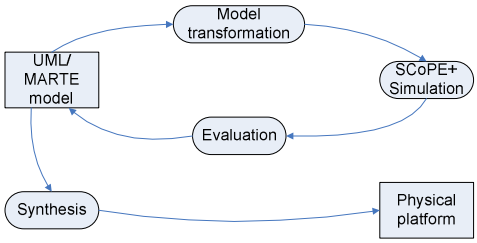

Fig 2. Proposed UML/MARTE-based optimization flow

When the system obtained fulfills the requirements, a synthesis tool is called. The synthesis tool analyzes the information of the UML model and generates all the elements required to create the executable SW to be mapped to the physical platform, including communication wrappers, main files, etc.

To support this flow, the UML/MARTE methodology proposed enables these changes to be performed by focusing on the deep modeling of communication channels as a way of enabling the modification and exploration of the application's concurrent architecture. This exploration will enable an optimal use of the HW platform resources, taking advantage of potential parallelism.

2.1 UML/MARTE Meta-modeling

The UML/MARTE system modeling methodology defined to enable the exploration and synthesis flow is a component-oriented one, following the Model Driven Architecture (MDA) principles in the development of HW/SW embedded systems. The application is divided into functional components that are connected through communication media, and mapped to the processing elements of the HW platform.

The SW components provide and require functions that are grouped in interfaces, using the MARTE modeling facilities. Additionally, new communication semantics have been added as a way to connect the provided and required interfaces of the functional components, considering different behaviors. These semantics are captured in channel models, providing a powerful, flexible and easy-to-use way to define and explore the system's concurrent architecture. Specific information about the new channel semantics can be found in [11].

At the same time, the modeling methodology is based on the idea of the separation of concerns. This separation is achieved by providing distinct system views to the designer; each one for a relevant aspect: data model, concurrency structure, communication mechanism, HW platform, SW functionality, etc.

3 Generation process

Once the model is properly created, the generation processes required to perform the exploration flow and the final synthesis can start. To do so, all the different codes required to perform the simulation and the final synthesis must be generated from the information in the UML model, as described in figure 3. Since the native co-simulation tool used in the flow (SCoPE) and the target domain supported (Linux) are both based on the POSIX API, the generated C files used to integrate all the components are valid for both steps of the flow, simplifying the process.

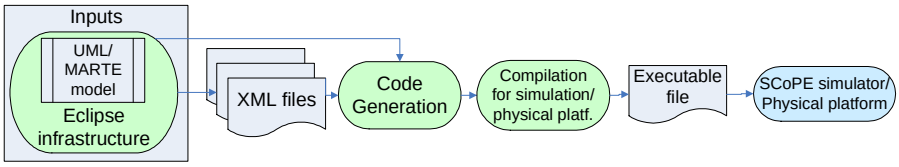

Fig. 3. Flow implemented to generate the files required for simulation and final implementation in the physical platform.

First, some generators developed in Acceleo and integrated in Eclipse analyze the UML/MARTE model, translating all the information into XML files and generating the Makefiles required during the compilation process. These files are:

• DataModelFile.xml, containing the specification of data types used in the model.

• ApplicationFunctionalityAndApplicationStructure.xml, describing external ports and internal characteristics of the components, such as their associated C/C++ files or the number of internal threads.

• Communication.xml, describing the communication mechanisms that interconnect the system components and their associated semantic characteristics.

• InterfaceFile.xml, describing the interfaces used by the application components.

• MemoryAllocation.xml, specifying the application component mapping to memory partitions

• HWPlatform.xml and SWPlatform, describing the HW components (processor, memories, buses, etc.) and the SW platform components (OS, drivers, etc.)

• Mapping.xml describing the allocation of the memory partitions to HW resources

Starting from these intermediate files, a code generator tool is called. This tool automatically generates a set of several files that are required to create the executable files and the configuration files for the SCoPE generator. The creation of the executable files includes the following steps.

The first step consists in the generation of the C files implementing the semantics of the channels. Generation of threads, service calls, data splitting and synchronization mechanisms are implemented in order to generate the concurrent architecture defined in the UML model. A file is generated for each application component defined in the model, implementing the management required to provide the communication behavior defined for each function of the component interfaces, depending on its role (provided or required).

The goals of the second step are the generation of wrappers for the interfaces related to each component. Transfer mechanisms are implemented in a generic communication library providing different implementations for inter-process communication, intra-process communication and communications between different operating systems (TCP/IP). Thus, adaptation wrappers are required to connect the generic functions of the library to each function of the component interfaces, considering the function name and the type, size and direction of the arguments and returned value. All this information is encapsulated in generic buffers that are transferred by the communication library and recovered in the target component.

When memory space definitions indicate the type of transfer required for each communication, main files are generated, one for each memory space. This enables the support of multi-OS systems, and simplifies the integration of third party codes.

Finally, all the generated C files are compiled together with the application files provided by the user using the Makefiles generated from the UML/MARTE model. Then, the compiled code can be executed in the SCoPE simulator together with the XML files required to configure the virtual platform. This compiled compiled code can also be used for is integration in the physical platform.

5 Application Example

The UML/MARTE approach proposed has been applied to a MPEG-4 encoder application, trying to decide on an optimal implementation on an OMAP4 HW platform. The MPEG-4 encoder is an industry-standard, consisting of a motion-estimation and compensation phase followed by transformation and entropy coding phases. The UML model created contains a set of functional blocks: MEMC (MotionEstimation-MotionCompensation), TCTU (TextureCoding-TextureUpdate), EntropyCoding (EC) and BitstreamPacketizing (BP).

The MPEG-4 encoder implementation used in this paper enables different system configurations to be established by modifying the channel semantics of the model, following figure 1. From an initial sequential implementation, channel semantics enable the definition of different parallel regions. Additionally, several copies of a parallel region can be called in parallel, operating with split data. Automatically generated code controls the concurrency, data management and synchronization required to interconnect all the components.

During the exploration, the 6 configurations shown in figure 1 have been evaluated with the simulation tool, as shown in table 1. As a result, one of the best configurations found ("concurrent 3") has been automatically integrated into the final platform without additional effort.

Table 1. Performance estimations obtained by the simulator during the exploration phase.

Architecture	Sequential	Concurrent 1	Concurrent 2	Concurrent 3	Concurrent 4	Concurrent 5
Estimated time	12.54 s	8.25 s	7.26 s	6.82 s	6.86 s	6.82 s

6 Conclusions

The paper presents an approach for easy exploration of concurrency architectures in embedded designs. The approach takes advantage of an extension to UML/MARTE to model the different architectures, and automatically synthesize the SW communications from the UML/MARTE models. The automatic synthesis process and the intermediate generation of XML files enable easy exploration of different allocations of SW components using the SCoPE tool. From the UML model, a generator synthesizes the communication wrappers and the main C files in a completely ad-hoc way for the application, reducing the overhead obtained with more generic design alternatives.

The approach enables easy and early exploration of the system concurrency architecture, simplifying the consideration of constraints in the design process, and obtaining the final executable file for the physical platform from a single, integrated flow.

References

1. Y. Vanderperren, W. Mueller, and W. Dehaene, "UML for electronic systems design: a comprehensive overview," Design Automation for Embedded Systems, 2008.
2. L. Lavagno, G. Martin, B. Selic. "UML for real: design of embedded real-time systems", ISBN 1-4020-7501-4
3. Zeligsoft CX, www.zeligsoft.com
4. É. Piel, R. Atitallah, P. Marquet, S. Meftali, S. Niar, A. Etien, J.-L. Dekeyser, P. Boulet: "Gaspard2: from MARTE to SystemC Simulation", proc. of the DATE'08 workshop, 2008.
5. V. Papailiopoulou, et al: "From design-time concurrency to effective implementation parallelism: The multi-clock reactive case". Electronic System Level Synthesis Conference, 2011
6. An Embedded System Modeling Methodology for Design Space Exploration. JCE 2012.
7. J. Barba, F. Rincón, F. Moya, J.D. Dondo J.C. López. "A comprehensive integration infrastructure for embedded system design", Microprocessors and Microsystems, 2012.
8. A. Pimentel, C. Erbas: A Systematic Approach to Exploring Embedded System Architecture at Multiple Abstraction Levels, IEEE Transactions on Computers, vol 55, Feb 2006
9. T. Kangas, P Kukkala et al: "UML-based Multiprocessor SoC design Framework", ACM Transactions on Embedded Computing Systems, May 2006
10. SCoPE www.teisa.unican.es/scope
11. P. Peñil, H. Posadas, A. Nicolas, E. Villar: "Automatic synthesis from UML/MARTE models using channel semantics", ACES-MB workshop, Models 2012