A Case Study of Hierarchically Heterogeneous Application Modelling Using UML and Ptolemy II

Sanna Määttä*, Leandro Soares Indrusiak[†], Luciano Ost[§], Leandro Möller[‡], Manfred Glesner[‡], Fernando Gehm Moraes[§], and Jari Nurmi*
*Department of Computer Systems
Tampere University of Technology, P.O.Box 553, FIN-33101 Tampere, Finland
Email: [sanna.maatta@tut.fi]
[†]Department of Computer Science
University of York, York, YO10 5DD, United Kingdom
[‡]Institute of Microelectronic Systems
Technische Universität Darmstadt, Karlstrasse 15, 64283 Darmstadt, Germany
[§]Catholic University of Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - P.32 - 90619-900, Porto Alegre, Brazil

***Abstract*—The heterogeneity of the state-of-the art embedded systems containing multicore platforms requires the application design flow to enable the creation of parallel applications using multiple Models of Computation. Components of such systems may interact in various ways, often using ad hoc methods. However, due to the poor reusability and analysability of ad hoc solutions, more structured approaches to heterogeneous modelling are needed. This paper takes advantage of the hierarchically heterogeneous modelling style of the Ptolemy II framework and presents a case study of building and simulating an autonomous vehicle application model using various Models of Computation.**

I. Introduction

State-of-the-art embedded systems are often heterogeneous and contain for instance analogue, digital, and mixed-signal parts as well as hardware and software requiring various Models of Computation (MoCs). Altough different MoCs have been described in detail [1] and compared [2], it is difficult to actually model or implement a system using various MoCs. However, there are several advantages when choosing a suitable MoC for each part of the system: First, since MoCs provide the designers with useful properties, such as determinism and deadlock protection as well as they ensure the preservation of properties, a suitable MoC may significantly improve the design productivity and quality. Second, choosing an appropriate MoC leads to optimal simulation speed since the simulator resources need to consider only the issues relevant to that particular MoC [3].

Hence, we need tools and modelling environments that support the modelling with multiple MoCs. Due to the dominant role of software in today's embedded systems, C-based design flows have been very popular. However, tool support for heterogeneous C-based embedded system design is not very well developed unlike Java-based tools, such as the Ptolemy II framework implemented at the University of California in Berkeley [4].

In contrast to amorphous heterogeneity (components can have various interaction mechanisms at the same time), the Ptolemy II framework presents hierarchical heterogeneity. That is, actors located at the same level of hierarchy follow the same semantic rules and each subsystem may follow the same or different rules. A MoC is a framework that specifies the interaction of components in a subsystem covering both the flow of data and the control between them. In Ptolemy II, a MoC does not only control the interaction between the components in the subsystem, but turns the subsystem into a component, which can be used in other systems. This kind of hierarchical heterogeneity having MoCs governing each subsystem allows designers to analyse the subsystems and to change them without affecting the overall system [4]. However, Feredj et al. point out that hierarchical heterogeneity creates an artificial level of hierarchy that will not exist in the real system structure. Moreover, they argue that hierarchical heterogeneity reduces re-usability and increases design costs if every MoC needs domain specific components [5].

Despite the issues in hierarchical heterogeneity, this paper presents a UML based application modelling methodology that supports hierarchically heterogeneous application modelling using UML sequence diagrams within the Ptolemy II framework. We explore how heterogeneous mixed-signal models can be built and validated using the Ptolemy II framework's approach to hierarchically heterogeneous modelling. The contribution of this paper is to build a concrete example of a heterogeneous system model in order to respond to the need of heterogeneity of the state-of-the art embedded systems. Moreover, we improve our previous work on concurrent application modelling for multicore systems as well as aim at more feasible design flow than when using only the Discrete Event (DE) MoC for the whole system.

The rest of the paper is organised as follows: Section II presents a few other heterogeneous system level modelling approaches and Section III our approach more in detail. Moreover, Section IV shows our simulation case study and finally, Section V draws conclusions.

II. Related Work

As mentioned before, C-based heterogeneous design approaches have been popular due to the dominant role of software in current embedded systems. However, for instance SystemC's standard simulation kernel supports only the simulation of the DE MoC. Therefore, the standard kernel has been extended to support for instance analogue [6] and heterogeneous [7] [8] designs. SystemC-AMS group is developing analogue and mixed-signal extensions to SystemC [6], whereas the Heterogeneous Specifications using SystemC (HetSC) introduce rules and guidelines for modelling different MoCs and enable the integration of various MoCs into the same specification [7]. SystemC-AMS does not modify the SystemC simulation kernel, but the simulation environment is built on top of the standard kernel. An opposed approach is taken by Patel and Shukla, who modify the kernel itself to support various other MoCs in addition to DE [8].

Sander and Jantsch present a framework for Formal System Design (ForSyDe) [9]. The two main goals of the ForSyDe project are heterogeneous system modelling using multiple MoCs and the development of a transformational design refinement methodology. Using the ForSyDe framework a system model can be refined into a synthesisable implementation model. The process starts from a functional system specification described with Haskell. Formally defined transformation steps are then applied until the implementation model is reached. The ForSyDe framework supports multiple MoCs, such as synchronous and untimed MoCs [9], [10].

Metropolis is a design environment for heterogeneous systems and it is designed to support Platform Based Design (PBD) [11], [12]. Metropolis contains a design methodology, base tools for simulation, and a meta-model of computation. Moreover, it has point tools for system refinement and synthesis as well as for analysis and verification [11]. The goal of the Metropolis project has been to develop a unified environment, where systems can be presented unambiguously at various levels of abstraction [12].

The Ptolemy project aims at modelling, simulating, and designing concurrent, real-time embedded systems [13]. The Ptolemy II framework supports actor-oriented modelling and hierarchically heterogeneous design having multiple MoCs for different subsystems. In the Ptolemy II framework, a domain implements a MoC. Some domains are thread-oriented (Rendezvous, Process Networks) and the components implement Java threads. Several other MoCs, such as DE, Continuous Time (CT), and Synchronous Data Flow (SDF), implement their own scheduling between actors making their execution much more efficient [14].

III. Hierarchically Heterogeneous Application Modelling

This paper presents ongoing work that addresses a few issues of embedded system design. First, this work alleviates application designers to design applications for multiprocessor systems. Moreover, it allows embedded system designers to validate the application and platform in the same model early at the design flow instead of using the traditional approach of separating the software and hardware until their integration much later in the design flow. Finally, it describes a model-based design flow for flexible modeling, joint validation of application and platform models, and fast design space exploration of Network-on-Chip (NoC) based MPSoCs early at the design flow.

In order to validate the application and platform in the same model using a same tool, we have modelled both of them within the Ptolemy II framework. We use an actor oriented NoC model, JOSELITO [15], as our platform. Moreover, we describe the application model using UML sequence diagrams that are encapsulated inside the Ptolemy II composite actors. In order to directly execute the UML sequence diagrams, we used a specific Director (in the Ptolemy II framework, a Director implements the MoC) for assigning the execution semantics to them, as described in more detail in [16].

We use an autonomous vehicle as our application model. The vehicle has sensors for sensoring for instance speed, tyre pressure, and vibration. Moreover, it has two cameras for snapshots. The vehicle can adjust its direction and speed according to the obstacle shown in the snapshots and its tyre pressure according to the information received from the sensors. The application model is described using actors. Furthermore, the application actors and their communication are described as UML sequence diagrams. A lifeline in a sequence diagram presents an application actor and a message between two lifelines communication between actors. Each lifeline is then mapped to a platform resource. That is, to a processing element connected to a NoC switch. The application actors contain one or more tasks that are executed on the processing elements of the platform.

The work this paper presents is based on our previous work of joint validation of application and platform models [17] [18]. Until now, we have used only a homogeneous system model, that is, both the application and the platform have been simulated under the DE MoC. This paper now describes how we have used the Ptolemy II framework's hierarchical heterogeneity to model a hybrid, mixed-signal application.

Figure 1 depicts the different hierarchical levels of our system model. The topmost level includes the platform model, the Mapper (the Mapper maps the application actors to the platform resources), and the application model. The topmost level runs under the DE MoC, thus the submodels also run under it if not otherwise defined. Communication networks are typically run under DE [14], therefore, our platform model runs under it. We have divided the autonomous vehicle application model to run under different MoCs, such as DE, SDF, and CT. The controller logic controlling the vehicle's speed, stability, and tyre pressure is simulated under the DE MoC. Moreover, physical characteristics, such as speed, acceleration, and position of the vehicle are modelled in the CT domain. Finally, the application parts requiring signal processing are run under the SDF MoC. The application actors and the composite actors containing the sequence diagrams are located at the second level of hierarchy. Finally, the third

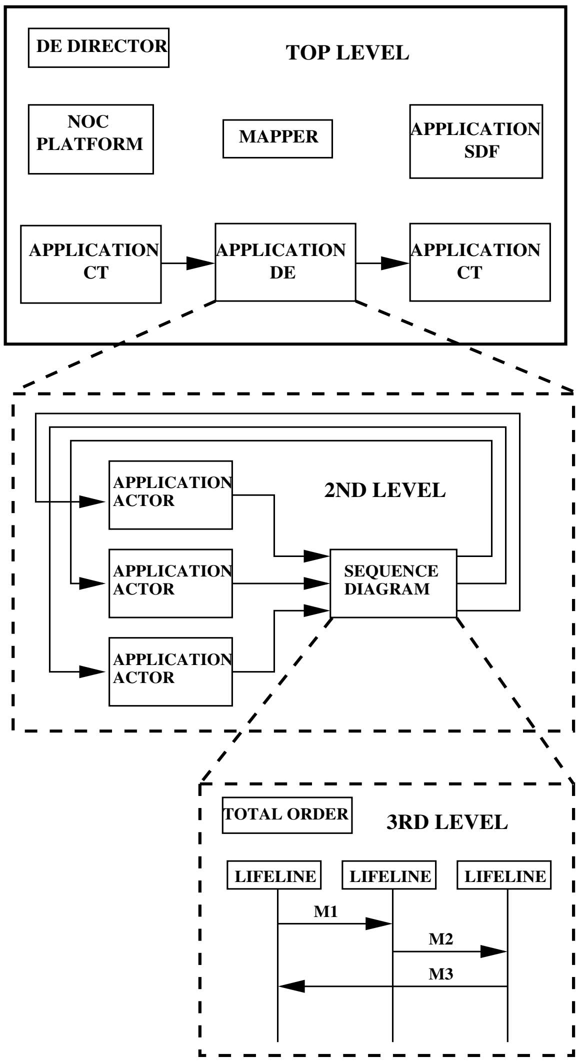

Fig. 1. Hierarchical application model

level of hierarchy contains the sequence diagrams that model the application and the Directors (Total Order in Figure 1) governing the execution of the sequence diagrams.

The messages between lifelines (M1, M2, and M3 in Figure 1) present the application tasks and communications. A message imposes computation on a processing element of the platform and after the computation is finished, a communication through a NoC platform.

IV. Case Study

We have built four different simulation cases using the same application model. The first case is a reference case, a homogeneous, non-hierarchical model using only the DE MoC. The second case is a hierarchical but homogeneous model using only the DE MoC. The third case is a hierarchical and hybrid model using the DE and SDF MoCs. The fourth case is a hierarchical and hybrid, mixed-signal model using CT, SDF, and DE MoCs. Using these simulation cases, we can explore how much the artificial level of hierarchy affects the simulation time and how much the use of multiple MoCs benefits us.

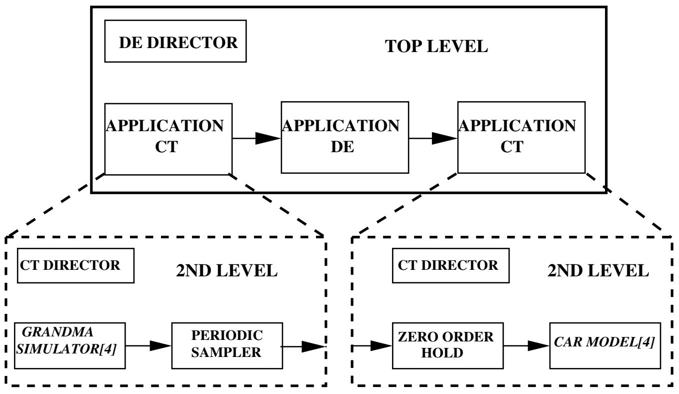

Fig. 2. CT domain inside the DE domain

From the vehicle's physical environment, we modelled the speed, acceleration, and position in the CT domain. Figure 2 illustrates the simulation setup, where a CT subsystem models the speed of the vehicle (the setup was inspired by an example of a car tracking application [20]) and a periodic sampler component changes the continuous signal into discrete, which controls the speed of the vehicle (the control logic is modelled in the DE domain). The actor controlling the vehicle's speed (Application CT on the left in Figure 2) fires on the DE domain and triggers the CT solvers to solve the differential equations for that particular time. After that, a periodic sampler transforms the continuous signal into discrete and the value the signal is carrying is forwarded to the DE part of the speed control (Application DE in Figure 2). The DE part contains the sequence diagram controlling the vehicle's speed. An incoming value triggers the sequence diagram and imposes computation to the platfrom resources and communication to the NoC. After the communication, the speed value is forwarded to the the other CT application in Figure 2. A zero order hold actor transforms the discrete signal into continuous and the car model uses the vehicles current and previous speed for calculating the vehicle's acceleration and position.

In the case study, we are building a really big and complicated model and simulating hundreds of actors when we simulate the application and platform models together. With this number of actors and complexity, we noticed that it is not easy to make various MoCs to work together. The biggest problem is the restricted amount of Java heap space. It turned out that a mixed-signal simulation containing both DE and CT MoCs uses all the available heap space very fast. Besides, the simulation of continuous time models is slow by default, due to the need to solve ordinary differential equations [3]. We managed to simulate the hybrid, mixed-signal model only for three and a half minutes wall clock time. Since we did not manage to run the simulation until the end, we cannot compare the results of a hybrid, mixed-signal model to the rest of the models.

Another problem is the static scheduling of the SDF MoC.

As a result, the arrival time of a token at any input and output port needs to be determined prior to the simulation. Because the communication between actors is defined as messages between lifelines in UML sequence diagrams, the communication is established through a NoC platform. The arrival time of a message cannot be determined in advance with absolute certainty due to the network latency. Therefore, we cannot establish a link between the SDF and DE subsystems using channels or sequence diagrams.

Furthermore, the CT model executes ahead of time when being inside the DE domain [21]. A skipped input event causes the CT subsystem rollback time. In this paper we have ignored the time consumed by rollbacks and have left it as future work to explore, how often the CT subsystem needs to rollback time and how much it decreases the simulation speed.

We simulated each model and compared the simulation times to the reference model (that is, the non-hierarchical homogeneous model). The hierarchically homogeneous model is 0.989 per cent slower and the hierarchical hybrid model is 1.34 per cent slower than the reference model. Surprisingly, the hybrid model is 0.347 per cent slower than the hierarchically homogeneous model. However, the difference is negligible, since in an approximately 24 hour simulation (wall clock time), the simulation speed difference is only about 5 minutes. From the results we can conclude that the extra layer of hierarchy has a negligible impact on the simulation time. Moreover, we do not actually benefit from using the SDF MoC in our simulation case regarding the simulation time. However, the slower simulation time might be caused by the simulator's overhead and does not mean that our hybrid application model performs worse than the homogeneous one. We were able to verify this from the simulation results: the communication and computation delays of the platform are approximately the same for both hybrid and homogeneous application models. Besides, as mentioned in the introduction, selecting a feasible MoC has also many other advantages.

V. Conclusions and Future Work

In this paper we presented a hierarchically heterogeneous application modelling method. We built four different simulation cases and compared the simulation times to the reference model. We found out that even if the Ptolemy II framework suits well for heterogeneous, multi-MoC modelling, it is still not easy to fully utilise its capabilities. Our application model does not benefit from for example the static scheduling of the SDF MoC regarding the simulation time. However, we did show that the artificial layer of hierarchy does have only a negligible impact on the simulation times.

In the future, we will improve the application model in order to benefit from the static scheduling as well as intend to establish a link between the DE and SDF parts of the application model.

Acknowledgment

This work was supported by the SYSMODEL project (http://www.sysmodel.eu)

References

[1] A. Jantsch, *Modeling Embedded Systems and SoCs*, Morgan Kaufmann, 2004.

[2] E.A. Lee and A. Sangiovanni-Vincentelli, A unified framework for comparing models of computation, in *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, vol. 17, no. 12, pp. 1217-1229, December 1998.

[3] A. Jantsch, I. Sander, Models of computation and languages for embedded system design, in *IEE Proceedings of Computers and Digital Techniques*, vol. 152, no. 2, pp. 114-129, March 2005.

[4] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neundorffer, S. Sachs and Y. Xiong, Taming heterogeneity – the Ptolemy approach, in *IEEE*, vol. 91, no. 1, pp. 127-144, January 2003.

[5] M. Feredj, F. Boulanger, A.M. Mbobi, A model of domain-polymorph component for heterogeneous system design, in *Journal of Systems and Software*, vol. 82, no. 1, pp. 112-120, January 2009.

[6] SystemC-AMS and design of embedded mixed-signal systems http://www.systemc-ams.org/ (Referenced 03/2010).

[7] F. Herrera and E. Villar, A framework for heterogeneous specification and design of electronic embedded systems in SystemC, in *ACM Transactions on Design Automation of Electronic Systems*, vol 12, no. 3, pp. 1-31, August 2007.

[8] H.D. Patel and S.K. Shukla, Towards a heterogeneous simulation kernel for system level models: a SystemC kernel for synchronous data flow models, in *Proceedings of IEEE Transactions in Computer-Aided Design of Integrated Circuits and Systems*, vol 24, no. 8, pp. 1261-1271, August 2005.

[9] I. Sander and A. Jantsch, System modeling and transformational design refinement in ForSyDe, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 23, no. 1, pp. 17-32, January 2004.

[10] ForSyDe: Formal system design http://www.ict.kth.se/forsyde/ (Referenced 03/2010).

[11] A. Sangiovanni-Vincentelli, Quo vadis, SLD? Reasoning about the trends and challenges of system level design, in *Proceedings of the IEEE*, vol. 95, no.3, pp. 467-506, March 2007.

[12] Metropolis: Design environment for heterogeneous systems: http://embedded.eecs.berkeley.edu/metropolis/ (Referenced 05/2010).

[13] Ptolemy project home page, http://ptolemy.berkeley.edu/index.htm (Referenced 05/2010).

[14] C. Brooks, E.A. Lee, X. Liu, S. Neundorffer, Y. Zhao, and H. Zheng (eds.) *Heterogeneous concurrent modeling and design in Java (Volume 3: Ptolemy II domains)*, EECS Department, University of California, Berkeley, UCB/EECS-2008-37, April 2008.

[15] L. Ost, F.G. Moraes, L. Möller, L.S. Indrusiak, S. Määttä, and J. Nurmi, A simplified executable model to evaluate latency and throughput of networks-on-chip, in *21st Annual Symposium on Integrated Circuits and Systems Designs*, pp. 170-175, Brazil, September 2008.

[16] L.S. Indrusiak and M. Glesner, Specification of alternative execution semantics of UML sequence diagrams within actor-oriented models, in *20th Annual Symposium on Integrated Circuits and Systems Design*, pp. 330-335, Brazil, September 2007.

[17] S. Määttä, L.S. Indrusiak, L. Ost, L.Möller, J. Nurmi, M. Glesner, F. Moraes, Validation of executable application models mapped onto network-on-chip platforms, in *3rd International Symposium on Industrial Embedded Systems (SIES 08)*, pp. 118-125, France, June 2008.

[18] S. Määttä, L. Möller, L.S. Indrusiak, L. Ost, M. Glesner, J. Nurmi, and F. Moraes, Joint validation of application models and multi-abstraction network-on-chip platforms, in *International Journal of Embedded and Real-Time Communication Systems*, vol. 1, no. 1, pp. 85-100, January-March 2010.

[19] S. Määttä, L.S. Indrusiak, L. Ost, L.Möller, M. Glesner, F.G. Moraes, and J. Nurmi, Characterising embedded applications using a UML profile, in *International Symposium on System-on-Chip*, pp. 172-175, Finland, October 2009.

[20] Ptolemy II CT domain, http://ptolemy.berkeley.edu/ptolemyII/ptIIlatest/ptII/ptolemy/domains/ct/doc/index.htm (Referenced 05/2010).

[21] J. Liu, *Continuous time and mixed-signal simulation in PtolemyII* Technical Report, EECS Department, University of California, Berkeley, USA, 1998.