

HARDWARE/SOFTWARE CO-DESIGN USING OBJECT-ORIENTED UML MODELS

Alexey Irkhin
TU Bergakademie Freiberg
Dept. Computer Science
irhin@mail.ru

Dominik Fröhlich
TU Bergakademie Freiberg
Dept. Computer Science
dfroehli@htwm.de

Bernd Steinbach
TU Bergakademie Freiberg
Dept. Computer Science
steinb@informatik.
tu-freiberg.de

Abstract

In this paper we present an object-oriented approach and a development environment for the system-level hardware/software co-design design of computation intensive applications using reconfigurable computer architectures. We use the Unified Modeling Language for the specification, modeling, and visualization throughout all phases of development. In this way we address a number of problems which are common to system-level development, like comprehensible and complete system representation, platform-independence, and seamless transition from specification to implementation.

1 Introduction

HW/SW co-design for object-oriented (OO) systems became a topic of interest to the research community during the last few years. The success of object-orientation for the development of complex software systems suggests to investigate its usefulness also for the development of hardware and mixed hardware/software systems. This requires the examination of specialized methodologies, algorithms for design-space exploitation and synthesis and respective tools.

We have developed a process and a supporting tool for the object-oriented system-level specification, design and implementation of applications for run-time reconfigurable computer architectures (RTR). In our approach both, the application under development, the implementation platform (languages, tools, etc.) and the target computer architecture are specified using models. The models are described with the Unified Modeling Language (UML) and an action language [6]. A dedicated UML model compiler, MOCCA (Model Compiler for reConfigurable Architectures), implements our methodology in that it automatically performs system validation, platform mapping and application synthesis.

There are many approaches to the application development for reconfigurable architectures. In [3] Chata and Vemuri presented the SPARCS design environment and a design flow. They do not address system-level specification, but high-level behavior specifications. Eisenring and Platzner [4] are developing an implementation framework for RTR applications. The system specification is done at the task-level. Vasilko et al. developed a modeling and implementation technique which builds upon the OCAPI-XL design environment [8]. The system is specified with

processes communicating through semaphores and mailboxes. In contrast to these approaches we specify systems with objects communicating through structured messages.

In Section 2 we will give a brief introduction of our UML-based co-design approach, Section 3 presents experimental results and the final section concludes this paper. For further information on issues related to modeling, design space exploration and synthesis see [1, 2, 9].

2 UML-Based Co-Design Approach

Figure 1 shows the basic activities and artifacts of our development approach. The illustrated approach incorporates the general methodology of hardware-software co-design into the concept of Model-Driven Architecture (MDA) [7]. It is a particular goal of MDA to separate the application from the platform, comprising of hardware, operating systems, compilers etc., used for its implementation. This not only supports the retargeting of the application but can also simplify the mapping of a given application to a particular platform.

In our approach the application and the utilized platforms, such as design platform and target platform, are defined by means of independent UML models. The definition of the function, structure, and behavior of the objects is captured in an executable and platform independent design model (PIM). The model of computation is objects communicating through structured messages. Non-functional requirements on implementation characteristics, such as maximum

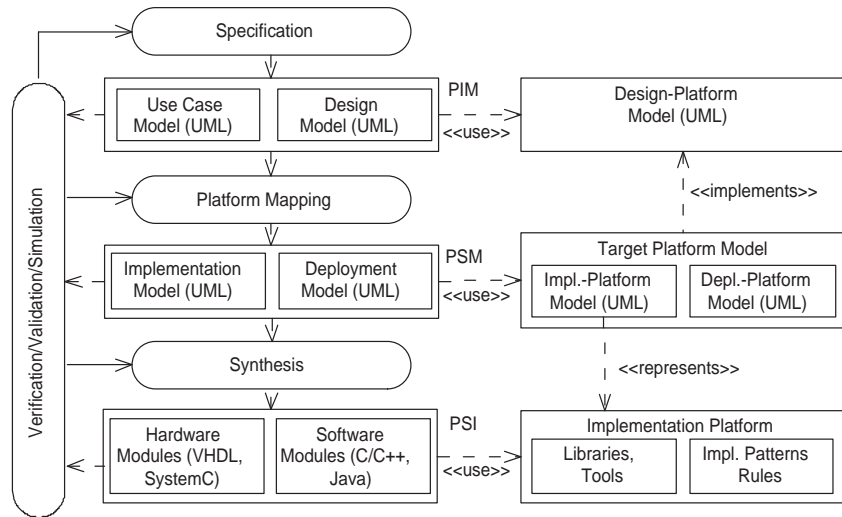


Figure 1: UML-Based Co-Design - Activities and Artifacts

latency, and area consumption, are specified using the UML extension mechanisms. Detailed behavior specification is done using a dedicated action language [1].

The design platform model makes all fundamental types and operation used for the specification of designs and non-functional requirements explicit. The target platform model defines an implementation of the design platform with the particular hardware architecture and development environment. The implementation platform is represented by its provided services (e.g. add, sub, mul, etc.) and service implementations (e.g. ripple-carry adder, carry-lookahead adder, etc.), which are characterized by their quality of service (e.g. latency, area, power dissipation). Because the platform models can be exchanged this approach features a straightforward notion portability of designs amongst different tool chains, libraries and hardware architectures.

Given a PIM of the application we transform it to a platform specific model (PSM) by mapping the PIM to the target platform model. During mapping we explore the design space defined by the target. For all behavioral and structural features of the application design we allocate sufficient service implementations of the target and estimate the quality of respective implementations. In order to improve implementation quality we perform structural and behavioral transformations. The best implementation is chosen for synthesis of a platform-specific implementation (PSI). The synthesis of software and hardware modules is parameterized by

the implementation language specific patterns and rules which are described in the respective platform models.

The methodological framework has been implemented specifically for reconfigurable architectures in the MOCCA environment. The specification of all models has several advantages in comparison to present approaches based on programming languages: First, due to the expressive power and generality of UML we can represent models of a system, from analysis to deployment, in a consistent and comprehensive manner to the user. Second, this approach supports the visualization and manipulation of all models using ordinary modeling tools. Third, the approach is independent from language specific semantics, libraries, and tools.

However, the object-oriented design space exploration and synthesis are not well researched today. The application of OO for hardware specification does not come without cost. Typical OO specifications are hard to analyze and to implement in hardware, due to their dynamic features, such as dynamic message dispatch and the common utilization of dynamic data-structures. In [1, 2, 9] we present particular transformations for UML-models and an approach to object-oriented synthesis.

3 Experimental Results

We modeled the AudioPaK encoder/decoder for high-quality audio streams with UML and our employed action language [5]. The application consists of a part that gathers the raw audio data from the audio hardware, encodes it according to the AudioPaK algorithm, and distributes the encoded streams over a network to a number of clients. The MOCCA compiler automatically generates a C++ implementation for the software modules; for hardware modules it generates VHDL-RTL implementations. The overall compilation/synthesis time of the design model into the final hardware/software modules takes approximately 5..10 minutes, depending on the degree of optimization. The generated implementations are, after a post synthesis/compilation pass, directly executable.

The coder was tested on a hardware platform comprising of a Pentium IV processor running at 2.4GHz (master) and a Xilinx Virtex-II FPGA with approximately 3Mio gate equivalents running at 100MHz (slave). Master and slave are connected with a 33MHz PCI bus. The slave implements the AudioPaK coder objects and the master is responsible for the filtering and distribution of the audio information to the coder objects and the audio clients in a network.

The FPGA implementation of one coder object requires about 1800 slices, which corresponds to 12% of the available area. Additionally two 16Kbit BlockRAMs are used to store the sample array. The majority of the resources is consumed by the encode operation. The resources required for the PCI bridge are neglectable. The FSM of the encode operation consists of 214 states, which results in a complex controller. In Figure 2 the performance characteristics of the coder implementation is shown. The coder implementation is efficient. Depending on the frame size between 15 and 20 audio channels can

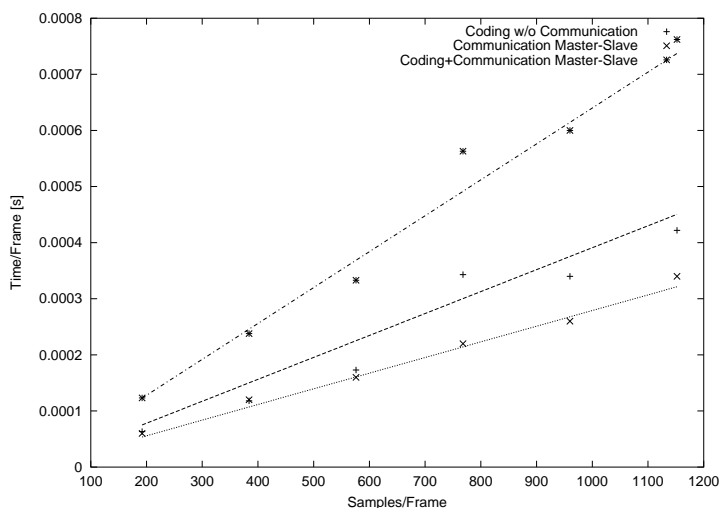


Figure 2: Experimental Results for AudioPaK Coder

be encoded concurrently with one coder object at 96KHz sampling rate. To avoid contention on the objects we beneficially use a few coder objects that are executed in round robin order.

4 Conclusions

In this paper we presented an approach and a supporting environment for the object-oriented system-level development of applications for reconfigurable computer architectures. To overcome some of the drawbacks of approaches which use conventional programming languages for system specification, we use the universal syntactic and semantic framework provided by the Unified Modeling Language. This approach enables a seamless and comprehensible transition of the system specification to a ready-to-run application, without the need to change the applied paradigms, methods, or language in between. This not only has the potential to decrease overall development time and cost, but also improves the quality, because error-prone manual translations and interpretations are avoided. The capabilities of the presented approach have been demonstrated with a real world design of a coder for high quality audio streams. With this example it has been shown that the approach provides significant gains in system quality and development efficiency.

References

- [1] Thomas Beierlein, Dominik Fröhlich, and Bernd Steinbach. UML-Based Co-Design of Reconfigurable Architectures. In *Proceedings of the Forum on Specification and Design Languages (FDL'03)*, Frankfurt a.M., Germany, September 2003.
- [2] Thomas Beierlein, Dominik Fröhlich, and Bernd Steinbach. UML-Based Development of Applications for Run-Time Reconfigurable Architectures. In *Proceedings of the DAC'2004 Workshop on UML for SoC Design (UML-SoC'04)*, San Diego, California, USA, 2004.
- [3] Karam S. Chata and Ranga Vemuri. Hardware-Software Codesign for Dynamically Reconfigurable Systems. In *Proceedings of the 9. International Conference on Field Programmable Logic and Applications (FPL'99)*, September 1999.
- [4] Michael Eisenring and Marco Platzner. An implementation framework for run-time reconfigurable systems. In *The 2nd International Workshop on Engineering of Reconfigurable Hardware/Software Objects (ENREGLE'00)*. Monte Carlo Resort, June 2000. Las Vegas, USA.
- [5] Mat Hans and Ronald W. Schafer. Lossless compression of digital audio. Technical report, Client and Media Systems Laboratory, HP Laboratories Palo Alto, 1999.
- [6] Object Management Group. OMG Unified Modelling Language Specification (Super Structure). <http://www.omg.org>, July 2003. Version 2.0.
- [7] Object Management Group - Architecture Board ORMSC. Model driven architecture - a technical perspective (mda). <http://www.omg.org>, June 2001. Draft.
- [8] Tero Rissa, Milan Vasilko, and Jarrko Niittylahti. System-level modelling and implementation technique for run-time reconfigurable systems. In *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)*, 2002.
- [9] Bernd Steinbach, Christina Dorotska, and Dominik Fröhlich. Synthesis of UML-Models for Reconfigurable Hardware. In *Proceedings of the International Conference on The Experience of Design and Application of CAD Systems in Microelectronics (CAD SM'05)*, Lviv-Slavske, Ukraine, 2005.