Recoding Embedded Applications into Flexible System-Level Models

Rainer Dömer
Center for Embedded Computer Systems
University of California, Irvine, USA
doemer@uci.edu

***Abstract*—Before we can ask "Quo Vadis, Virtual Platforms?", we should ask ourselves "Unde venis, Virtual Platforms?" and discuss their origin. In this paper, we argue that virtual platforms originate from embedded applications and form an executable system model that in more or less abstract form specifies an implementation of the original application in target hardware and software.**

Specifically, we discuss the key concepts in system modeling which are needed explicitly in a virtual platform, such as the clear separation of computation and communication. We also describe a method, called computer-aided recoding, that allows to derive a virtual platform model directly from original reference code of the application. In the recoding process, flat and sequential C code is converted into a flexible and parallel system model that exhibits the required features of structural hierarchy, explicit concurrency, and exposed communication, and thus can serve as an effective virtual platform for further design space exploration, functional validation, and system synthesis.

I. Introduction

Virtual platforms are seen as an important technology to cope with the design challenges created by the constantly growing complexity of embedded systems. A virtual platform can serve as a stable intermediate milestone on the long way of designing a suitable implementation in hardware and software for a given application. While the design path from a virtual platform down to its target implementation is challenging (it includes the complex tasks of design space exploration, functional validation, and system synthesis) and the definition and specification of the virtual platform itself is still evolving (*Quo Vadis, Virtual Platforms?*[1]), this paper focuses on the origin of virtual platforms. *Unde venis, Virtual Platforms?*[2]

A. Virtual Platforms

Virtual platforms are usually built as an abstract software model of a target hardware platform for a set of embedded applications. Given the virtual platform, an application can be developed, executed, and evaluated before the actual hardware platform becomes available, saving precious design and development time. At the same time, virtual platforms typically offer advanced simulation and debugging features that are not available in the real target hardware.

[1]Where are you going, Virtual Platforms?
[2]Where do you come from, Virtual Platforms?

QVVP'12, "Quo Vadis, Virtual Platforms?", Workshop at DATE'12

While virtual platforms can be manually built from scratch (or refined from a previous version of a similar model), we describe in this paper a system design flow that includes the automatic generation of a virtual platform. Starting from reference code of the application, we propose a *recoding* technique that allows to generate an executable system model that can serve as a flexible virtual platform for further implementation.

II. Computer-Aided Recoding

In contrast to the application reference code, which typically is given in the form of flat and sequential C code, System-level Description Languages (SLDLs), such as SystemC [9] and SpecC [8], allow designers to describe hardware and software components together. These SLDSs support specific constructs for the clear separation of computation and communication.

Specifically, the C-based SLDLs support the following key system modeling concepts:

- Explicit structure: block diagram structure and connectivity through ports
- Explicit hierarchy: system composed of components
- Explicit concurrency: potential for parallel or pipelined execution
- Explicit communication and computation: channels and interfaces, vs. modules/behaviors
- Explicit timing: simulation time and timing constraints

Having these intrinsic features of an application explicitly described in its design model enables efficient design space exploration and automatic refinement by computer-aided design (CAD) tools.

Now, *computer-aided recoding* allows to derive a virtual platform model with these explicit system concepts directly from the original reference code of the application. In the recoding process, the flat and sequential C code is converted into a flexible and parallel system model that exhibits the required system features and thus can serve as an effective virtual platform.

Computer-aided recoding automates various steps in the process of writing system models. We use a designer-controlled approach that relies on automated source code transformations available to the system designer in form of an integrated development environment [1]. Here, the designer makes the decisions, whereas the tool automatically transforms the source code.

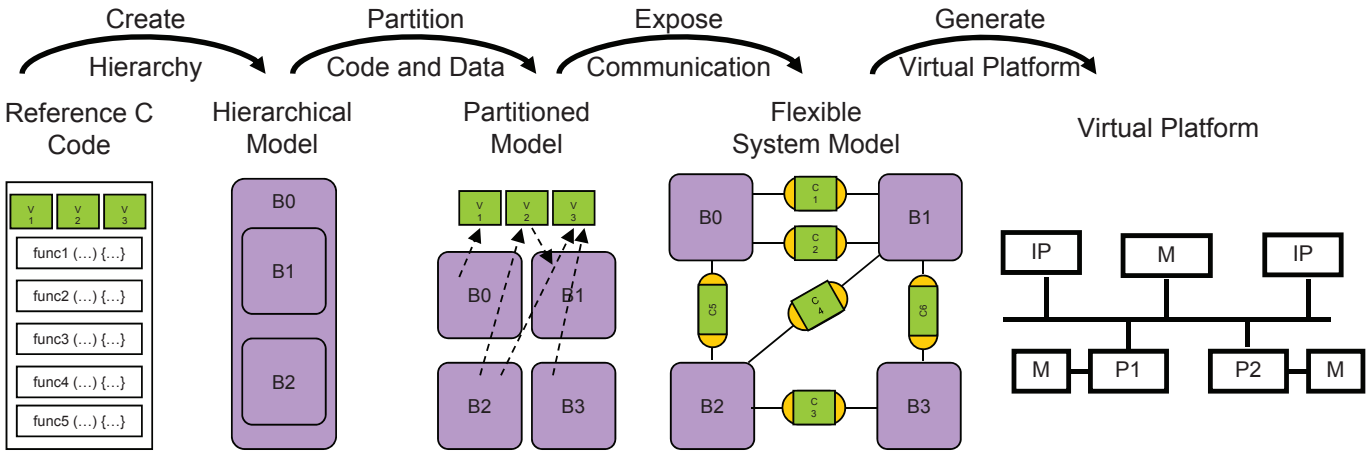

Fig. 1. Recoding application reference code into a flexible system model for automatic generation of virtual platform models.

As illustrated in Fig. 1, the recoding process consists of several types of source code transformations, including (1) creation of structural hierarchy [4] to properly organize the initially unstructured (flat) application code, (2) code and data partitioning [2], [5] to create a parallel and flexible system model, (3) creation of explicit communication and synchronization [6] to enable plug-and-play in the system model, and (4) pointer recoding [3] to eliminate unwanted pointers in the given reference code. The result of this recoding process is a flexible system model that can be fed into a regular system design flow, such as the System-on-Chip Environment (SCE) [7]. From here, several virtual platform models at different levels of abstraction, e.g. a Transaction Level Model (TLM) or Bus-Functional Model (BFM), can automatically be generated.

In summary, computer-aided recoding can derive an executable parallel system model directly from available sequential reference code. Automatic source code transformations relieve the system designer from complex code analysis and tedious coding tasks, allowing uninterrupted focus on system modeling and design space exploration. As a result, an application-specific virtual platform can be quickly generated, enabling a shorter design time and higher productivity.

III. Conclusion

Given the constantly growing complexity of the digital systems around us, virtual platforms are essential in the design and development of today's embedded systems. Without virtual platforms, efficient design space exploration, effective functional validation, and cost-effective system implementation would not be possible.

In this paper, we outlined a method to automatically generate virtual platforms via a flexible system model which can be built from the original application source code by use of computer-aided recoding. Thus, using recoding and model generation a virtual platform matching the application's needs can be derived directly from the original reference code of the application.

Acknowledgment

This work has been supported in part by funding from the National Science Foundation (NSF) under research grant NSF Award #0747523. The authors thank the NSF for the valuable support. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] P. Chandraiah and R. Dömer. An Interactive Model Re-Coder for Efficient SoC Specification. In A. Rettberg, M. C. Zanella, R. Dömer, A. Gerstlauer, and F. J. Rammig, editors, *Embedded System Design: Topics, Techniques and Trends*, Boston, MA, 2007. Springer.

[2] P. Chandraiah and R. Dömer. Designer-Controlled Generation of Parallel and Flexible Heterogeneous MPSoC Specification. In *Proceedings of the Design Automation Conference (DAC)*, June 2007.

[3] P. Chandraiah and R. Dömer. Pointer re-coding for creating definitive MPSoC models. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, Salzburg, Austria, September 2007.

[4] P. Chandraiah and R. Dömer. Automatic re-coding of reference code into structured and analyzable SoC models. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*, Seoul, Korea, Jan. 2008.

[5] P. Chandraiah and R. Dömer. Code and Data Structure Partitioning for Parallel and Flexible MPSoC Specification Using Designer-Controlled Recoding. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems (TCAD)*, 27(6):1078–1090, June 2008.

[6] P. Chandraiah, J. Peng, and R. Dömer. Creating Explicit Communication in SoC Models Using Interactive Re-Coding. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan, Jan. 2007.

[7] R. Dömer, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu, S. Abdi, and D. Gajski. System-on-Chip Environment: A SpecC-based Framework for Heterogeneous MPSoC Design. *EURASIP Journal on Embedded Systems*, 2008(647953):13 pages, 2008.

[8] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Design Methodology*. Kluwer, 2000.

[9] T. Grötker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Kluwer, 2002.