Methodology and tools for performance analysis of embedded multiprocessor industrial applications

Ismail Assayad and Sergio Yovine
ismail.assayad@imag.fr sergio.yovine@imag.fr
CNRS/VERIMAG France

Abstract— **We present a framework and its supporting simulation tool for modelling and performance analysis of multiprocessor embedded systems. Our framework consists of component-based models for modelling parallel software and multiprocessor hardware, and tools for code generation and performance analysis. The framework component meta-model relies on transaction-level description of hardware and programmer-level description of software where timing properties of the hardware micro-architectures are modelled as annotations in the concurrency meta-model. The framework has the advantage of allowing fast precise and scalable joint analysis of software and hardware performance. Keywords: Multiprocessor Embedded Systems, performance, software/Hardware Analysis.**

I. Introduction

Video compression, HDTV, packet routing and other high performance embedded applications motivate the use of off-the-shelf, configurable, heterogeneous hardware platforms offering multiple processing units, such as Philips' VIPERand Wasabi/Cakearchitectures, and Intel's IXP family of network processors. However, the complexity of such *multiprocessor embedded systems* (MES) makes software programming and analysis difficult, leading to sub-optimal software and hardware performances. An integrated software/hardware modelling and performance analysis methodology, supported by the appropriate tools, gives system developers means to improve field upgradability and time to market, and therefore lower development costs, of embedded product lines.

A. Current practices

Several techniques have been proposed to address this issue. These techniques are classified into three categories according to their modelling scope, namely *software*, *hardware* and *platform* based approaches; and into two categories according to their modelling method, i.e. *analytical* and *simulation* approaches.

In contrast to software- and hardware-based design, platform-based design (PBD) [1] provides the adequate level of abstraction that can be used for analyzing the impact of software implementation choices into hardware *micro-architectures* performance, and evaluating the impact into software performance of changing a hardware configuration parameter.

Simulation-based techniques use either *wrapper*-based or *annotated* approaches. In the wrapper-based approach, the actual timing behavior is modelled independently of the micro-architecture's functionality model in such a way that delays are computed during the execution of an external existing timing model. Wrappers then have to synchronize between timing and functionality. In the annotation-based approach, timing delays at the micro-architecture level are given as annotations attached to the functional model operations.

B. Related work

Most PBD approaches found in the literature are not thought to provide complete solutions for MES performance modelling and analysis. ARTS [2] aims at porting communication concurrency modelling of micro-architectures at system-level by describing them using an abstract RTOS model. This approach uses annotated DAGs of tasks for modelling software and reduces the hardware model to communication latencies. Therefore, it does not resolve software timing into hardware performance. METROPOLIS [3] provide general purpose frameworks in the sense that they do not make any assumption about the functional and timed models of micro-architectures. This has the advantage of broadening the applicability of the frameworks for modelling concurrency at unfixed abstraction levels. Nevertheless, none of these approaches proposes a *methodology* for modelling and micro-architectures performance analysis (and, thus, concurrency) which resorts to the specific skills of the designer. The industrial applications of SystemC/TLM [4] and MPARM [5] use the wrapper-based timing model as a method for performance modelling. Here, we propose an original PBD approach for, not only modelling concurrency, but also handling the issue of performance modelling and prediction of MES, using time-annotated simulation meta-models.

C. Contribution

In addition to the generality of its application to MES modelling and analysis, the contribution of our PBD approach are manyfold. First, it provides *components* meta-models for assembling and binding micro-architectures and software components without the need of interfacing wrappers, and a formal semantics for composing software tasks and hardware. Second, components are self-contained with a well-defined structure composed of a set of blocks whose characteristics are shown to be pertinent not only for capturing MES concurrency but also for precise *predictions* of MES performance.

Third, our simulation tool, P-WARE, provides *joint*, *scalable*, and *fast* performance analysis of concurrent embedded software and multiprocessor hardware. Fourth, the *methodology* supported by our tools starts from a high-level modelling of software and hardware, all the way down to the implementation of software on the hardware, while going through the synthesis of an application-specific software scheduler, the analysis of software-hardware joint performance simulation, and the generation of executable code.

II. Framework overview

We developed a PBD framework which supports MES modelling at transaction-level for hardware components and task-level for software components. We also proposed a methodology for composing software and hardware and analyzing their joint performance. To support this framework, compilation and simulation tools which enable automated settings of the components, fast performance prediction and automated code generation have been developed.

A. Tools

1) Jahuel: The first tool is a compilation chain, called JAHUEL, for a high-level formal language, called FXML, for modelling software hierarchical task graphs [6]. The purpose of the language is threefold. First, it provides simple and platform-independent constructs to specify the behavior of the application using an abstract execution model. Second, it provides semantic and syntactic support for correctly refining the abstract execution model into the concrete one. Third, the language and the compilation chain are extensible to easily support new concrete execution models, without semantic break-downs.

2) P-Ware: The second tool is a SystemC-based simulation platform, called P-WARE [7], for jointly predicting and analyzing performance of software and hardware components generated by JAHUEL. The hardware view decomposes software tasks into flows of components' transactions. P-WARE combines several hardware transaction-level components and programmer-level software components and allows for composing software tasks with hardware.

B. Methodology

Setting up a system with this framework is operated as follows. First of all, JAHUEL is used to define the models of the application and the architecture. The application is the software (eg., a video encoder) and its environment (eg., camera and display devices), while the architecture is the underlying execution hardware (eg., processors, buses, caches, etc.).

Then, the step "constraint synthesis" [8] derives a scheduler of this application and a constraint on the parameters which must be satisfied at runtime, by any parallel mapping of software which will be defined later. After that, we look for software implementations, i.e., mappings, which must satisfy the latter constraint. This is done by considering several classes of parallel implementations (eg., data parallel, task parallel and hybrid implementations).

Software components corresponding to each of these implementations are generated, in addition to the hardware ones and their bindings, as input to P-WARE. The correctness of the implementation is guaranteed if two performance objectives are met: (1) software constraints are satisfied, and (2) the predicted hardware performance (eg., available bandwidth) covers the environment communication needs. The first condition states that the requirements of the application model are met. The second condition guarantees that hardware-dependent issues will not affect the predicted software execution times.

If performance objectives are met, JAHUEL synthesizes the executable code of software.

III. VIDEO ENCODING PLATFORM

Let us see the results of applying the framework to the encoding platform of figure 1. The software, which runs on the processors, is made up of a format converter (FC) and a video encoder (VE), in charge of converting then encoding the images in a MPEG-4 format.

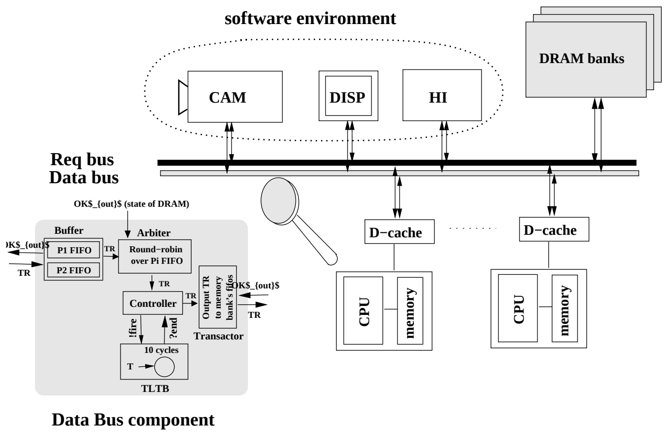

Fig. 1. Video encoding platform

The execution times of the software including communication are treated as parameters, whereas timing values of the environment (i.e. CAM, HI, and DISP) are known and equal to a half of the period : $\frac{1}{15}$. The synthesis procedure [8] derives a scheduler and a constraint over the execution times of the software. The constraint is a deadline of $\frac{1}{25}$ over the execution time of the encoder VE. It gives the condition under which the scheduler respects the requirements of the model.

We programmed several implementations: a is the sequential implementation, b, c and d are implementations of the first class with data oriented parallelism using two, three and four processors respectively. e, f and g are hybrid implementations using two, three and four pipelines respectively.

Implementation	Proc	execution times of VE (s)	used bandwidth (MB/s)	cache misses (%)
a (sequential)	1	0.13	320.43	2.28
b (parallel)	2	0.06	426.14	1.28
c (parallel)	3	0.044	570.90	1.25
d (parallel)	4	$\boxed{0.036} \leq \frac{1}{25}$	630.9	1.33
e (hybrid)	8	0.05	445.8	1.30
f (hybrid)	12	$\boxed{0.036} \leq \frac{1}{25}$	600.9	1.28
g (hybrid)	16	$\boxed{0.032} \leq \frac{1}{25}$	682.72	1.27

TABLE I
PREDICTED PERFORMANCE

Table I shows the predicted performance results for these implementations, by using P-WARE. This joint analysis shows that implementations d, f and g are realizable implementations. However, implementation d is by far the one which provides the best compromise since it consumes slightly more bandwidth than f but uses less processors than f and g.

IV. CONCLUSION

We have proposed a framework and a simulation tool for analyzing software and hardware performance rather than inefficiently evaluating each one in isolation. The framework relies on an annotated and component-based modelling and analysis approach which combines transaction-level hardware and task-level software models.

The framework has the advantage of providing (1) component meta-models having a well defined structure, which is to be instantiated by the designer for modelling concurrency and performance of micro-architectures, (2) a precise semantics for composing task-level software and transaction-level hardware, and (3) a joint evaluation which enables predicting impact of hardware on software performance and the ability of hardware performance to accommodate other services or applications.

Moreover, the experiments carried out on complex real-life industrial MES show that our simulation prototype is scalable while achieving fast simulation speeds, and delivers correct performance trends. We have successfully used our framework to analyze the performance of several parallel implementations of a MPEG-4 video encoder and a IPv4 packet forwarder on NP architectures such as the Philips' Wasabi/Cake and Intel's IXP2800 [7].

Acknowledgment

This work has been partially supported by projects MEDEA+ NEVA and Minalogic SCEPTRE. The first author has been funded by a joint CNRS/STMicroelectronics research grant.

REFERENCES

[1] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EEdesign, EETimes*, February 2002.
[2] S. Mahadevan, M. Storgaard, J. Madsen, and K. M. Virk, "ARTS: A system-level framework for modeling mpsoc components and analysis of their causality," in *MASCOTS'05*. IEEE, 2005.
[3] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52, 2003.
[4] F. Ghenassia, *Transaction Level Modelling with SystemC. TLM Concepts and Applications for embedded systems*. Springer, 2006.
[5] L. Benini, D. Bertozzi, A. Bogliolo, F. Menichelli, and M. Olivieri, "Mparm: Exploring the multi-processor soc design space with systemc," *J. VLSI Signal Process. Syst.*, vol. 41, no. 2, pp. 169–182, 2005.
[6] I. Assayad, V. Bertin, F.-X. Defaut, P. Gerner, O. Quévreux, and S. Yovine, "JAHUEL: A formal framework for software synthesis," in *ICFEM'05*, LNCS.
[7] I. Assayad and S. Yovine, "P-ware: A precise and scalable component-based simulation tool for embedded multiprocessor industrial applications," *EUROMICRO DSD'07*, IEEE CS.
[8] ——, "Compositional constraints generation for concurrent real time loops with interdependent iterations," in *I2CS'05*, LNCS.