Virtual Platforms in System-on-Chip Design

Katalin Popovici[1] and Ahmed A. Jerraya[2]
[1] *The MathWorks, Inc., Natick, MA, USA*
[2] *CEA-LETI, Grenoble, France*

ARTICLE: Virtual Platforms

Virtual Platforms in System-on-Chip Design

Katalin Popovici[1] and Ahmed Jerraya[2]

[1] *The MathWorks, Inc., Natick, MA, USA*
[2] *CEA-LETI, Grenoble, France*

Abstract— Due to continuously increasing system-on-chip design complexity and tight time-to-market requirements, virtual platforms have become a widely adopted solution to achieve concurrent hardware/software design for embedded architectures. This article explains the advantages of using virtual platforms for software design on multicore/multiprocessor system-on-chip architectures. It also gives a survey of well-known key players in the field: virtual platform tools providers, design companies, and emerging standards for enabling easy intellectual property interoperability. Important research directions required to meet the current challenges of virtual platform design are also summarized.

Index Terms— System-level design, virtual platforms, software design, hardware verification, SystemC, TLM.

I. INTRODUCTION

Before multiprocessor system-on-chip (MPSoC) architectures became so complex, the hardware and software components of an embedded system were designed sequentially. In other words, the software engineers did not begin development of the operating system, device drivers, and inter-processor communication protocol stacks until they had a very solid hardware prototype on which to base their work.

The ever-increasing complexity of MPSoC architectures, along with tight time-to-market requirements, have now made such a scenario entirely unworkable, resulting in missed market windows and revenue opportunities. Thus, software development teams need a means to get an early start on their work, long before the RTL (register-transfer level) of the hardware is finalized.

The International Technology Roadmap for Semiconductors (ITRS) predicts that software development costs will increase, and by 2013 reach rough parity with hardware costs, even with the advent of multicore software development tools [1]. It appears that the growing cost of SoC development is mostly attributable to the embedded software challenge. The 2007 ITRS stated that "software aspects of IC design can now account for 80% or more of embedded systems development cost." Software development and verification methodologies are fairly "ad hoc" and are generally lacking in sophisticated tools.

As research pushes for better programming models for multiprocessor and multicore embedded systems, virtual platforms solve one of today's biggest challenges in these systems: software development, debug and validation before the hardware board is available, enabling concurrent hardware/software design. Virtual platforms model the hardware architecture in the form of a simulator, including processors, memories, communication links and peripherals [3]. They enable engineers to start developing and testing the software substantially earlier than has been possible in the past. Furthermore, they can enable system performance analysis and optimization as well as hardware development and verification.

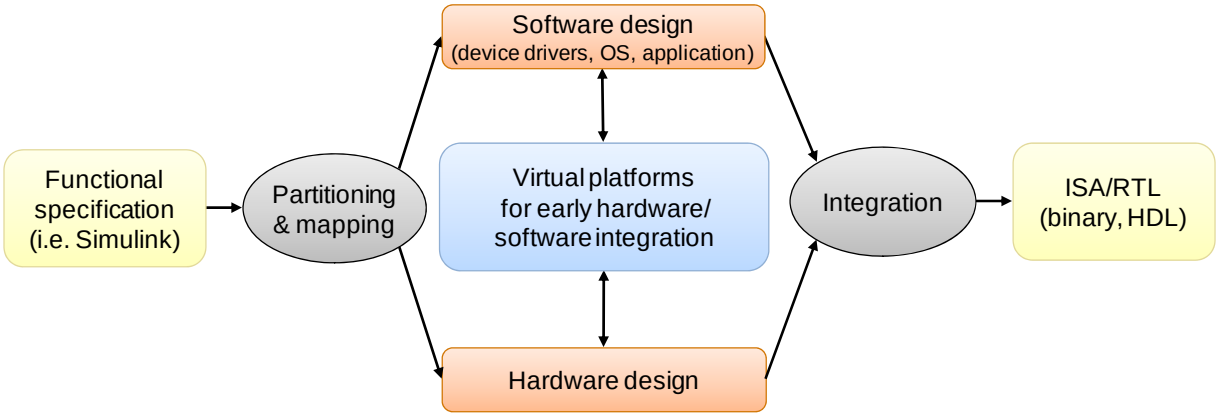

Figure 1: Concurrent hardware/software design.

Figure 1 illustrates a simplified flow of concurrent hardware/software design, where both software and hardware are developed in parallel, and the software design uses virtual platforms. The flow starts with a system-level functional specification. This may be a Simulink functional model simulated using the corresponding environment. Afterwards, the application is partitioned and mapped to either hardware or software target implementations, followed by concurrent hardware and software design. The hardware design produces RTL or gate-level models of the hardware components, usually represented in a hardware description language such as VHDL or Verilog. The software design produces the binary code for the software components. Software development may begin as soon as a virtual platform of the hardware is available, so that the software teams can use the abstracted functional version of the hardware to continually verify their development.

As the hardware development process progresses and the hardware is further refined, updated virtual platforms can be generated and distributed to the software development teams. In this manner, hardware and software development can progress together in lockstep.

The hardware team can verify the design of the hardware components by co-simulating their RTL descriptions in VHDL/Verilog with the virtual platform model of the system. They can also use the virtual platform to generate stimuli and testbenches for verification of the hardware components with an EDA (electronics design automation) simulator.

By performing progressive and early integration of the software with the hardware, various architectural choices can be taken in both software and hardware worlds, so as to optimize the system performance, power consumption and memory footprint. Virtual platforms help the designers to determine tradeoffs in the hardware-software partitioning before the final integration. By analyzing the performance metrics and power consumption, architects can make decisions in terms of type of processing units, choice of memories and size of caches, type of communication network, and peripherals to be included in the chip.

As a result, the software team is able to stay abreast of changes in the hardware development until they have initial silicon prototypes from the foundry. At that point, with software that has been updated throughout the hardware design cycle using the virtual platform, final co-verification of the software and hardware is a relatively easy process. The final integration step consists of verification of the whole system by running the full binary on the hardware prototype.

II. KEY INDUSTRIAL PLAYERS IN VIRTUAL PLATFORMS TECHNOLOGY AND CHALLENGES

A. Emerging standards

Virtual platform technology is the path that overcomes the challenge of early software design by taking advantage of a SystemC-based approach to hardware modeling. SystemC is a standard C++ class library that was devised to provide the concurrency, bit-accuracy and timing required in chip design [4].

Transaction-level modeling (TLM) is an interface modeling methodology used increasingly in complex SoC design. TLM relies on high-speed processor instruction-set simulators (ISSs) and high-level, fully functional SystemC/C++ models of the other hardware building blocks. Standardized TLM modeling rules are critical for ensuring interoperability across platforms, IP (intellectual property) and design tools. OSCI (Open SystemC Initiative) has been working on a SystemC-based TLM standard for some time, and the release of TLM 2.0 in January 2009 addresses various software development needs, as perceived by OSCI. TLM-2 proposes two transaction-level modeling styles: loosely timed (LT) and approximately timed (AT). The loosely timed (LT) modeling style is more suitable for software application development, software performance analysis and software architectural analysis.

The approximately timed (AT) modeling style is closer to timed behavior, for example, in modeling network contention and arbitration. As a result, it is more suitable for hardware architectural analysis and performance verification.

Virtual platforms are usually built by plugging together various TLM models of the hardware components. These models can originate from different IP sources and model providers. To promote IP interoperability, the Spirit consortium proposes the IP-XACT standard to facilitate the exchange among different tools and vendors [5]. The proposal consists of an XML metadata schema and APIs (application programming interfaces) for describing and manipulating the hardware IPs. An example of adoption of this standard is the Socrates chip integration platform, which fully supports IP-XACT for export and import of custom hardware blocks [6].

B. Virtual platform tools providers

The market for virtual platform tools is maturing. A number of EDA vendors, such as Synopsys and Carbon Design Systems, have gone to market with tools that create such virtual platforms, which comprise transaction-level models of the hardware.

With the recent acquisitions of CoWare and VaST, **Synopsys** has become an important provider of diverse tools for virtual platforms and software development, consistent with its advocacy role with respect to TLM-2. Innovator is an integrated virtual platform development environment provided by Synopsys [7]. It supports virtual platform assembly from SystemC TLM-2 hardware models and software development on top of it. The environment includes the DesignWare System Level Library, a huge library of transaction-level models for a rich set of components such as processors, memories, and peripherals, that can be extended by user-defined modules.

Platform Architect is another tool used for virtual platform development, initially developed by **CoWare** and recently acquired by Synopsys [7]. The tool has some similarities with Synopsys Innovator, in the sense that it represents a graphical environment for platform development from existing TLM hardware components provided in a library. But, compared with Innovator, the components are at a low level of abstraction, and thus are more suitable for performance estimation and architecture exploration.

VaST, another recent acquisition of Synopsys, developed the tools CoMET and METeor [7]. CoMET is a system engineering environment which enables system architects to create and analyze platforms. With cycle-accurate modeling, CoMET produces meaningful quantitative results for both timing and power dissipation. Architects can address the optimum balance of speed, power and cost (size). CoMET is used during chip hardware development for co-verification of RTL along with software and other components modeled at the system level. METeor is a software development environment which allows embedded software developers to create code using a virtual system prototype that runs at near real-time speeds on an off-the-shelf PC (personal computer). METeor thus forms a pure software implementation of a development board and its in-circuit emulator.

OVP (Open Virtual Platform) developed by **Imperas**, provides ultra-fast, instruction-accurate virtual platform models [11]. OVP is made up of three main components: APIs that enable modeling in C of a hardware component, a collection of free open-source processor and peripheral models, and the OVPsim simulator which executes these models.

Mentor already has a solid market position with Catapult C; this position has been enhanced with SystemC support [8].

Carbon Design Systems provides solutions to build cycle-accurate IP models for virtual platforms [9]. The generation of the IP models consists of compiling the IP's RTL implementation into a high-speed software model. This illustrates a bottom-up approach, which starts from RTL and goes to a higher level of abstraction as employed by virtual platforms.

The other approach for IP generation is top-down, as proposed by **The MathWorks** [10]. The EDA Simulator Link tool from The MathWorks provides glue to link the design of high-level algorithms with existing virtual platforms. It automatically generates SystemC components from high-level applications modeled in Simulink. The generated SystemC module has a TLM-2 standard interface, so that it can be incorporated into a virtual platform which supports such an import. The SystemC generation also includes testbenches, so that the IP designer can verify the behavior of the generated TLM component with respect to the modeled functionality in Simulink.

CoFluent Studio is another tool which allows generation of SystemC transactional models from high-level UML (Unified Modeling Language) descriptions or DSL (Domain-Specific Language) descriptions [21]. Platforms are built by assembling generic models of universal components, like processors, integrated circuits, memories, busses, and interfaces. Each generic model provides variable parameters to easily adjust its behavior and performance characteristics. No instruction set simulators are used.

C. System-on-Chip design companies

Besides tool providers, System-on-Chip design companies play an important role in determining the direction and usage of virtual platforms. Most of them have developed their own virtual platform modeling environments, or they provide models of different IPs with export capabilities to the previously described virtual platform tools.

ARM provides a set of fast models of various ARM processors and peripherals, which can be used to create a virtual platform for a custom multi-core chip [23]. The virtual platform design is available through a block diagram editor, called System Canvas, which has export capability to SystemC TLM-2. The models are instruction accurate and run at speeds comparable with the actual hardware boards.

Tensilica proposes a system-level modeling environment of multiprocessor architectures with XTENSA processors, called XTMP [26]. This includes models of memories, connectors, and queues that can be interconnected with configured XTENSA processors into an overall system model that runs either as a SystemC model or as a C/C++ model. The processor and device interfaces are at transaction level.

An industrial case study of virtual platform utilization for the development of a new hard disk system is described by **Samsung** in [12]. According to the authors, the virtual prototype allowed significant optimization of the software code before the real board was available, resulting in a 50% improvement in the system performance.

Most mobile platform providers release software development kits (SDKs) which include a software simulator of the hardware. For instance, **Apple** released an SDK for iPhone applications [24]. **Qualcomm** provides an SDK application development environment for the Brew mobile platform [22]. This SDK includes an integrated hardware simulator and a set of programming APIs to access the different devices of the platform. **Samsung** has made available a mobile widget SDK for web developers to build and emulate their applications with the Samsung widgets [25]. Software developers can test their applications by remotely accessing Samsung's Virtual Device Lab on current Samsung phones.

ST Microelectronics is a pioneer of SystemC and contributor to the TLM-2 standard from the beginning [27]. They use SystemC models of hardware components extensively for both software design and hardware verification, i.e. network-on-chip SystemC models.

Freescale has developed virtual models to decouple the software and hardware design cycles for automotive applications [28]. The models simulate the Power Architecture e200 processor cores and peripherals available for the MPC55xx and MPC56xx processor families.

The authors of [29] from **Infineon Technologies** present a case study of application development using a virtual platform. The authors implement the receiver part of the DVB-T/H digital multimedia broadcasting system on a SDR (software-defined radio) platform called MuSIC (multiple SIMD Cores). They used a cycle accurate SystemC model of the architecture for the application development and performance analysis.

III. RESEARCH DIRECTIONS AND PERSPECTIVES FOR VIRTUAL PLATFORMS

Current literature includes several academic research directions involving virtual platforms. The concept of virtual platform appears in most work regarding software development and code optimization before a hardware board is available [12]. Other research directions use virtual prototype simulation to perform software profiling, e.g., execution cycles required by a given application and software performance analysis [13].

A. Techniques of simulation speedup for virtual platforms

Most existing virtual platforms use instruction set simulators for the software execution. This implies high accuracy and low simulation speed. Moreover, simulation time increases exponentially with the number of processor cores integrated on the chip and, thus, required to be part of the virtual platform. As a result, finding new methodologies that speed up simulation but still maintain accurate performance evaluation represents an important aspect and focus of many ongoing research projects.

Because execution of software on a virtual platform using ISS still suffers from low simulation speed compared to hardware, many researchers focus on developing new techniques to attain high simulation speed. In this context, [14] mixes interpreted ISS simulation with compiled ISS simulation to allow a multiprocessing simulation approach that increases performance.

The authors of [15] describe an ultra-fast ARM and multi-core DSP instruction set simulation environment based on just-in-time (JIT) translation technology, which refers to dynamic translation of target instructions (ARM, DSP) to host machine instructions (x86) during the execution.

The authors of [16] present a fast, hybrid simulation framework which allows switching between native code execution and ISS-based simulation. In this approach, the platform-independent parts of the software stack are executed directly on the host machine, while the platform-dependent code executes upon an ISS. Thus, the framework allows debugging a complex application by executing it natively until the point where the bug is expected, and then executing on the ISS to examine the detailed software behavior and execution trace stack.

Other research groups focus on integrating ISS within existing design flows, like Ptolemy which is a design environment that leverages time-approximate cosimulation based on source code estimation of execution time, and refines its precision using an ISS.

The work presented in [19] proposes cycle-accurate TLM platforms which use native software execution and timing annotation for accurate performance estimation. The proposed platforms allow fast simulations because of the native execution of the software, while still capturing low-level hardware details such as shared resources between processors (i.e., memory mapping, caches, or synchronization mechanisms).

B. Automatic generation of virtual platform models

The virtual platform has to be available much earlier than the real hardware, in order to allow concurrent software and hardware design. One of the solutions which address this challenge is

automatic generation of the virtual platform models. This permits shortening of design time and reduction of human coding and modeling errors.

The authors of [17] present automatic generation of virtual execution platforms for the hardware architecture, to analyze the run-time behavior of an application running on a real-time operating system, and to accurately estimate performance data.

[18] proposes automatic software TLM platforms synthesis. Automatic generation of HdS (hardware-dependent software) is supported, including application code generation, communication software synthesis, multi-task synthesis and generation of the configuration and makefiles to control the cross-compilation and linking of the generated code for a particular processor.

C. Virtual platforms at different abstraction levels

Other research works focus on raising the level of abstraction used by the virtual platforms to higher-level software development platforms [2]. This is a natural direction given the layered structure of the software stack. The software running on chips is often represented as a stack composed of three main layers: (i) application, usually expressed as a process network or multithreaded application description; (ii) OS (operating system) or middleware which provides services to the application layer, such as threads/tasks scheduling, inter-thread communication and synchronization; and (iii) HAL (hardware abstraction layer) which permits access to and configuration of the hardware resources, such as peripherals. These layers are shown in Figure 2.

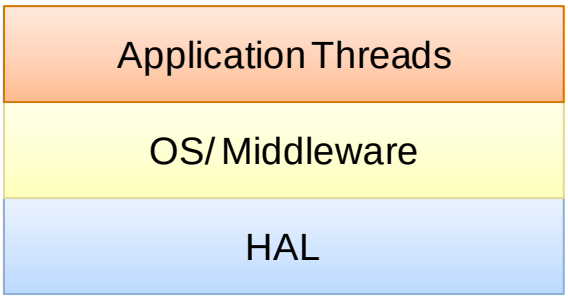

Figure 2: Software stack.

Each of these software layers corresponds to a different level of abstraction, and they have different requirements in terms of design and debug capabilities of the underlying hardware models. Adopting a multi-layer programming approach facilitates incremental software design and validation. This consists of using software development platforms with different hardware details depending on the software component: high-level platforms for the application development, transaction-accurate platforms for OS and scheduling, and low-level virtual platforms for the full software stack validation.

Furthermore, the multi-layer approach makes the software debug an iterative process, because the different components need different details in order to be validated. For example, the debug of the application code running on different processors does not need accurate implementation of the synchronization protocol between the processors, i.e., detailed models of mailboxes in the development platform. In the same time, the debug of the threads scheduled by an OS requires this kind of detail. All these requirements are considered during the abstraction of the architecture in the platform used at each design step. Thus, depending on the software component to be designed and

validated (application threads code, threads scheduling using an OS, HAL integration in the software stack) the development platform may model only the minimal subset of hardware components required by the software validation. The rest of the hardware components, which are not relevant for a specific software component, are abstracted.

In practice, not all of the various development platforms are used by all the design teams, although many use one or two. The high-level platform is useful for algorithms developers, who implement new algorithms or optimize existing ones; it can also serve as a functional specification of the system architecture and design requirements. The transaction-accurate platform models can be used by system architects – who mostly determine the hardware-software partitioning but do not require accurate results yet for performance estimation – or by embedded software engineers who integrate the application threads with the OS, by implementing inter-thread communication, scheduling and synchronization. The virtual platform is useful for device driver development and architecture exploration, as well as for verification of RTL design by hardware designers.

REFERENCES

[1] *International Technology Roadmap for Semiconductors*, Software cost estimation for System-on-Chip, 2007.
[2] K. Popovici, F. Rousseau, A. Jerraya and M. Wolf, *Embedded Software Design and Programming of Multiprocessor System-on-Chip: Simulink and SystemC Case Studies,* Springer, 2010.
[3] B. Bailey and G. Martin, *ESL Models and Their Application: Electronic System Level Design and Verification in Practice (Embedded Systems)*, Springer, 2009.
[4] Open SystemC Initiative, *TLM-2.0 User Manual*, http://www.systemc.org/downloads/standards.
[5] Spirit, *IP-XACT*, http://www.spiritconsortium.com.
[6] Duolog Technologies, *Socrates chip integration platform*, http://www.duolog.com.
[7] Synopsys, *Innovator*, *Platform Architect, CoMET, METeor,* http://www.synopsys,com
[8] Mentor Graphics, http://www.mentor.com
[9] Carbon Design Systems, http://carbondesignsystems.com
[10] The MathWorks, *EDA Simulator Link*, *Simulink,* http://www.mathworks .com
[11] OVP, http://www.ovpworld.org/
[12] S. Hong, S. Yoo, S. Lee, H. J. Nam, B. S. Yoo, J. Hwang, D. Song, J. Kim, J. Kim, H. S. Jin, K. M. Choi, J. T. Kong and S. Eo, “Creation and Utilization of a Virtual Platform for Embedded Software Optimization: An Industrial Case Study”, *Proc. IEEE/ACM Intl. Conf. on Hardware/Software Codesign and System Synthesis*, 2006, pp. 235-240.
[13]. M. Oyamada, F.R. Wagner, M. Bonaciu, W. Cesario and A. Jerraya, “Software Performance Estimation in MPSoC Design”, *Proc. ASP-DAC*, 2007, pp. 38-43.
[14] W. Qin, J. D’Errico and X. Zhu, “A Multiprocessing Approach to Accelerate Retargetable and Portable Dynamic-Compiled Instruction-Set Simulation”, *Proc. IEEE//ACM Intl. Conf. on Hardware/Software Codesign and System Synthesis*, 2006, pp. 193-198.
[15] S. Singhai, M.Y. Ko, S. Jinturkar, M. Moudgill and J. Glossner, “An Integrated ARM and Multi-Core DSP Simulator”, *Proc. CASES*, 2007, pp. 33-37.
[16] J. Ceng, W. Sheng, J. Castrillon, A. Stulova, R. Leupers, G. Ascheid and H. Meyr, ”A High-Level Virtual Platform for Early MPSoC Software Development”, *Proc. IEEE/ACM Intl. Conf. on Hardware/Software Codesign and System Synthesis*, 2009, pp. 11-20.
[17] S. Park, W. Olds, K.G. Shin and S. Wang, “Integrating Virtual Execution Platform for Accurate Analysis in Distributed Real-Time Control System Development”, *Proc. Real-Time Systems Symposium*, 2007.
[18] D. D. Gajski, S. Abdi, A. Gertslauer and G. Schirner, *Embedded System Design: Modeling, Synthesis and Verification*, Springer, 2009.
[19] P. Gerin, M.M. Hamayun and F. Petrot, “Native MPSoC Co-Simulation Environment for Software Performance Estimation”, *Proc. IEEE/ACM Intl. Conf. on Hardware/Software Codesign and System Synthesis*, 2009, pp. 11-20.
[21] CoFluent Design, http://www.cofluentdesign.com
[22] Qualcomm, http://www.qualcomm.com

[23] ARM Ltd, http://www.arm.com
[24] Apple, http://www.apple.com
[25] Samsung, http://www.samsung.com
[26] Tensilica, http://www.tensilica.com
[27] F. Ghenassia, *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*, Springer, 2005.
[28] Freescale, http://www.freescale.com
[29] Y. Jiang, W. Xu, and C. Grassmann, “Implementing a DVB-T/H Receiver on a Software-Defined Radio Platform”, *Intl. J. of Digital Multimedia Broadcasting,* February 2009, Article ID 937848.