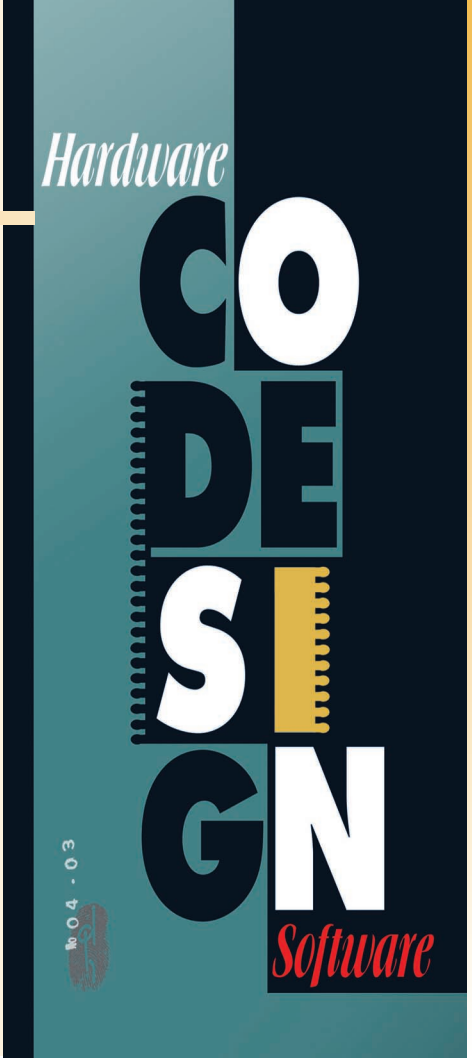
Taking on the Embedded System Design Challenge

A new generation of synthesis tools has provided the primary driving force behind the integration of hardware and software components during system design and development.



Jörg Henkel
NEC Laboratories America

Xiaobo Sharon Hu
University of Notre Dame

Shuvra S. Bhattacharyya
University of Maryland

Although hardware/software codesign is an old art that has challenged engineers since hardware first existed and some software ran on that hardware, major research efforts in the field were first undertaken in the early 1990s. The emergence of a new generation of synthesis tools for hardware and software provided the primary driving force behind these efforts.

EVOLUTION OF CODESIGN

The term *hardware/software codesign* refers to the methodology, tools, and practices that support the integration of hardware and software components during system design and development.

Hardware/software codesign originally focused on computer-aided design techniques, such as specification and modeling, partitioning, performance estimation, interface design, cosimulation, and coverification. However, the discipline has far exceeded its original roots, and it currently has an impact on most aspects of embedded system design.

This integrated design approach has brought new thinking to process scheduling, communication protocols, code generation, and software development environments. It has introduced new approaches to the design of application-specific processors and reconfigurable and customizable architectures.

The complexity of today's embedded systems and the trend of fabricating an entire embedded system—or the majority of it—on a single chip have doomed the custom of separating hardware and software in the early design phase. Hardware/software codesign's philosophy and techniques have permeated numerous application domains, such as wireless and telecommunications, automotive and other consumer electronics, and computer security.

In addition to having become a well-recognized area in the field of electronic design automation, hardware/software codesign also has experienced a certain level of commercial success, with several emerging companies focusing on various aspects of the design paradigm.

Technology advances and increasingly sophisticated embedded applications are propelling an ever-increasing demand for chip capacity that will present greater challenges and opportunities for hardware/software codesign research.

As Daya Nadamuni and Gary Smith describe in the "Electronic System-Level Design: Challenges on the Road Ahead" sidebar, hardware/software codesign is a reality today and a necessity for tomorrow.

IN THIS ISSUE

The articles in this special issue review the progress made in the past decade and introduce cutting-edge research results in hardware/software codesign.

In "A Decade of Hardware/Software Codesign,"

0018-9162/03/$17.00 © 2003 IEEE

Electronic System-Level Design: Challenges on the Road Ahead

Daya Nadamuni and Gary Smith, Gartner Dataquest

The idea of creating a virtual product with all the associated subsystems and then testing and verifying that product before manufacture is attractive from both the cost and time-to-market perspectives.

The International Technology Roadmap for Semiconductors, 2001 Edition offered an example to illustrate the cost advantages of moving to electronic system-level design. In 2001, the cost of designing an 8-million-gate personal digital assistant using register-transfer-level tools as well as reusing blocks was US$15 million. In 2005, the cost of designing the same 8-million-gate PDA will be US$8 million, assuming that electronic system-level tools and methodologies are used. Because the PDA is evolving beyond its current configuration, PDA design will be far more complex in 2005. Consequently, moving to electronic system-level design will be a necessity to contain costs and meet time-to-market demands.

Today's biggest challenge is to create a unified path toward true hardware/software codesign. The move to electronic system-level design will be the next major breakthrough in electronic design automation. Gartner Dataquest has identified three major areas for using electronic system-level design: hardware/software codesign and cosimulation, behavioral synthesis tools, and test and verification tools.

Clearly, developers must overcome several challenges before they can move to electronic system-level design. While many of these are technology challenges, some significant marketing challenges also exist. Because good technology does not sell itself, equal consideration must be given to overcoming these marketing challenges.

System definitions

A principal challenge lies in overcoming the confusion caused by different definitions of a system. A system-on-chip design provides one definition of a system. An embedded board, perhaps preloaded with a real-time operating system and some device driver software, is also a system.

The definition of the system is really based on the product being developed and the system designer's perspective. From a marketing perspective, rather than assuming the system's value can be found only in SoC technology, tool vendors should approach the challenges of hardware/software codesign from the user's perspective.

Design styles

The design style for a particular user base is derived from the user's competitive advantage. Before electronic system-level design-tool vendors enter the market, they need to consider whether they can support the particular design styles of their targeted user base.

SoC technology has an ASIC focus, and therefore primarily a silicon design focus. In embedded design, the competitive advantage resides in the software.

A component-based system glued together using multiple off-the-shelf parts also falls within the scope of our definition of a system. Such systems may have a different competitive focus, depending on the particular industry in which they are used. For some companies, like toy manufacturers, it could be a seasonal focus. For others, the advantage could lie in a highly specialized design that is low in volume but has high margins.

SoC design users typically are leading-edge companies that have access to internal CAD teams, require extensive customer support, and have internal tools that must be integrated with the commercial solution. They will pay a high price per seat, but support costs can be high. Component-based designers, on the other hand, are unwilling to pay a high price per seat, but can be counted on for high volume because they comprise a much larger

Wayne Wolf reviews the past, present, and future of this dynamic field. Wolf succinctly summarizes significant research results achieved in the past 10 years and projects some challenging problems to be solved in the next decade.

The ultimate goal of hardware/software codesign is to provide a rigorous yet flexible environment in which users can develop their own systems. However, the myriad available computation models, tools, and design flows make creating such an environment a great challenge. In "Metropolis: An Integrated Electronic System Design Environment," Felice Balarin and his colleagues propose a novel metamodel that has precise semantics and can support various computation models as well as different simulation, analysis, and synthesis tools. Based on this model, the authors have developed the Metropolis environment, a unified framework for designing complex embedded systems.

Hardware/software cosimulation is an indispensable process in any system development. The ease of use and performance of cosimulation tools have been a major concern ever since the onset of hardware/software codesign research. It has been well accepted that cosimulation in an environment based on C++ is desirable. How to make such an environment user friendly and efficient has been under constant investigation.

In "SystemC Cosimulation and Emulation of Multiprocessor SoC Designs," Luca Benini and his colleagues present a generic C++ cosimulation framework based on using the standard GDB remote debugging interface between the instruction set simulator and the wrapper used to link it to the simulation environment. Their effort provides an environment that enhances simulation tool performance.

Many real-time, safety-critical systems require formal verification to guarantee the system's behavior both functionally and timewise. The presence of multiple heterogeneous processing elements in a single system makes verification a daunting task. In "A Formal Approach to MpSoC Performance Verification," Kai Richter and his colleagues illustrate through practical examples the importance of formal verification. They propose using event model inter-

community than the power users.

Similarly, developers must ask several questions while researching the product they plan to design. The design style can determine how successful they will be because the project's cost can increase dramatically depending on the design style. A SoC design is the most expensive of all three styles.

Platform-based design

The original definition of a platform was a frozen architecture. Once developers freeze the architecture, engineers can standardize the interfaces and make some choices among the building blocks. These choices range from making modifications using software to using multiple blocks.

Choices for a specific platform are limited in the beginning, but increase as time goes by. Developers work from the assumption that all the blocks and interfaces have been verified before being released. While flexibility is good, this is easier said than done.

Architecture changes disrupt the interfaces. This means that keeping the platform viable requires a major ongoing effort in verification and setting up new standards.

One platform-based design technique, *derivative design*, has been used by the cell phone industry for some time. Instead of introducing new cell phone designs multiple times a year into the same market, developers introduce derivative designs into different market segments.

We can now do derivative system-on-chip design with a good architecture and a strict hierarchical design methodology by replacing cores. This approach resembles replacing integrated circuits in a PC board design. Verification is still critical in this scenario.

Platform-based design provides a good fit for embedded software designs but is only a temporary solution for SoC designs. All silicon design breakthroughs happen at the architectural level, so freezing the architecture as it is in platform-based design severely limits the ability to use silicon as a competitive advantage.

The market

In terms of software revenue, analysts estimated that the EDA and embedded software development tools market reached US$3.5 billion in 2001. Of this, the electronic system-level tools market accounted for only US$85.9 million in revenue. However, Gartner Dataquest estimates that this market will grow to US$327 million in 2006.

Electronic system-level design has gone through two incarnations so far. The appearance of domain-specific tools characterized the first generation. A less than entirely successful attempt to merge the data path and control logic domains characterized the second generation.

The next generation of electronic system-level tools must be something more than an extension of existing register-transfer-level tools. The biggest challenge will be to resolve the current incompatibility of the hardware and software design methodologies. The vendors and users who successfully address this challenge will be the winners in their markets. ■

***Daya Nadamuni** is a principal analyst in Gartner Dataquest. She received an MS in economics from Cambridge University. Contact her at daya.nadamuni@gartner.com.*

***Gary Smith** is a chief analyst for the Electronic Design Automation Worldwide program in Gartner Dataquest's Technical Software group. He received a BS in engineering from the United States Naval Academy at Annapolis, Maryland. Contact him at gary.smith@gartner.com.*

facing and adaptation techniques to support existing approaches to formal performance verification.

New applications bring new challenges to hardware/software codesign. In "Domain-Specific Codesign for Embedded Security," Patrick Schaumont and Ingrid Verbauwhede discuss how hardware/software codesign concepts help in adding security support to various embedded and networked platforms and how such efforts present new opportunities for enriching the hardware/software codesign area.

We thank all the authors who submitted papers for this special issue. Unfortunately, we could not include many excellent articles due to the page limit and the consideration of content balance for this issue. In addition, we also thank the reviewers for their efforts.

For readers who want to be on the edge of future developments in hardware/software codesign and system-level design, we recommend a new forum, tentatively named CODES/ISSS, which will be offered in October 2003 (www.ece.uci.edu/~codes/). This conference represents the merger of the former CODES and ISSS, reflecting the widened scope of these two major international symposia on hardware/software codesign and system synthesis. ■

***Jörg Henkel** is a senior research staff member at NEC Laboratories America, Princeton, N.J. He received a PhD in electrical engineering from the Technical University of Braunschweig. Contact him at henkel@nec-labs.com.*

***Xiaobo Sharon Hu** is an associate professor in the Department of Computer Science and Engineering at University of Notre Dame. She received a PhD in electrical engineering from Purdue University. Contact her at shu@cse.nd.edu.*

***Shuvra S. Bhattacharyya** is an associate professor in the Department of Electrical and Computer Engineering and the Institute for Advanced Computer Studies at the University of Maryland. Bhattacharyya received a PhD in electrical engineering and computer sciences from the University of California, Berkeley. Contact him at ssb@eng.umd.edu.*