What makes high-performance embedded computing so difficult is not just the performance levels but achieving that performance in the face of two major requirements: real time and low power.

Embedded computing is all too often associated with simple controllers in low-end products. While important—billions of 8-bit microcontrollers are sold every year—small microcontrollers certainly aren’t the whole story. High-performance microprocessors are used singly and in networks in all sorts of applications.

What do we mean by high performance? Generally speaking, giga-ops of computing power. For example, next-generation cell phones that use software-radio techniques will need to deliver at least 10 billion operations per second. Multimedia applications similarly require huge performance levels, and they require that performance continuously, not just in bursts.

However, what makes high-performance embedded computing so difficult is not just the performance levels but achieving that performance in the face of two major requirements: real time and low power.

Real-time computing doesn’t just mean that you have to buy a really expensive PC to make the application run fast enough—it means meeting deadlines. Traditional desktop applications like word processing don’t have to meet hard deadlines; they can slow down when they perform some complex tasks or if other jobs are running on the same machine. Automobiles, digital still cameras, and other embedded systems must finish certain tasks in a fixed amount of time. In the case of a camera, the picture may be ruined if the embedded system doesn’t meet its deadlines; in the case of cars, a catastrophic accident could be the result of a missed deadline.

Low-power computing is of interest everywhere, but it is particularly challenging for battery-operated embedded devices. Cell phones, for example, must operate for hours on subwatt batteries. These devices must deliver server-level performance at two to three orders of magnitude less power consumption. Traditional computing...
platforms are several orders of magnitude too inefficient to be useful in these applications.

Embedded computing is critical to many products that are mainstays of modern society. Automotive costs are now approaching the proportions long held by aircraft: the cost of an automobile is now about one-third frame, one-third power plant, and one-third electronics. Automobiles couldn’t meet the fuel efficiency and pollution targets set by law without sophisticated embedded controllers. First-generation cell phones used analog transmission with simple microcontrollers to handle basic tasks, but modern cell phones rely on microprocessors for core tasks like voice compression and baseband signal processing, not to mention additional features like phone books and multimedia.

The tough challenges imposed by applications like automotive, multimedia, and cellular telephony mean that embedded computing needs to introduce new hardware and software concepts—tweaks on traditional techniques aren’t enough.

First, we need new computer architectures—and almost certainly more than one. The desktop architecture isn’t the answer to all of the world’s problems—desktop computing is itself an application with a particular set of goals and requirements.

High-performance embedded platforms used to be derived from desktop processors. However, given the large volumes at which some embedded platforms, such as cell phones, are produced, we might start to see designs flow the other way, with desktop and other systems based on cell phone platforms.

Multiprocessor systems-on-chips have emerged as powerful platforms for embedded systems—see Computer’s July 2005 issue for several articles on MPSoCs. MPSoCs combine programmable processors, specialized function units, and on-chip memory to create true single-chip multiprocessors. MPSoCs predate chip multiprocessors and multicore processors by five years or more; these chips provide specialized features that allow them to meet the stringent low-power and real-time requirements of high-end embedded systems.

We also need new software architectures. We still have all the traditional software engineering problems in embedded computing systems, including software architecture, maintainability, and so on. Embedded systems often have astonishing amounts of code. For example, ST Microelectronics estimates that supporting all of the major audio standards requires 1 to 2 million lines of code. But embedded software must meet other requirements as well. Real-time software requires accurate performance and power estimation at several levels of abstraction so that we can refine software designs that satisfy timing and power properties as well as functionality.

IN THIS ISSUE

The first feature article in this special issue outlines the educational, industrial, and policy changes that will enable manufacturers and, more importantly, the public to capitalize on the huge potential that emerging hardware and communications technologies offer. The other three articles describe state-of-the-art technologies in major embedded systems—automotive, consumer, and cellular telephony—that shape our daily lives.

“The Discipline of Embedded Systems Design” by Thomas A. Henzinger and Joseph Sifakis points out the gap between our current understanding of computer science and the requirements of embedded systems. Embedded systems design is peripheral to both computer science and electrical engineering curricula. Much of the cultural wall between the two fields can be traced to differences between the discrete mathematics of computer science and the continuous mathematics of traditional engineering. The authors argue that industry desperately needs engineers who feel equally at home in both worlds and that embedded systems design has the potential to produce such integrated talent. Defining its scientific foundation, however, will take a concerted, coordinated effort on the part of research, academia, industry, and policy makers.

In “Embedded System Design for Automotive Applications,” Alberto Sangiovanni-Vincentelli and Marco Di Natale delve into automotive electronics. As each car receives an ever-increasing number of electronic control units, software complexity escalates to the point that current development processes and tools can no longer ensure sufficiently reliable systems at affordable cost. Novel system-level analysis, modeling methods, and tools are needed not only for predictability and composability, but to support the designer in the critical architecture-evaluation process that profoundly affects a vehicle’s cost, performance, and quality. To make automotive electronics systems a relatively inexpensive plug-and-play subsystem to which OEMs can add novel and compelling functionality, the essential technical problem for this vision is the establishment of standards for interoperability among IPs—both software and hardware—and tools. AUTOSAR, a worldwide consortium of most of the players in the automotive electronics supply chain, has this goal clearly in mind.

“Using DaVinci Technology for Digital Video Devices” by Deepu Talla and Jeremiah Golston describes processors for digital media. Developing innovative and cost-efficient digital video products requires systems that encompass open and flexible system-on-chip processors, software, and development tools. For the past decade, SoC technology has been the primary answer for the DV
market. However, increasingly complex DV applications require design flexibility for greater customization and advanced feature updates. It’s not practical for one SoC to fit all equipment and meet today’s aggressive performance, power, and cost targets. The DaVinci integrated processor portfolio offers software, tools, and support for developing a broad spectrum of optimized DV equipment.

Finally, in “Software-Defined Radio Prospects for Multistandard Mobile Phones,” Ulrich Ramacher reviews next-generation processors for cell phones. Multiple standards have become the norm in the high-end mobile phone market. This multitude of existing and evolving standards, coupled with the need to react quickly to market requirements, is radically changing the system architect’s task. Software-defined radios could well provide multistandard solutions on the RF baseband and transceiver side, but they also represent a new stage in complexity. Semiconductor vendors, software houses, and major handset manufacturers must cooperate closely to support the various use cases, with the required levels of flexibility, while simultaneously achieving high enough volumes to justify the return on the huge investment needed.

Wayne Wolf is the Rhesa “Ray” S. Farmer Jr. Distinguished Chair in Embedded Computer Systems and GRA Eminent Scholar at the Georgia Institute of Technology’s School of Electrical and Computer Engineering. His research interests include embedded computing, multimedia systems, and VLSI CAD. Wolf received a PhD in electrical engineering from Stanford University. He is a member of the IEEE Computer Society and serves as an area editor on Computer’s editorial board. Contact him at wayne.wolf@ece.gatech.edu.