The term *hardware-software codesign* has generated a great deal of interest in the past year. The articles in this issue show that the term covers a lot of ground, ranging from system-level performance evaluation to hardware-software partitioning. All these articles focus on a common concern: How do we use computers to build systems that must satisfy goals and constraints?

In the first article, Thomas, Adams, and Schmit formulate the cosynthesis problem as how to meet performance goals with a minimum of hardware resources. A key feature of their method is that hardware-software partitioning takes place at the task level. Kalavade and Lee combine two heterogeneous methodologies within a single framework that allows the designer to integrate hardware and software components through the synthesis, simulation, and evaluation phases. Gupta and De Micheli describe a hardware-software partitioning algorithm that minimizes hardware costs while satisfying real-time constraints.

Salinas, Johnson, and Aylor present a methodology using VHDL to model computer architectures independent of implementation attributes. The methodology can serve as a component of a unified design environment supporting hardware-software codesign. Giving us a look at the practical application of hardware-software codesign, Smialagic and Siewiorek describe the concurrent hardware and software development of the wearable computer called VuMan 2.

This collection's two remaining articles will appear in *IEEE D&T's* December issue. In “Hardware-Software Cosynthesis for Microcontrollers,” Ernst, Henkel, and Benner present a software-oriented, iterative partitioning process based on “hardware extraction” controlled by a cost function. In “Design Issues in Parallel Simulation Languages,” Rajaei and Ayani discuss parallel simulation techniques for system performance evaluation.

Hardware-software codesign isn’t a new field—engineers have been doing it for years. What is new is an interest in systematic, scientific codesign methods. Many consumer products now incorporate 32-bit processors with caches executing megabytes of code. Such systems are too complex to be designed by seat-of-the-pants techniques. Almost every design lab has horror stories about embedded systems built with ad hoc methods that are late to market and don’t meet their performance goals. And in safety-critical applications such as avionics and automotive electronics, the specter of fatal errors gives even more impetus to the drive for systematic design methods.

Embedded computer system design is in a state very similar to VLSI design in the late 1970s. Although integrated-circuit designers had been building chips for years, they realized that advances in fabrication technology were outrunning their ability to make use of the available chip area. Mead and Conway’s *Introduction to VLSI Systems* was a clarion call to develop a fresh approach to chip design. Today, design groups around the world are discovering that the computing power available in microprocessors is outriving their ability to design hardware-software systems by hand. The next 10 years will be as revolutionary for the design of microprocessor-based systems as the last 10 years have been for IC design.

Address correspondence about this special issue to Wayne Wolf, Princeton University, Dept. of Electrical Engineering, Princeton, NJ 08544; or wolf@princeton.edu.