PANEL: HARDWARE/SOFTWARE CO-DESIGN: THE NEXT EMBEDDED SYSTEM DESIGN CHALLENGE


With the proliferation of consumer electronics, the number of embedded systems is growing dramatically, according to Collett International Inc.'s research. At the same time, embedded systems are growing in size and complexity. Another major trend is to increasingly implement functionality in software. Design teams are using software to differentiate products, increase flexibility, respond to changing standards, enable inexpensive upgradability, and get products to market sooner. This confluence of forces is causing design teams to confront a host of new challenges and to even re-evaluate their fundamental design practices. The challenges of designing hardware and software in concert are now coming to the forefront. Some of the topics to be discussed include: Short of producing synthesizable output, can hardware/software co-design provide meaningful value? Can automatic hardware and software partitioning tools replace engineering judgement? Who is responsible for hardware/software co-design? System architects? Hardware designers? Software designers? Can today's tools truly optimize system functionality and performance tradeoffs between hardware and software implementation? Are commercial real-time operating systems impacting hardware and software tradeoff decisions?

James A. Rowson, Alta Group of Cadence Design Systems, Sunnyvale, CA

The consumer-driven, low-cost, fast time-to-market mentality that pervades electronics design these days has begun to force many companies to rethink their methodology for product development. A new consensus on design methodology is beginning to emerge where the functionality of the product is described separately from the architecture of the product. Many alternative mappings of function onto architecture must be explored quickly in order to get a cost-effective implementation. This function-architecture co-design will utilize reuse to sharply improve design time. Synthesis will be used to glue large IP together. The central task in function-architecture co-design will actually be the design exploration, performance analysis phase where the feature, cost, performance, and power will all be defined.

Guido Arnout, CoWare, Inc., Santa Clara, CA

Whether you call it an embedded system, a system-on-a-chip, a tightly integrated consumer device or simply a system, one thing is becoming clear to chip designers: systems, large or small, are implemented as a combination of software and hardware. We all know what it takes to develop software and hardware independently of each other. So it appears that hardware/software co-verification is essential in those cases where hardware and software must be designed closely together. But, what exactly are we co-verifying? Co-verification implies that we first have fully implemented the part of the system that will be executed as software running on a processor and what will be implemented as dedicated hardware. But, how do we get to the right partitioning? Don't we first have to define what the system is supposed to be doing before we dabble in hardware/software partitioning, and its performance tradeoffs, before we try to synthesize anything, before we co-verify the result of all this work? The key to co-design is to first describe and fully debug the complete system functionality as an architecture-independent executable specification that can be re-used throughout the implementation process. Only then do partitioning, synthesis and detailed co-verification really come into play.

Fred Rose, Honeywell, Inc., Minneapolis, MN

Much of the tool development work for embedded hardware/software co-design focuses on reasonably small, high-volume systems (such as consumer applications). In larger embedded systems (such as avionics), many factors affect the choice of partitioning functions across a hardware and software architecture. These factors include fault tolerance, redundancy, reliability, life cycle costs, legacy hardware and software issues, and new capabilities such as configurable computing and multiple domains such as optical and MEMS devices. System designers for these systems need flexible co-design tools which can interface with domain-specific notations and provide robust analytic capability. The co-design tools must support a spiral model of development, which means multiple views of the design and multiple layers of abstraction. Co-design tools are needed for design space exploration as well as for validation and verification.

Vess L. Johnson, Omniview Design, Inc., Pittsburgh, PA

As system and chip designs become more complex, the ability to evaluate different architectures to optimize performance has suffered. What is required is a way to evaluate hardware/software tradeoffs at the architectural stage of the design process. The impact of multiple processors, types of processors, memory sizing, bus structure, etc. as well as hardware/software partitioning must be examined very early in the design process. Hardware and software subsystems need to be modeled separately and the allocation of functions should be performed between the hardware and software components in an optimal manner.
Such performance optimization at the early architectural stage of the design process will result in major reductions of expensive design iterations later in the design cycle, yielding dramatic savings in cost and time to market.

Mark Medovich, Sun Microsystems, Inc., Palo Alto, CA

Embedded system design has been a testimonial to the existence of engineering disorders. Engineers often complain about a host of ailments, among which are "reinvention of the wheel" syndrome, and its cousin, "not invented here" syndrome. There are almost no standard design methodologies in place which provide an easy migration from one generation of an embedded system design to its next incarnation. The embedded system designer and design team manager have long since realized that embedded system design is too low level for complex designs which need to be deployed and maintained over a long term product life cycle. Design and development environments for individual components within systems, be they hardware or software, are often non-homogenous. That is, if multiple processors, ASICs, and FPGAs are implemented on a system board which is networked or connected in some way with different system boards, the associated development environments are radically fractured. In design development we use tools for synthesis, simulation, analysis, emulation, test, design capture and layout, assemblers, debuggers, linkers, loaders, compilers, executables, kernels, embedded operating systems, full blown operating systems, and so on. This plethora of design environments represents a very intense management challenge. There is hope. Embedded tools of the future will provide a homogenous, top down approach to system design and cure embedded development disorders.

John Fogelin, Wind River Systems, Alameda, CA

In consideration of the ever-expanding market of embedded software, the geometric increase in software content, and the stringent market windows that must be satisfied, the time has never been better to focus on a fundamental means to compress the design and implementation cycle of an embedded application. Proponents of hardware/software co-design contend that de-serializing the hardware and software design phases offers the greatest opportunity to streamline application development. While progress in the past decade has enabled full simulation of OS/application interaction well in advance of stable hardware, and also even enabled simulation environments that allow software simulators to effectively integrate hardware simulators for full simulation of software I/O with ASIC designs, a principal challenge remains. Without the emergence of a uniform embedded application development methodology, the hardware CAD systems and software IDEs still largely speak an entirely different language. With regard to upcoming innovation, it will be in those disciplines of application design for which a uniform language exists, with little semantic variance, that the full benefits of hardware/software co-design will be realized. With the mathematical underpinnings of most DSP algorithms, those applications characterized by a high degree of signal filtering, compound filtering, signal transforms, etc. will likely be the earliest benefactors of this emerging design technique.