Perspectives

Challenges of Rapidly Emerging Consumer Space Multiprocessors

Zeljko Zilic
McGill University

Sandeep Shukla
Virginia Tech

Prabhat Mishra
University of Florida

■ **In the computer** systems industry, for the first 30 years (roughly, 1950–1980), the advances in technology were primarily aimed at high-end or enterprise-class hardware, such as Cray or IBM mainframes. For the next 30 years, the personal computing industry was the arena in which technological advances (as predicted by Moore's law) were put into practice. For the past couple years, we have observed another shift in which the traditional domain of consumer electronics, such as smartphones, tablets, and game consoles, is leading the pack of technological trailblazers.

While the placement of the newest technology into the hands of a wide variety of users is tremendously empowering, industry has—by widening the base of early adopters—effectively cornered itself such that there is little room for errors. Design, validation, and quality assurance practice need to be strengthened as the user base widens. At the same time, the difficulties in making sure systems are correct are increasingly complex and spread across all levels, including that of application development.

Modern smartphones and tablet devices ought to be multiprocessor-based to continue providing a satisfying user experience, on par with today's powerful PCs, in terms of functionality and speed. Moreover, the delivery of increasingly more-powerful devices operating under a fixed, modest energy consumption regimen ultimately requires new and scalable SoC architectures. Creating such multiprocessor platforms results in architectures that are increasingly complex and heterogeneous, and which employ irregular memory hierarchy organizations. Ultimately, such architectures do not readily lend themselves to the development of efficient software.

Indeed, a quick check through emerging smartphone and tablet SoCs shows that the multicore SoCs (most often ARM Cortex-A9 or -A15 processors) are complemented with other types of processing power. Consider, for instance, the Texas Instruments OMAP 4 and the OMAP 5 platforms, which include four processors: two Cortex A-15 and two Cortex-M4 cores, aimed at providing sufficient general-purpose and DSP computing, respectively, while at the same time reducing energy consumption. Other vendors have followed suit, notably Qualcomm's multicore Snapdragon and Nvidia's incoming quad-core Tegra 3 platform with dozens of GPU cores that complement four high-end Cortex A-series cores.

With the growing number of cores in the multicore architecture, as well as with the cores' increasing heterogeneity, design challenges are going to escalate. Most progress made thus far in realizing multicore hardware has been achieved by reusing the designs tailored for small-scale shared-memory processors (SMPs). By increasing the number of processors, it is apparent that the processor (and the cache) connectivity—including the cache coherence protocols—needs to evolve, which leads system, circuit, and software designers into unfamiliar territory and results in an increased development effort as well as a reduced verification ability before the first silicon is even fabricated. For instance, Intel and ARM teams could have helped themselves by reusing the verification of cache coherence controllers through substantial reuse of their previously developed traditional bus-based snoop protocols. The need for new interconnects (e.g., Intel has just taken the first steps in that regard by introducing the ring bus), coupled with the emergence of more scalable ways of cache sharing, such as the nonuniform cache architecture (NUCA), will prevent quick progress, and will instead require that new system design, verification, and debug progress proceeds in parallel.

Verification tasks already spill over significantly into post-fabrication territory, and the emerging multiprocessors are going to exacerbate the ability to provide platforms and solutions for meaningful post-fabrication debug. Increasingly multithreaded software cannot be debugged by traditional intrusive methods, which will mask or possibly cause faults involving asynchrony between the threads or even between disparate computational cores. At the same time, the amount of data to be processed during debug, along with the presence of the extended interconnect, are rendering existing debug methods for small-scale multiprocessors inadequate. Suitable debug solutions are needed *today*, even as the system architectures evolve.

The problems presented by these issues is compounded by the obvious gap in software paradigms, tools, and productivity frameworks, a gap that has been further widened by the new types of applications companies are developing. This gap is causing challenges unlike anything seen before, with ever shorter development times before delivering a product to the mass market.

The solution for these woes in the design must be multifaceted, including simultaneous advances in a variety of design and verification techniques. However, it is a certainty that vendors must approach development at higher abstraction levels and maintain automation throughout the development process. If vendors do that, the advances in high-level design and validation will be decisive. Simply put, the designers must maintain a high-level view of the system to quickly explore the innovation space, assess the performance for intended applications, and then, once the architecture is finalized, ensure that the detailed implementation process maintains the correctness and quality goals. At the same time, given the enormous difficulties facing the software side (programming models, tools, productivity, and so on), high-level design would allow the programmers to address software development needs early and in parallel to architecture and hardware development.

To ensure continuing progress at the speed of consumer electronics deployment, there are two unrelated requirements. First, with the apparent compelling need for innovation in architectures, tools, and methodologies, sound high-level design and verification practices will be the main prerequisite. On the other hand, the means enabling engineers to develop for, and to debug, such heterogeneous multicore systems must be devised and built correctly from the very beginning. The two requirements are perhaps conflicting, which only adds to the challenge. ■

The author biographies, and contact information for Prabhat Mishra and Sandeep Shukla, appear on p. 7 of this issue.

■ Direct questions and comments about this article to Zeljko Zilic, McGill University, 546 McConnell Engineering Bldg., 3480 University St., Montreal, Quebec, Canada H3A-2A7; zeljko.zilic@mcgill.ca.