Shuvra S. Bhattacharyya, Jeff Bier,
Wanda K. Gass, Ram K. Krishnamurthy,
Edward A. Lee, and
Konstantinos Konstantinides

Advances in Hardware Design and Implementation of Signal Processing Systems

This *IEEE Signal Processing Magazine (SPM)* forum discusses advances, challenges, and future trends in hardware design and implementation of signal processing (SP) systems. The invited forum members who bring their expert insights are: Prof. Shuvra S. Bhattacharyya (University of Maryland, College Park), Jeff Bier [Berkeley Design Technology, Inc. (BDTI)], Dr. Wanda K. Gass (Texas Instruments), Dr. Ram K. Krishnamurthy (Intel), and Prof. Edward A. Lee (Univerisity of California, Berkeley and BDTI). The moderator of the forum is Dr. Konstantinos Konstantinides (Hewlett-Packard and associate editor of *SPM*).

Our readers may agree or disagree with the ideas discussed next. In either case, we invite you to share your comments with us by e-mailing to SPM_columns_forums@yahoo.com

Moderator: Our forum discussion will focus on hardware design and implementation of SP systems, and will touch on some of the related software aspects.

In your view, what are the most important recent developments in the design of SP systems?

J. Bier: Overall, I believe the most important development is the proliferation of digital SP (DSP) into a vast range of systems and applications. Five or ten years ago, SP systems referred to a distinct class of systems, including wireless networking gear for example. These days, it is more and more difficult to find electronic systems that *do not* include some sort of SP. Likewise, it is becoming difficult to find SP systems that do not contain a large amount of non-SP functionality, such as packet processing, three-dimensional graphics, or databases. As a result, SP systems are losing their distinct identity, and DSP is becoming a nearly ubiquitous enabling technology.

E.A. Lee: I agree. There is an increasing convergence of general-purpose computing with SP as part of this trend. Many applications require support for integration of media such as video and audio, migration to handheld platforms, processing of streaming data, and wireless connectivity. Meanwhile, programmable DSP architectures continue to acquire features of general-purpose processors, such as floating-point hardware and caches. Increasing uses of flash memories and energy management in general-purpose computing also make it resemble SP more. Finally, parallelism appears to be finally here to stay with the widespread usage of multicore architectures.

S. Bhattacharyya: One of the most important recent developments I would point to has been the increasing use of embedded multiprocessor technology in the design and implementation of SP systems. A variety of commercial vendors offer single-chip multiprocessor DSP platforms, each with a collection of homogeneous or heterogeneous processors integrated on a single die, and this form of platform is becoming increasingly common in important DSP application areas, such as wireless communications, and digital video processing.

R. Krishnamurty: Indeed, the complete integration of complex analog, digital, RF, wireless, and memory functions on a single die has enabled superior performance and cost advantages for signal/video/media processing systems on a chip (SoCs). The extent of innovations needed in the architecture, circuit design, and tool methodologies to pull this off is simply mind-boggling.

In terms of energy management mentioned earlier, the emergence of on-die specialized hardware accelerators for compute-hungry workloads like video, audio, and graphics has improved the energy efficiency (performance-per-Watt) of these workloads by over 100 times. Today, a number of general-purpose microprocessors and DSPs employ on-die fixed-function hardware to accelerate tasks that would otherwise consume a lot more power when executed through the general-purpose processor units.

Moderator: Would you provide specific examples along the trends mentioned?

S. Bhattacharyya: The MSC8144E processor from Freescale Semiconductor integrates four StarCore programmable digital signal processor cores and is aimed at high-performance communications applications, such as wireless infrastructure.

Another example is the Texas Instruments TNETV 3020, which incorporates six 500 MHz TMS320C64x cores, and over 5.5 MB of memory on a single-chip platform.

A third example is the picoChip PC205, which is based on an array of 248 VLIW processor cores that is integrated

Digital Object Identifier 10.1109/MSP.2008.929838

Hard-to-wear. Cartoon by Tayfun Akgul (tayfun.akgul@ieee.org).

along with an ARM 926EJ-S microprocessor for control functions.

Modern platform field programmable gate arrays (FPGAs) also provide integration of multiple hard or and soft processor cores along with DSP-oriented hardware accelerators and high-speed input/output interfaces.

J. Bier: My thought too—and it is interesting how FPGAs are increasingly being used as compute engines in SP-intensive applications.

Moderator: What is the relevance/impact of these developments on the SP consumer market?

J. Bier: In my view, the most significant impact is that consumer products—from automobiles to MP3 players to appliances—are becoming more efficient (for example, more efficient appliances enabled by DSP-based motor control), more connected (e.g., an in-car navigation system that receives real-time traffic information), more aware of their environment (e.g., automotive lane-departure warning systems), and easier to use (e.g., spoken messages instead of beeps to alert the user).

Moderator: What are the implications of these developments on the SP systems currently in use?

S. Bhattacharyya: The processor architectures and development environments are mostly seeing such implications.

Moderator: Would you elaborate further, first on the impact on the processor architectures?

S. Bhattacharyya: Because of the increasing diversity in the types of different architectures to consider, including conventional single-processor engines, small collections of relatively complex processors, large collections of simple processors, different heterogeneous combinations of processors, etc., it is more and more difficult and costly to understand what the best architectural options are for a particular application and to retarget designs to emerging platforms.

J. Bier: The main changes in processor architecture have been:

- incorporation of DSP-oriented features into a wide range of general-purpose microprocessors, ranging from microcontrollers (e.g., dsPIC) to CPUs for network server applications. Of course, this has been going on for a long time, but in the last few years it has become nearly universal.
- incorporation of DSP-oriented features in to some FPGAs and the adoption of FPGAs as compute engines for DSP applications
- widespread use of heterogeneous multicore architectures in SoCs for many high-volume DSP applications, e.g., cellular handsets and portable media players.

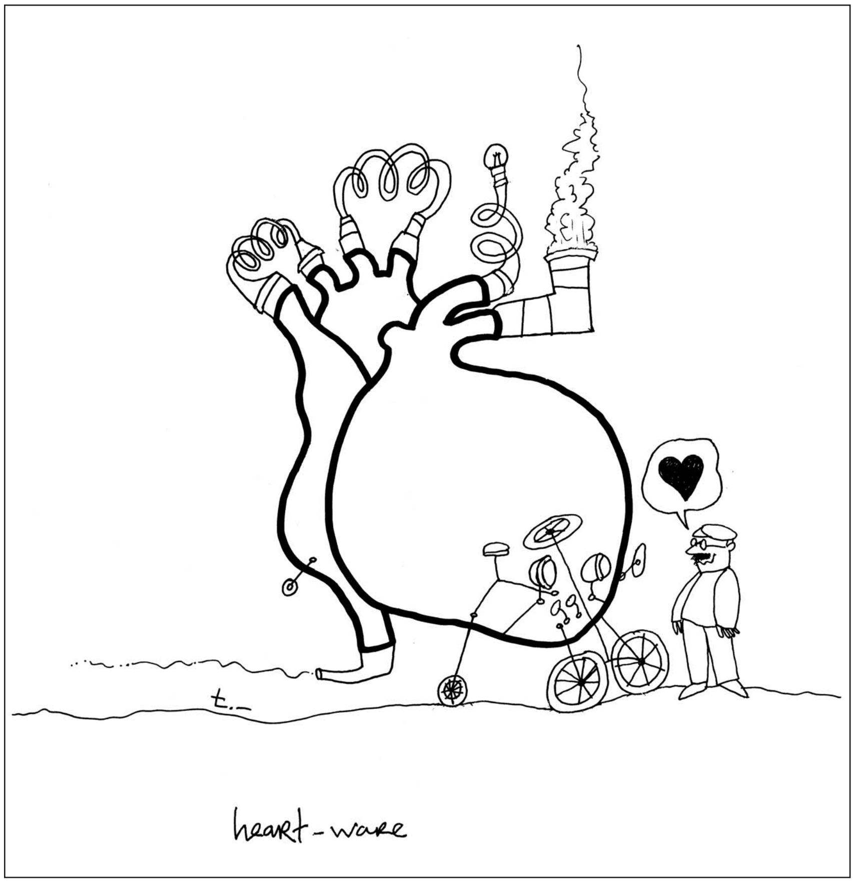
W. Gass: The best example of a heterogeneous multiprocessor system that combines hardwired accelerators, special purpose processors, and micro-controllers is the application processor that is found in cell phones. This contains an open platform for developing the user interface, accelerators for video standards, graphics processor for games, hardwired image processing block for taking pictures, and many different interfaces to other devices.

Moderator: Let us touch on the impact on the development environments as well.

W. Gass: Early in the design process, architecture decisions need to be made regarding hardware versus software, interconnect topology/bandwidth, and memory architecture. It is important to note that for such a complex heterogeneous system such as the one I just mentioned, the design and debugging requires a variety of tools, and that the traditional multiprocessing tools are not sufficient.

S. Bhattacharyya: In addition, due to the trend towards multiprocessors and platform FPGAs, development environments need to provide increasing support for concurrent programming and hardware/software codesign. Unlike single-processor designs, where C language is like a de-facto choice for DSP implementations, there is relatively large variation in the forms of support for multiprocessor and hardware/software system integration. This makes it much more difficult to migrate designs across platforms and explore different implementation options.

Novel programming models based on dataflow graph representations have been used. Although such programming models have been in use in simulation for over two decades—e.g., in popular



Heart-ware. Cartoon by Tayfun Akgul (tayfun.akgul@ieee.org).

commercial tools such as, Agilent ADS, National Instruments LabVIEW, and several others—their utilization for software implementation has been receiving greater attention in recent years, due largely to their potential for exploiting the parallelism in multiprocessor DSP platforms.

Moderator: What are the current challenges in the hardware design and implementation of SP systems?

W. Gass: I believe the biggest challenges are in the area of delivering the lowest power while also providing the programmability required of system upgrades, changing standards, and the need to add new features.

R. Krishnamurthy: Yes, the ability to reduce operating power supply voltage to levels unforeseen before, i.e., voltage levels well below 0.5 V is essential to allow further integration of complex DSP functions and accelerators on a single chip. While the developments discussed so far have enabled complex energy-efficient SP SoCs, we still face significant challenges in tacking some of the emerging workloads, e.g., multiprotocol radios, advanced low-power graphics, encryption, speech and video tasks, and extremely low voltage operation that is essential for some of the futuristic biomedical applications.

Along the lines of the accelerator development I mentioned above, it is not going to be cost effective to integrate a large number of accelerators for diverse workloads. One of the impending challenges we are going to face is defining and developing a unified accelerator methodology that can help accelerate a varied set of workloads through a common fabric. And how do we tie it all together? What would be on-die interconnect network look like? We are already well into the era of multicore computing—does this mean some of the cores would be committed to accelerating SP tasks? If so, would a reconfigurable architecture make sense so that it can be customized depending on application?

J. Bier: I agree. The problem with today's approach of relying heavily on a multitude of diverse, specialized accelerators is that it lacks flexibility and scalability. Some mainstream chips today incorporate 20 or more unique processing engines. As chip transistor budgets continue to grow, this approach doesn't scale—I can't imagine anyone being able to program a chip containing 200 distinct processing engines.

W. Gass: Well, hardwired accelerators provide the most power efficient solution, but the least flexibility. For systems that are battery operated such the cell phone, delivering the most functionality at the lowest power is a key requirement. For example, most of the video codecs are implemented in hardwired modules because the need for flexibility is very limited. Likewise image processing algorithms in a digital-camera subsystem are customized for the camera front end. On the other hand, music and speech functions are usually implemented on the general-purpose processor while it is in a low-power mode due to the low MIPS requirements.

Overall, delivering a low power solution requires architecture tradeoffs, software and hardware control of power modes, advanced circuit techniques, and ability to adapt to workloads. There are tradeoffs to be made during architecture definition as well as flexibility that needs to occur at runtime. There is a need to model the whole system, because one may optimize one component, but when placed into the whole system, bottlenecks or even deadlock conditions can result in inefficient or useless implementations.

J. Bier: You made a critical point, that the sum of optimized elements is not necessarily an optimized whole.

Moderator: With the challenges mentioned in our discussion, it makes sense to take a look at the software side as well in SP systems. What challenges do we see if any?

E.A. Lee: To handle increasing complexity in applications, designers of embedded software for SP have to use more elaborate software engineering techniques. In the past, the key challenge was to get the requisite throughput from inexpensive and low-power hardware, so finely tuned assembly code on idiosyncratic processor architectures dominated the practice. This approach does not adapt well to increasing richness of functionality. Designers need to use higher-level languages, object-oriented design, data abstraction, memory management, and multitasking. Unfortunately, these techniques typically obscure performance and predictability. Building reliable systems that meet specifications under all operating conditions becomes extremely challenging.

At the same time, as DSP architectures more resemble general-purpose architectures, their timing behavior becomes more complex. Pipeline interlocks, speculative execution, cache management, and memory management techniques in the hardware introduce considerable variability in timing, making it difficult to ensure repeatable behavior.

Combining these two effects (elaborated software engineering techniques and architectures), getting predictable and repeatable behavior gets difficult. Designers are left with a test-and-refine design methodology that depends much more on bench testing than on thoughtful analysis or understanding of the design. Designs are brittle, in that small changes can have big consequences. Having a task finish unexpectedly early, for example, may cause other tasks to miss deadlines, paradoxically. Testing for all possible operating conditions, of course, becomes impossible because systems are increasingly networked, open, and complex.

Moderator: Let us look ahead. What future directions can you identify in hardware design and implementation for SP systems?

E.A. Lee: In my opinion, computer architects have gone overboard in elaborate architectural techniques that deliver average case performance improvements at the cost of increased timing variability. I believe that for most SP applications, repeatable behavior is more important than higher average case throughput. SP systems are normally held to higher reliability standards than general-purpose software, and failures due to untested operating conditions are not tolerated by the customers. As a consequence, we need a new generation of architectures that efficiently support deep pipelines, memory hierarchy, distributed and shared memory, and memory management, but not at the cost of increased timing variability. The timing behavior has to be predictable and, most importantly, repeatable.

J. Bier: Also, the industry needs more scalable processor architectures, so that a single architecture can be used efficiently across a very wide range of applications. We have far too many different processor architectures today, and each has to be supported with its own tools and software components. I believe that we will see more homogeneous architectures (which is not necessarily to say completely homogeneous architectures). This is in part because homogeneous architectures tend to be more scalable and more flexible. I believe that over the next ten years we will have fewer unique chips, each serving a wider range of applications. Also, more homogeneous architectures tend to be easier for tools to manage.

R. Krishnamurthy: My belief is that the next era of nanoscale SP systems will evolve into integrating even more complex analog, digital, memory, accelerator, and memory functions on a single heterogeneous chip and percolate into newer application domains such as medical, implantable, wearable electronics and sensor networks.

S. Bhattacharyya: And I would say, adding support for reconfigurability and adaptivity in a more integrated way across different layers in the design and implementation process, from algorithm to implementation. This will be important to support in more reliable ways the diverse and dynamically changing operational constraints and needs for multistandard operation in SP applications. There have been important advances over the years in isolated subsets of layers, such as advances in adaptive modulation and coding schemes for wireless communications; FPGA technology, including dynamically reconfigurable FPGAs; dynamic power management technology to extend battery life while maintaining required levels of performance; and DSP-oriented dataflow languages for handling more diverse kinds of application dynamics. New methods are needed to manage adaptivity and reconfigurability as first-class citizens in the design process and provide more seamless support for them across different layers of abstraction.

W. Gass: Support for low power optimizations at several different levels seems to be an important direction of work as well. Different tools are needed for top-level power control, module-level architectures, circuit techniques for logic and memory, and adaptive techniques to compensate for variations in process and operating temperature.

At the top level, a power controller is required to control which modules or subsystems are active and running in low power mode or high performance mode and which ones are in sleep mode. At the module level, tradeoffs between power and performance need to be made, to get to the lowest power solution. At the circuit level, designs must include retention FF for logic and standby mode for SRAM. At the lowest level, delay-locked loops must be used to ensure that performance is met at the minimum power. For chips that have faster transistors, the operating voltage can be reduced so reduce the leakage and active power dissipation. For slower transistors, the operating voltage can be increased to meet performance requirements.

Moderator: What can we predict in terms of software design to accompany the directions mentioned earlier?

E.A. Lee: On the software engineering side, SP system designers need to develop techniques that match their needs better than the established techniques. For example, object-oriented technologies offer powerful ways of building large systems out of software components. But their realizations today do not express concurrency, timing, or stream processing. A new generation of component technologies that do express these properties is emerging.

Additionally, getting repeatable behavior depends heavily on timing. Techniques developed for real-time computing, however, continue to have serious weaknesses. For example, they rely on knowing "worst-case execution time" (WCET) for software components, something that in today's environment is often effectively unknowable. Worse, even knowing these WCETs is insufficient, because many scheduling techniques suffer scheduling anomalies when tasks finish unexpectedly early. They also rely on a style of multitasking (threads) that in essentially incomprehensible to designers, and therefore cannot result in reliable systems.

W. Gass: On top of my list would be multithreading/multitasking for heterogeneous systems. There needs to be a control mechanism to allow memory to be shared when necessary or local to be more power efficient. Architectures must be able to coordinate tasks that might have different operating systems or no operating system at all. The system must be able to schedule tasks and handle message passing and data flow between processes. The top-level architecture must comprehend the high-level scheduling of tasks and provide a mechanism for initiating the tasks and monitoring their completion in a distributed heterogeneous multiprocessor system. Data flow may be handled by the processors in a shared memory system or by the scheduler in a distributed memory system. In real-time SP systems, this schedule is known prior to run time and then monitored by the scheduler to maintain real-time performance.

S. Bhattacharyya: To the ideas mentioned earlier, I would add the development of effective programming environments and compiler technology based on dataflow and related streaming-oriented programming models, as an important direction of further work.

Moderator: It looks like the future of SP system design will continue to be as exciting and challenging as ever. Thanks to all of you for your valuable insights and for participating in this forum.

PANELISTS

Shuvra S. Bhattacharyya (ssb@umd.edu) is a professor in the Department of Electrical and Computer Engineering, University of Maryland at College Park. He holds a joint appointment in the University of Maryland Institute for Advanced Computer Studies (UMIACS), and an affiliate appointment in the Department of Computer Science. He is also the chair of the IEEE Signal Processing Society Technical Committee on Design and Implementation of Signal Processing Systems.

Jeff Bier (bier@BDTI.com) is cofounder and president of Berkeley Design Technology, Inc (BDTI). He oversees BDTI's benchmarking and competitive analysis of chips, tools, and other technologies as related to signal processing systems.

Wanda K. Gass (gass@ti.com) is a Texas Instruments (TI) Fellow and the ISA architect for high performance DSP processors at Texas Instruments. She is a Fellow of the IEEE.

Ram K. Krishnamurthy (ram.krishnamurthy@intel.com) is a senior principal engineer with the Microprocessor Technology Labs, Corporate Technology Group, of Intel Corporation, Hillsboro, Oregon, where he heads the high-performance and low-voltage circuits research group. He is a Senior Member of IEEE.

Edward A. Lee (eal@eecs.berkeley.edu) is the Robert S. Pepper Distinguished Professor and former chair of the Electrical Engineering and Computer Sciences (EECS) department at U.C. Berkeley. He is also a cofounder of BDTI, Inc., where he is currently a senior technical advisor. He is a Fellow of the IEEE.

Konstantinos Konstantinides (k.konstantinides@ieee.org) is a senior technical product marketing manager in the Connected Entertainment group of Hewlett-Packard. He is a Senior Member of the IEEE and an associate editor of *IEEE Signal Processing Magazine*. [SP]

Errata

There was an error in the article "Predicting Reaching Targets from Human EEG," *IEEE Signal Processing Magazine*, vol. 25, no. 1, pp. 69-77, 2008. The original and corrected text appear below.

ORIGINAL TEXT

The vectors in $\hat{S}$ are termed "independent components" (ICs). The columns of $\hat{A}^{-1}$ indicate how to construct an individual IC as a weighted combination of channels of X. Thus, we can visualize the distribution of an IC over the scalp by plotting the values from a column of $\hat{A}^{-1}$ at each electrode location on the scalp to generate IC scalp maps (see Figure 1).

CORRECTED TEXT

The rows of $\hat{S}$ are termed "independent components" (ICs). A row of $\hat{A}^{-1}$ indicates how to construct an individual IC as a weighted combination of sensor readings in X, and a column of $\hat{A}$ indicates how much the corresponding IC contributes to each individual sensor reading. Thus, we can visualize the distribution of an IC over the scalp by plotting the values from a column of $\hat{A}$ at each electrode location on the scalp to generate IC scalp maps (see Figure 1). [SP]

Digital Object Identifier 10.1109/MSP.2008.930488