SODA: A High-Performance DSP Architecture for Software-Defined Radio

Software-defined radio (SDR) belongs to an emerging class of applications with the processing requirements of a supercomputer but the power constraints of a mobile terminal. The authors developed the Signal-Processing On-Demand Architecture (SODA), a fully programmable architecture that supports SDR, by examining two widely differing protocols, W-CDMA and 802.11a. It meets power-performance requirements by separating control and data processing and by employing ultrawide SIMD execution.

Yuan Lin
Hyunseok Lee
Mark Woh
Yoav Harel
Scott Mahlke
Trevor Mudge
University of Michigan at Ann Arbor

Chaitali Chakrabarti
Arizona State University

Krisztián Flautner
ARM Ltd.

Communication has become one of the central uses of computing technology, and applications that facilitate interpersonal communication, such as desktop publishing, graphic design, e-mail, and Web browsing, have been primary factors in driving the evolution of microprocessors and computer systems. Meeting applications' processing requirements has historically dominated the concerns of processor and system architects. With the proliferation of wireless mobile communications, the emphasis has shifted to the networking protocols and signal processing required to sustain these applications' necessary bandwidth.

Recently, an increasing number of wireless protocols have emerged that make interoperability more challenging and the cost of supporting multiple protocols using hardwired application-specific integrated circuit (ASIC) solutions more expensive and complex. In this article, we present a design study for the Signal-Processing On-Demand Architecture (SODA), a fully programmable architecture that supports software-defined radio (SDR).

An important aspect of our work is to identify wireless protocols' power-performance characteristics and propose a digital signal processing (DSP) system based on these characteristics, which can meet requirements previously unmet by other DSP solutions. Our analysis shows that embedded DSP processors' data concurrency requires different solutions from those of general-purpose processors. We've proposed a DSP system consisting of one controller and four ultrawide SIMD processing elements (PEs), with data communication done through explicit direct memory access (DMA) instructions.[1] This type of system, with a control processor and multiple data processors, has been explored previously for high-end general purpose multimedia computation, such as the IBM Cell processor.[2] However, to our knowledge, this is the first time researchers have proposed such a system with energy characteristics suitable for mobile computing.

SDR challenges and benefits

The operation throughput requirements of current third-generation (3G) wireless protocols are already an order of magnitude higher than the capabilities of modern DSP processors, a gap that's likely to grow in the future. Figure 1 shows the computation and power demands of a typical 3G wireless protocol. The figure also compares SODA's theoretical peak performance throughput and power consumption with that of other existing DSP, media, and general-purpose processor systems. Although most DSP processors operate at an efficiency of approximately 10 million operations per second (MOPS) per milliwatt (mW), the typical wireless protocol requires 100 MOPS/mW. Hence, most wireless protocols to date have been implemented with custom hardware. Although custom hardware can meet the operational requirements, a programmable solution offers many potential advantages:

- A programmable architecture would allow multimode operation, running different protocols depending on the available wireless network—GSM in Europe, CDMA in the USA and some parts of Asia, and 802.11 in coffee shops. This is possible with less hardware than custom implementations require.
- A protocol implementation's time to market would be shorter because it could reuse the hardware. The hardware integration and software development tasks could progress in parallel.
- Prototyping and bug fixes would be possible for next-generation protocols on existing silicon through software changes. The use of a programmable

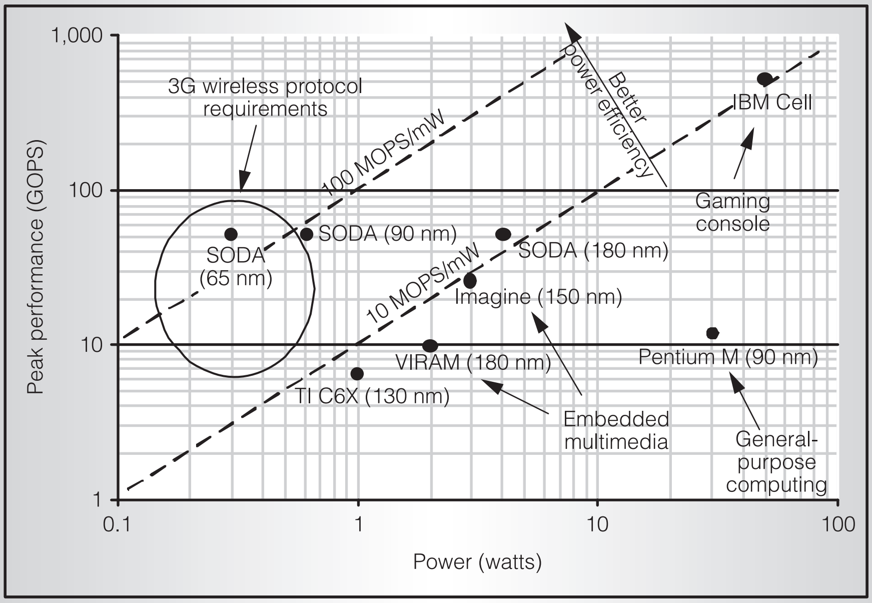


Figure 1. Throughput and power requirements of typical 3G wireless protocols, and theoretical peak computational efficiency of SODA, other DSPs, and general-purpose processors. We calculated the results for 16-bit fixed-point operations.

solution would support the specification's continuing evolution; after the chipset's manufacture, developers could deploy algorithmic improvements by changing the software without redesign.
- Chip volumes would be higher because the same chip could support multiple protocols without requiring hardware changes.

Ultimately, we believe that the need to support many increasingly complex wireless protocols will make the use of programmable systems for these protocols inevitable.

Analysis of wireless protocols

Wireless protocols are collections of disparate DSP algorithm kernels that work together as one system. There are four major components: filtering, modulation, channel estimation, and error correction. Filtering algorithms suppress signals transmitted outside the allowed frequency band to minimize interference with other frequency bands. Modulation algorithms map source information onto the transmitter's signal waveforms, and receivers demodulate the signal waveforms back into source information. Channel estimation algorithms calculate the channel conditions to synchronize the two communicating terminals to ensure lock-step communication between the sender and receiver. Error-correction algorithms combat noisy channel conditions. The sender encodes the original data sequence with a coding scheme that inserts systematic redundancies into the output, which the receiver decodes to find the most likely original data sequence.

We use two representative wireless protocols to understand the architectural requirements of physical layer signal processing for SDR: Wideband code division multiple access (W-CDMA)[3] is one of the most common 3G cellular protocols, and 802.11a[4] is a standard wireless local area network (WLAN) protocol. We chose these two benchmarks because they are sufficiently different from each other algorithmically and are representative of the large spectrum of algorithms that an SDR platform must support. Both protocols have complex interalgorithm and intra-algorithm interactions and behaviors. Figure 2 shows overall block diagrams for the operation of W-CDMA and 802.11a, including each DSP algorithm's vector width and data precision.

Our key observations fall into two categories: protocol system-level behavior and DSP algorithm-level behavior.

System level

Wireless protocols generally consist of multiple DSP algorithm kernels connected together in feed-forward pipelines. Data packets stream through these macro pipelines sequentially, resulting in almost no temporal locality. Thus, cache structures provide little additional benefits, in terms of power and performance, over software-controlled scratch-pad memories.

For low-throughput interkernel communication traffic, the data throughput between DSP algorithm kernels isn't high. This implies that we can map interkernel data traffic onto low-throughput, low-power interconnects with minimal performance degradation.

We can stream some interkernel communications, where the receiving kernel (such as a filter) can process input data individually. Other interkernel communications must be buffered because the receiving kernels (such as the interleaver and turbo decoder) require blocks of data. Kernels with the same throughput but different communication patterns have dramatically different hardware requirements. Streamed kernels need only small first-in, first-out (FIFO) queues, but the buffered kernels require a large memory space.

All wireless protocols have real-time deadlines. Meeting these deadlines, a challenge that has received scant attention in previous DSP architectural studies, requires concurrent execution management for multiple DSP algorithms.

Algorithm level

Most of the computationally intensive DSP algorithms have abundant data-level parallelism. As Figure 2 shows, the heaviest workloads of the W-CDMA and 802.11a protocols, including the searcher, low-pass filter (LPF), fast Fourier transform (FFT),

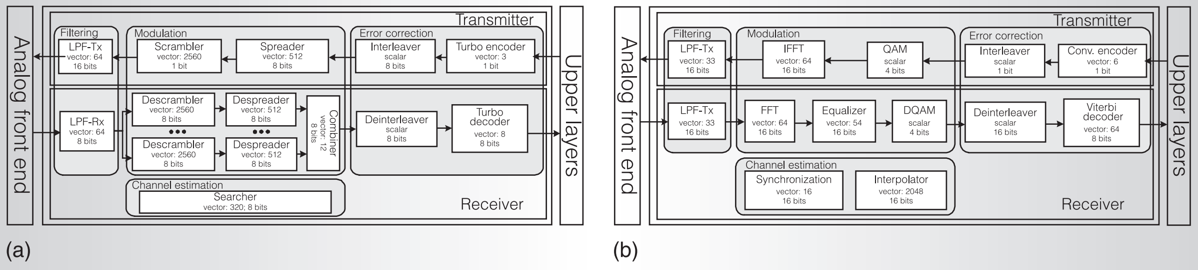

Analog front end
Transmitter
Filtering
LPF-Tx vector: 64 16 bits
Modulation
Scrambler vector: 2560 1 bit
Spreader vector: 512 8 bits
Error correction
Interleaver scalar 8 bits
Turbo encoder vector: 3 1 bit
LPF-Rx vector: 64 8 bits
Descrambler vector: 2560 8 bits
Despreader vector: 512 8 bits
Descrambler vector: 2560 8 bits
Despreader vector: 512 8 bits
Combiner vector: 12 8 bits
Deinterleaver scalar 8 bits
Turbo decoder vector: 8 8 bits
Channel estimation
Searcher vector: 320; 8 bits
Receiver
Upper layers
(a)
Analog front end
Transmitter
Filtering
LPF-Tx vector: 33 16 bits
Modulation
IFFT vector: 64 16 bits
QAM scalar 4 bits
Error correction
Interleaver scalar 1 bit
Conv. encoder vector: 6 1 bit
LPF-Tx vector: 33 16 bits
FFT vector: 64 16 bits
Equalizer vector: 54 16 bits
DQAM scalar 4 bits
Deinterleaver scalar 16 bits
Viterbi decoder vector: 64 8 bits
Channel estimation
Synchronization vector: 16 16 bits
Interpolator vector: 2048 16 bits
Receiver
Upper layers
(b)

Figure 2. Physical layer operation of W-CDMA (a) and 802.11a (b) wireless protocols. Each block includes the algorithm's name, vector or scalar computation, vector width, and the data precision. We also grouped the algorithms into four categories (shown in shaded boxes): filtering, modulation, channel estimation, and error correction.

and Viterbi decoder, all operate on wide vectors.

Most algorithms operate on variables with small values. Our analysis of W-CDMA and 802.11a suggests that the architecture should provide support for 8- and 16-bit fixed-point operations; 32-bit fixed-point and floating-point support are unnecessary. As Figure 2 shows, W-CDMA's algorithms operate mostly on 8-bit data, whereas 802.11a's algorithms operate mostly on 16-bit data.

Design trade-offs for SDR

Now we'll look at the architectural implications of supporting wireless protocols. Considerations include managing concurrent DSP algorithm execution, controlling interalgorithm communication, meeting real-time deadlines, and supporting high-throughput DSP algorithms.

Control plane versus data plane

Complete software implementations of wireless protocols usually require two steps: implementing the DSP algorithms and the protocol system. The DSP algorithms are computation-intensive kernels that have relatively simple data-independent control structures. The protocol system has a relatively light computation load but complicated data-dependent control behavior. Therefore, our proposed SDR solution includes a two-tiered architecture: a set of data processors that handle heavy-duty data processing and a control processor that handles the system operations and manages the data processors through remote procedure calls and DMA operations.

Static multicore scheduling versus multithreading

Traditional microarchitectural techniques that researchers developed for server-class multiprocessors, such as simultaneous multithreading and cache coherency, offer programmers a convenient abstraction, but they can be of limited value in high-throughput embedded systems. Strict real-time requirements make deterministic architectural behavior necessary. This implies that microarchitectural features that trade off good average-case performance for nondeterministic and poor worst-case performance (such as caching, multithreading, and prediction) are ill suited to these applications.

To reduce hardware complexity and produce efficient deterministic code behavior, we omit multithreading and cache coherency support. Instead, our execution model breaks up each protocol pipeline into kernels and statically assigns each kernel to a PE, which is then statically scheduled to execute according to the algorithm data flow. This model grew out of our observations that interkernel communication throughput is low, and intrakernel computational throughput is high. Therefore, the static scheduling approach can result in less communication traffic than splitting kernels into threads.

Scratch-pad memory with data streaming

In addition to data computation, programmers also need to handle the data communications between algorithms. These interalgorithm communications are usually data-streaming buffers that are ideal for nonblocking DMA transfers: while the processors operate on the current data, we can stream the next batch of data between the memories and register files in the background. Researchers have previously proposed streaming computations for multimedia processor architectures, including the Imagine[5] and IBM Cell[2] processors. They have shown that scratch-pad memories, instead of cache, are best suited for streaming applications. We find that streaming computation is also suitable for wireless protocols.

Wide SIMD execution

The computationally intensive DSP algorithms in wireless protocols usually contain operations on wide vectors. In addition, vector widths and strides can generally be calculated at compile time. Although a vector architecture would be a good fit for a wide-vector computation, the extra hardware support for dynamic vector management is unnecessary because we can statically schedule all data operations. This favors a wide single-instruction, multiple-data style clustered execution. Traditional SIMD architectures have a narrow SIMD width because of the difficulties in data alignment. In addition, general-purpose SIMD accelerators usually support a large range of data sizes (for example, Intel's MMX[6] supports 8-, 16-, 32-, and 64-bit SIMD operation). Therefore, a SIMD system's bottlenecks are often the data movement and alignment operations, not data computation. Previous studies have addressed this problem through complex multiported memories and register files or a full crossbar interconnect. In the context of the power budgets for mobile devices, however, these are infeasible solutions.

Wireless protocols' DSP algorithms have well-defined intravector data-permutation patterns and operate on 8- and 16-bit data. These traits significantly simplify the data-movement requirements. Therefore, we can afford to scale up the SIMD width to exploit DSP algorithms' wide vector operations, with the intravector data permutation supported through an SIMD shuffle network.

Asymmetric VLIW instructions

In addition to the heavy vector computation, wireless protocols include many small, yet equally important scalar DSP algorithms. Wide SIMD execution units are too inefficient for these scalar and narrow SIMD operations. Therefore, architectural support for scalar execution is also necessary. In most cases, scalar operations can execute concurrently in VLIW (very long instruction word) lock-step with the SIMD operations. The VLIW is asymmetric because instructions for SIMD pipeline can't execute on the scalar pipeline and vice versa.

SODA architecture implementation

Figure 3 shows the SODA multiprocessor architecture, which consists of multiple PEs, a scalar control processor, and global scratch-pad memory, all connected through a shared bus. Each SODA PE consists of five major components:

- a SIMD pipeline for supporting vector operations,
- a scalar pipeline for sequential operations,
- two local scratch-pad memories for the SIMD pipeline and the scalar pipeline,
- an address generation unit (AGU) pipeline for providing the addresses for local memory access, and
- a programmable DMA unit to transfer data between memories and interface with the outside system.

The SIMD, scalar, and AGU pipelines execute in VLIW-style lock-step, controlled with one program counter (PC). The DMA unit has its own PC; its main purpose is to perform memory transfers and data rearrangement. It's also the only unit that can initiate memory accesses to the global scratch-pad memory.

The SIMD pipeline, designed to handle computationally intensive DSP algorithms, consists of a 32-way 16-bit integer data path, with 32 arithmetic units working in lock-step. Each data path includes a two-read-port, one-write-port 16-entry register

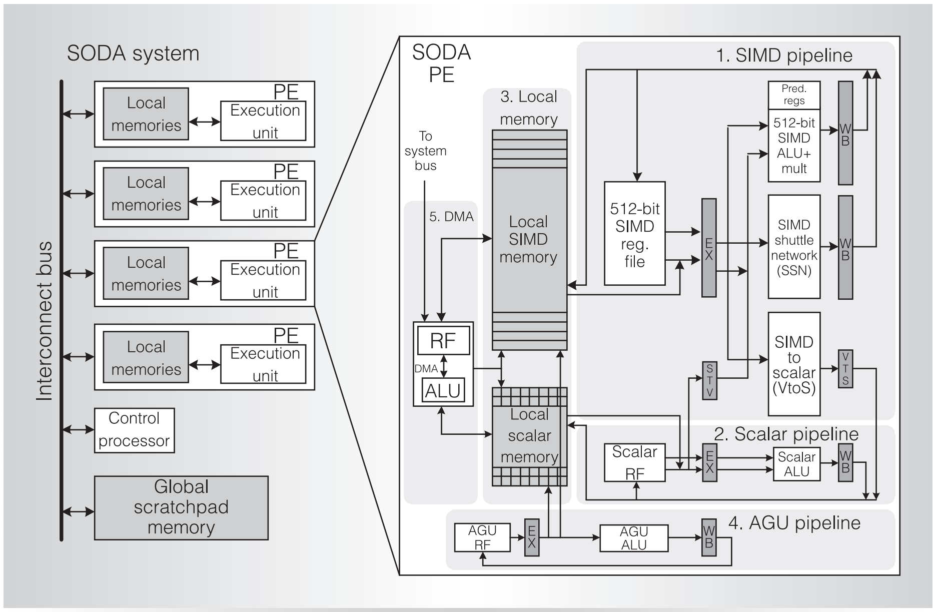


Figure 3. SODA for software-defined radio. The system consists of four data processing elements (PEs), one control processor, and a global scratch-pad memory, all connected through a shared bus. Each PE consists of a 32-wide 16-bit SIMD pipeline, a 16-bit scalar pipeline, two local scratch-pad memories, an address generation unit (AGU) for calculating memory addresses, and a direct memory access (DMA) unit for interprocessor data transfer.

file, and one 16-bit arithmetic logic unit (ALU) with multiplier. The multiplier takes two execution cycles when running at the targeted 400 MHz. The SIMD pipeline supports intraprocessor data movements through the SIMD Shuffle Network (SSN), which consists of a shuffle exchange (SE) network, an inverse shuffle exchange (ISE) network, and a feedback path. Previous work has shown that any permutation of size N can be done with $2\log_2 N - 1$ iterations of either the SE or ISE network, where N is the SIMD width.[7] For SDR algorithms' permutation patterns, we found that we can reduce the number of iterations if we include both the SE and ISE networks.

The SIMD pipeline can take one of its source operands from the scalar pipeline. This feature, which is useful in implementing trellis computations, is done through the scalar-to-vector (STV) registers (see the SIMD pipeline in Figure 3). The STV contains four 16-bit registers, which only the scalar pipeline can write and only the SIMD pipeline can read. The SIMD pipeline can read one, two, or all four STV register values and replicate them into 32-element SIMD vectors. SIMD-to-scalar operations transfer values from the SIMD pipeline into the scalar pipeline, through the vector-to-scalar (VTS) registers (see Figure 3). SODA supports several SIMD reduction operations, including vector summation and finding the minimum and maximum.

The DMA controller is responsible for transferring data between memories. It's the only component in the processor that can access the SIMD, scalar, and global memories. A traditional DMA controller performs copies from one memory region to another, where regions are either contiguous or have a simple stride access patterns. It is usually implemented as a slave device, controlled through a set of DMA registers and synchronization instructions that execute on the master processor. In our processor, we also implement the DMA as a slave device that the scalar pipeline controls. However, it can execute its own instructions on its internal register file and ALU, similar to the scalar pipeline. This gives the DMA the ability to access the memory in various application-specific patterns without assistance from the master processor. This lets us efficiently implement inherently scalar memory-transfer algorithms.

Experimental evaluation

To test SODA's performance, we first developed W-CDMA and 802.11a physical-layer system implementations in C. Next, the benchmarks were hand-coded with the SODA instruction set. For performance evaluation, an interprocessor network simulator was built on top of our PE's cycle-accurate processor simulator. We also implemented SODA in Verilog and synthesized it for 400 MHz using the Taiwan Semiconductor Manufacturing Company (TSMC) 180-nm library. We generated the memories with the Artisan SRAM memory generator. We then estimated the power and area results for 90-nm and 65-nm processes using the Predictive Technology Models (PTM; http://www.eas.asu.edu/ptm/) and SPICE simulations assuming a logic delay modeled by 20 fan-out-of-4 gates in 180-nm technology at 1.8 V, 90-nm technology at 1 V, and 65-nm technology at 0.8 V.

Through our simulations, we found that we can meet the real-time computational requirements of 2-Mbps W-CDMA and 24-Mbps 802.11a with four SODA PEs running at 400 MHz. 802.11a has higher overall computation requirements than W-CDMA, but W-CDMA requires higher computational cycles per bit due to higher computation requirements for channel estimation and error-correction algorithms.

Through our power analysis, we found that SODA can support W-CDMA and 802.11a consuming approximately 3 W in 180-nm process technology. A typical cellular phone's power budget for the physical layer is approximately 200 mW. To see if we could meet this constraint, we estimated SODA's power and area at state-of-the-art technology nodes, 90 and 65 nm. Designs in both technologies fall within the range of acceptable power consumption: 450 and 250 mW, respectively.

The rest of our discussion focuses on W-CDMA (instead of 802.11a) because its behavior is more complex. Figure 4a shows W-CDMA's algorithm kernel mapping onto SODA, and Figure 4b shows the real-time system execution of one frame of data. The dedicated channel (DCH) is a full duplex channel consisting of the dedicated physical data channel (DPDCH) for uplink and the dedicated physical channel (DPCH) for downlink. In the W-CDMA specification, DCH also includes the dedicated physical control channel (DPCCH) uplink, which we didn't model in this study. The uplink and downlink channels are mapped onto their own PEs. This assignment achieves a relatively balanced workload across the four PEs.

To better understand W-CDMA execution, consider Figure 4b. The horizontal axis is time, and the vertical axis lists the SODA's PEs and their real-time processing utilization. The utilization of PE1, PE2, PE3, and PE4 are 60, 50, 100, and 94 percent, respectively. One W-CDMA frame contains 15 slots. Two hard, real-time deadlines must be met in W-CDMA. The first involves the power control critical path that controls the transmission power based on received signal quality. It needs to update periodically once per slot (0.67 ms). The critical path is the channel between the finite impulse response (FIR) receiver filter, demodulation, and power control. This is a streaming channel with minimal memory storage requirements. The other real-time critical path is the channel from the FIR filter to the searcher, which must complete within 5 ms and requires a large amount of data buffering. Because of the static timing characteristics, compile-time scheduling of the kernels suffices to reduce unnecessary context switching overhead. The interprocessor data-streaming communications can be handled through the DMA.

Untethered digital devices are already ubiquitous. The world has more than 1 billion active cell phones, each a sophisticated multiprocessor. With sales totaling about $400 million every year, the cell phone has arguably become the dominant computing platform, a candidate for replacing the PC. We expect to see both the types and numbers of mobile digital devices

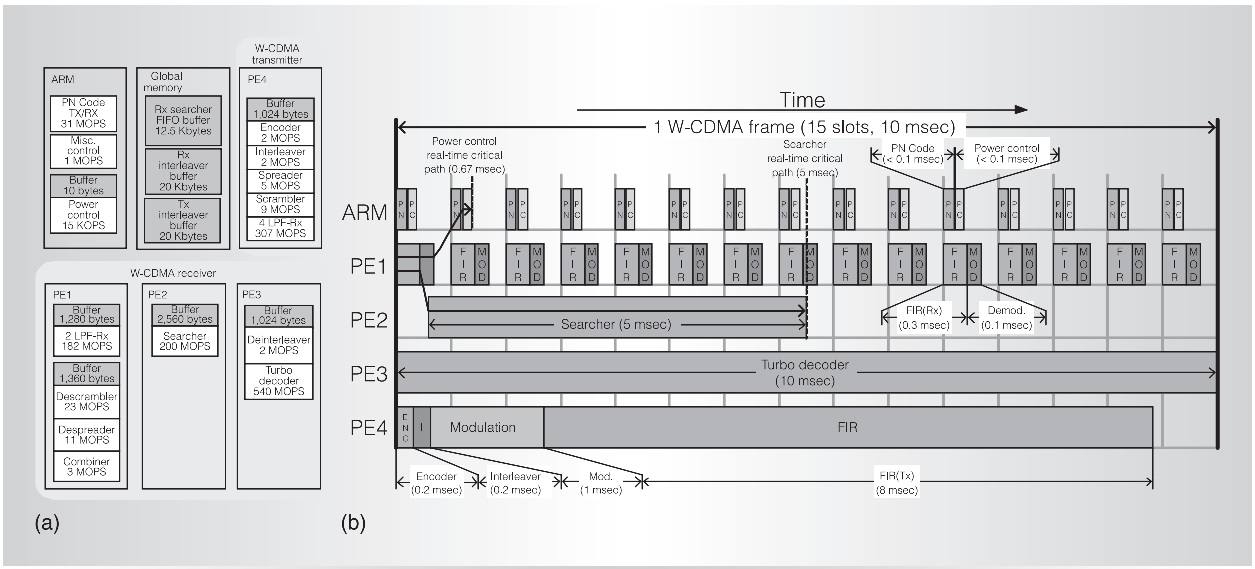


Figure 4. W-CDMA 2-Mbps DCH data channel implementation: kernel mapping with the algorithm mapping and memory allocation on the processing elements (PEs), control processor, and global memory (a); execution trace with periodic power control and searcher real-time deadlines (b).

increase in the near future; new devices will improve on the mobile phone by incorporating advanced multimedia functionalities. We also anticipate the emergence of relatively simple, disposable devices that support the pervasive computing infrastructure. With the performance requirements of mobile devices increasing exponentially, computer architects must adapt to keep up. SDR promises to deliver a cost-effective, flexible mobile communication solution by implementing the various wireless protocols in software. However, an SDR solution requires extraordinary amounts of computation on a mobile device's power budget. For architects, this means that we need to bring the power of a supercomputer to mobile devices—a mobile supercomputer.[8]

The market's interest in SDR is growing rapidly, with companies like Sandbridge,[9] Icera (www.icerasemi.com), and Phillips[10] releasing new solutions. Sandbridge's Sandblaster is a multicore system that consists of four multithreaded SIMD processors. Icera's DXP is a deep-pipelined LIW processor with four-wide SIMD execution. And Phillips EVP is a 16-wide SIMD processor with ASIC accelerators.

SODA is a DSP system consisting of one general-purpose processor and four SIMD processing elements, with data communication done through explicit DMA instructions. For current-generation wireless systems, SODA can meet the processing requirements of two widely differing protocols (W-CDMA and 802.11a) within the strict power constraints of a mobile terminal. SODA is also well positioned to meet the demands of next-generation wireless protocols through more processing elements with wider SIMD execution.

It's worth noting that a large class of computing platforms, especially those in the embedded world, are organized as a "control plane" processor with a collection of "data plane" processors. Our solution is squarely in this space, applying the control-data plane multicore architecture to SDR's embedded domain. With the rapid growth of embedded applications, we expect to see more of these types of architectures in the future. MICRO

Acknowledgments

Yuan Lin is supported by a Motorola University Partnership in Research Grant. This research is also supported by ARM Ltd. and the National Science Foundation grants NSF-ITR CCR-0325898, CCR-EHS 0615135, and CCR-0325761.

References

1. Y. Lin et al., ''SODA: A Low-Power Architecture for Software Radio,'' *Proc. 33rd Ann. Int'l Symp. Computer Architecture* (ISCA 06), IEEE CS Press, 2006, pp. 89-101.
2. D. Pham et al., ''The Design and Implementation of a First Generation Cell Processor,'' *Proc. 2005 IEEE Int'l Solid-State Circuit Conf*, IEEE Press, 2005, pp. 184-185, 592.
3. H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, John Wiley & Sons, 2001.
4. *ANSI/IEEE Std 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE, 1999.
5. J.H. Ahn et al., ''Evaluating the Imagine Stream Architecture,'' *Proc. 31st Annual Int'l Symp. Computer Architecture (ISCA 04)*, IEEE CS Press, 2004, pp. 14-25.
6. A. Peleg and U. Weiser, ''MMX Technology Extension to the Intel Architecture,'' *IEEE Micro,* vol. 16, no. 4, 1996, pp. 42-50.
7. A. Waksman, ''A Permutation Network,'' *J. ACM,* vol. 15, no. 1, 1968, pp. 159-163.
8. T. Austin et al., '' Mobile Supercomputers,'' *Computer,* vol. 37, no. 5, 2004, pp. 81-83.
9. J. Glossner, E. Hokenek, and M. Moudgill, ''The Sandbridge Sandblaster Communications Processor,'' *Software Defined Radio*, W.H.W. Tuttlebee, ed., John Wiley & Sons, 2004, pp. 129-159.
10. C. van Berkel et al., ''Vector Processing as an Enabler for Software-Defined Radio in Handsets from 3G+WLAN Onwards,'' *Proc. 2004 Software Defined Radio Tech. Conf*, SDR Forum, 2004, p. B125.

Yuan Lin is a PhD candidate in electrical engineering and computer science at the University of Michigan at Ann Arbor. His research interests include architecture and compiler design for embedded and signal processing applications. Lin has a BS in electrical engineering from Cornell University.

Hyunseok Lee is a PhD candidate in electrical engineering and computer science at the University of Michigan at Ann Arbor. His research interests include digital hardware systems for baseband signal processing and L2/L3 protocols for wireless communication.

Mark Woh is a PhD candidate in electrical engineering and computer science at the University of Michigan Ann Arbor. His research interests include low-power microarchitecture and wireless communication. Woh has a BS in electrical engineering and computer science from the University of Michigan at Ann Arbor.

Yoav Harel is a component design engineer in the Graphics Department at Intel. His research interests include DSP algorithms and low-power architecture. Harel has a MS in electrical engineering and computer science from the University of Michigan at Ann Arbor.

Scott Mahlke is an associate professor in the Electrical Engineering and Computer Science Department at the University of Michigan, where he leads the Compilers Creating Custom Processors group. His research interests include application-specific processors, high-level synthesis, compiler optimization, and computer architecture. Mahlke has a PhD from the University of Illinois, Urbana-Champaign. He is a member of the IEEE and ACM.

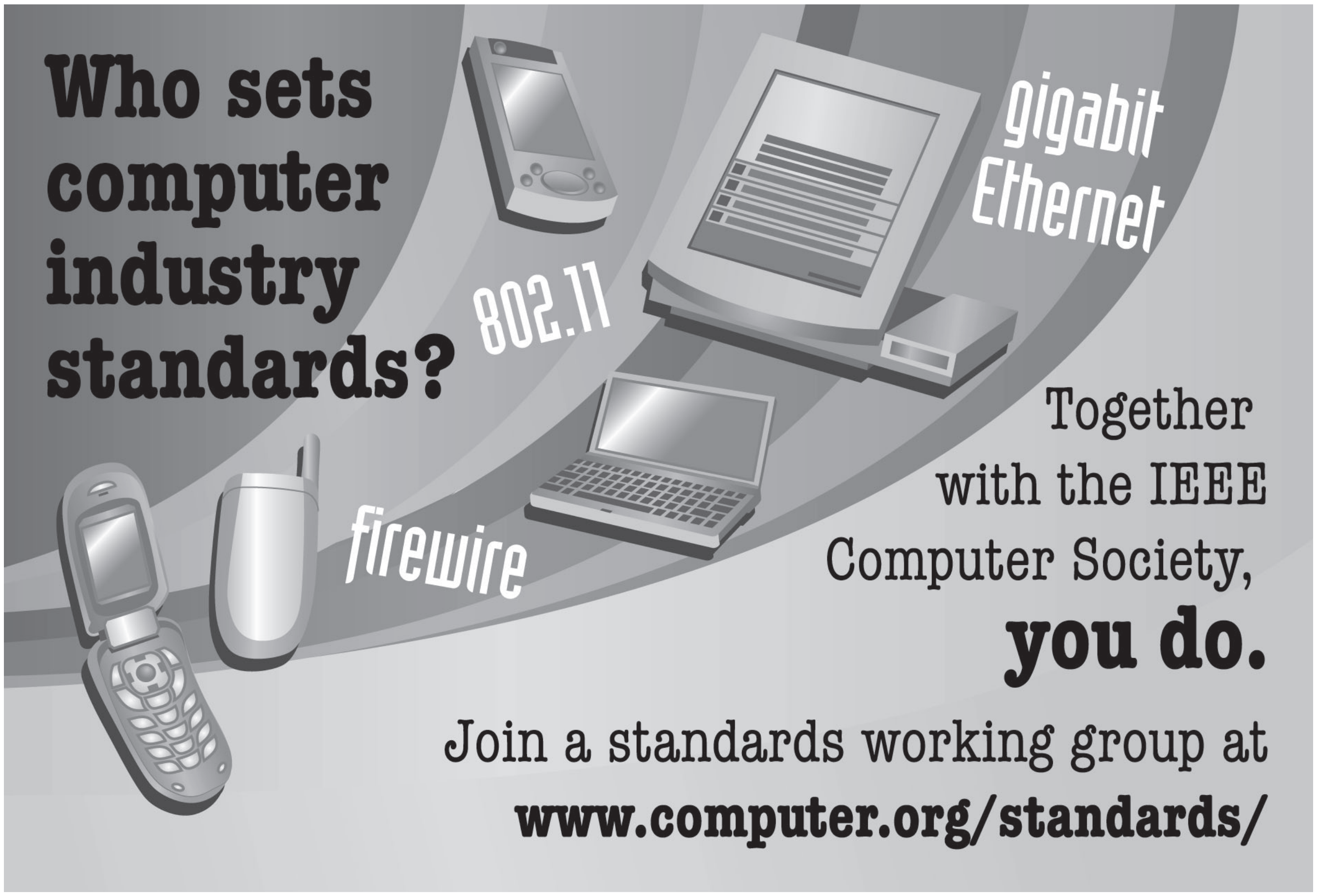
Trevor Mudge is the first Bredt Family Professor of Electrical Engineering and Computer Science at the University of Michigan at Ann Arbor. He also operates Idiot Savants, a chip design consultancy. His research interests include computer architecture, CAD, and compilers. Mudge has a PhD in computer science from the University of Illinois, Urbana-Champaign. He is a member of the ACM, Institution of Engineering and Technology, and British Computer Society and a fellow of the IEEE.

Chaitali Chakrabarti is a professor of electrical engineering at Arizona State University. Her research interests are in VLSI architectures and algorithms for signal processing and communications and low-power system design. She serves as the Technical Committee Chair of Design and Implementation of Signal Processing Systems for the IEEE Signal Processing Society. Chakrabarti received her PhD from the University of Maryland, College Park.

Krisztián Flautner is director of research at ARM Limited. His research interests include high-performance, low-energy processing platforms that support advanced software environments. Flautner has a PhD in electrical engineering and computer science from the University of Michigan at Ann Arbor. He is a member is the ACM and IEEE.

Direct questions and comments about this article to Trevor Mudge, Univ. of Michigan at Ann Arbor, 2260 Hayward, Ann Arbor, MI 48109-2121, tnm@eecs.umich.edu.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.


Who sets computer industry standards?
802.11
gigabit Ethernet
firewire
Together with the IEEE Computer Society, you do.
Join a standards working group at
www.computer.org/standards/