Mobile Supercomputers for the Next-Generation Cell Phone

Mark Woh, Scott Mahlke, and Trevor Mudge, *University of Michigan*
Chaitali Chakrabarti, *Arizona State University*

AnySP demonstrates that power efficiency can be achieved on a fully programmable processor in the context of a future mobile terminal supporting 4G wireless and high-definition video coding.

Mobile devices have proliferated at a spectacular rate, with more than 3.3 billion active cell phones in the world. Soon, improvements to today's smart phones, such as high-bandwidth internet access, high-definition video processing, and interactive video conferencing will be commonplace.

The International Telecommunications Union has proposed fourth-generation (4G) wireless technology to increase bandwidth to maximum data rates of 100 Mbps for high-mobility situations and 1 Gbps for stationary and low-mobility scenarios like Internet hot spots (www.ieee802.org/secmail/pdf00204.pdf). This translates into an increase in computational requirements of 10 to 1,000 times over previous third-generation (3G) wireless technologies, with a power budget of approximately 1W for all the computation. Other forms of signal processing, such as high-definition video, are also up to 100 times more compute-intensive than current mobile video. Figure 1 shows the peak processing throughputs and power budgets of 3G and 4G protocols. Conventional processors cannot meet these protocols' power-throughput requirements.

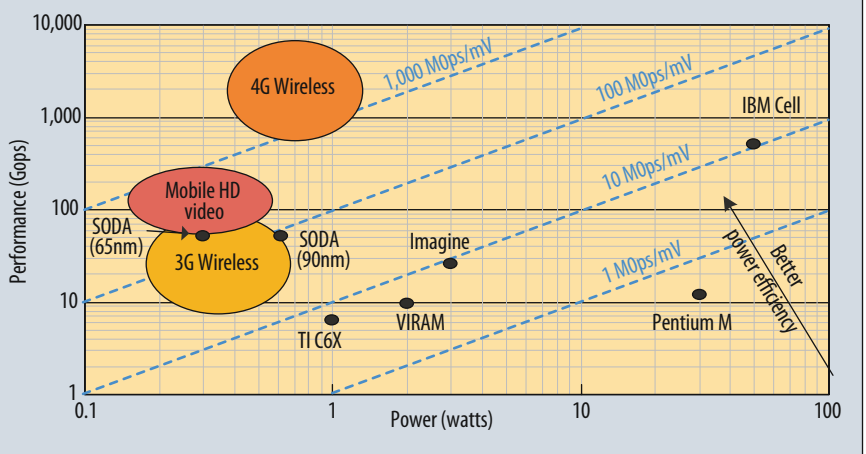

Figure 1. Peak processing throughputs and power budgets of 3G and 4G protocols. Conventional processors cannot meet these power-throughput requirements.

Research solutions, such as VIRAM and Imagine, can achieve the performance requirements for 3G, but generally exceed the power budgets of mobile terminals. The signal-processing on demand architecture (SODA) improved upon these solutions and could meet both the power and throughput requirements for 3G wireless (Y. Lin et al., "SODA: A Low-Power Architecture for Software Radio," *Proc. 33rd Ann. Int'l Symp. Computer Architecture*, 2006, pp. 89-101).

For 4G wireless protocols, the computational efficiency of mobile computer systems must be increased to greater than 1,000 Mops/mW. 4G uses three central technologies: orthogonal frequency division multiplexing (OFDM), low-density parity check (LDPC) code, and multiple-input multiple-output (MIMO) techniques.

0018-9162/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

Fast Fourier transforms (FFTs) are key because they route signals from the baseband to the subcarriers. LDPC codes provide superior error-correction capabilities, however, parallelizing the LDPC decoding algorithm is more challenging because of the large amount of data shuffling. MIMO is based on the use of multiple antennae for both transmission and signal reception and requires complex signal-detection algorithms. The need for higher bandwidth and increased computational complexity are the main reasons for the two-orders-of-magnitude increase in processing requirements when moving from 3G to 4G.

Next-generation mobile computer system designs must address three issues: efficiency, programmability, and adaptivity.

High-definition video is also an important application that these platforms must support. Figure 1 shows that the performance requirements of video exceed those of 3G wireless, but are less than those for 4G wireless. However, the power budget dedicated to video is generally smaller. Moreover, the data access complexity in video is much higher than wireless, since algorithms operate on two- or three-dimensional blocks of data. Thus, video applications push designs to have more flexible, higher-bandwidth memory systems. High-definition video is just one example of a growing class of applications with diverse computing and memory requirements that next-generation mobile devices must support.

NEXT-GENERATION DESIGN STRATEGIES

3G mobile computer systems employ a combination of general-purpose processors, digital-signal processors, and hardwired accelerators to provide the giga-operations-per-second performance on milliWatt power budgets that today's cell phones require. However, such heterogeneous organizations are inefficient for companies to use in developing, building, and maintaining software. Further, as the amount of functionality integrated onto their mobile terminal increases, hardwired solutions waste silicon area and power with many single-use hardware blocks.

Next-generation mobile computer system designs must address three issues: efficiency, programmability, and adaptivity. The existing computational efficiency of 3G solutions is inadequate and must be increased for 4G. As a result, straightforward scaling of 3G solutions by increasing cores or data-level parallelism is inadequate. Programmability provides the opportunity for a single platform to support multiple applications and even multiple standards within each application domain. Last, hardware adaptivity is necessary to maintain efficiency as the core computational characteristic of the applications change.

3G solutions rely heavily on the widespread amounts of vector parallelism in wireless signal processing algorithms, but lose most of their efficiency when vector parallelism is unavailable or constrained, as happens with high-definition video.

Designing efficient architectures for future mobile systems requires analyzing the anticipated workloads. While performing detailed analysis of the computation kernels for 4G wireless and high-definition video encoding and decoding (h.264), we elicited five key insights:

- Opportunities for single-instruction, multiple data (SIMD) parallelism vary widely across the algorithms. Some have large inherent vectors, up to 1,024 elements in length. However, most algorithms have small to moderate vectors.
- Algorithms with smaller vector lengths frequently contain a high degree of identical threads, where each thread performs the same instructions, but on discontinuous data. A large percentage of temporary values generated during the computation are short-lived and need not be saved to a register file.
- There is a small set of arithmetic instruction pairs that occur with high frequency.
- Each algorithm repeatedly uses a small set of predetermined data-shuffling patterns.

ANYSP MOBILE SUPERCOMPUTER

To address these challenges, we highlight the AnySP advanced-signal-processing architecture proposed by researchers at the University of Michigan, Arizona State University, and ARM Limited (M. Woh et al., "AnySP: Anytime Anywhere Anyway Signal Processing," *Proc. 36th Ann. Int'l Symp. Computer Architecture, 2009,* pp. 128–139). AnySP seeks to create a fully programmable architecture that supports 4G wireless communication and high-definition video decoding.

Such a design would need to reach the computation efficiency levels of nearly 1,000 Mops/mW that only ASIC solutions have achieved previously. Programmability is recognized as a first-class design constraint, thus no fixed-function hardware blocks are employed.

To overcome the typical pitfalls of relying on SIMD parallelism across a wide variety of algorithms, a configurable SIMD datapath is created. This supports three execution scenarios: wide vector computation (64 lanes), multiple independent narrow vector computation threads and chained computation subgraphs on moder-

ately wide vector computation. This inherent flexibility lets the datapath be customized to the application while still retaining the high execution efficiency that SIMD offers by reducing control overhead. AnySP also attacks the traditional inefficiencies of SIMD computation: register file power, data shuffling, and reduction operators.

Figure 2 shows the AnySP processing element (PE) architecture, which consists of integrated SIMD and scalar datapaths. The SIMD datapath in turn consists of eight groups of 8-wide SIMD units, which can be configured to create SIMD widths of 16, 32, and 64. Each of the 8-wide SIMD units comprises groups of flexible functional units (FFU). The FFUs contain the functional units of two lanes connected through a simple cross bar. Eight SIMD register files (RFs) feed the SIMD datapath and each 8-wide RF has 16 entries. The data shuffle—or swizzle—network aligns data for the FFUs. It can support a fixed number of swizzle patterns of 8-, 16-, 32-, 64-, and 128-wide elements. Finally, a multiple output adder tree can sum groups of 4, 8, 16, 32, or 64 elements, then store the results in a temporary buffer.

The local memory consists of 16 memory banks. Each bank is an 8-wide SIMD containing 256 16-bit entries, totaling 32 Kbytes of storage. Each 8-wide SIMD group has a dedicated address generation unit (AGU). When not in use, the AGU can run sequential code to assist the dedicated scalar pipeline. The AGU and scalar unit share the same memory space as the SIMD datapath. To accomplish this, the design includes a scalar memory buffer that can store 8-wide SIMD locations. Because many of the algorithms access data sequentially, the buffer acts as a small cache that helps to avoid multiple vector-bank accesses.

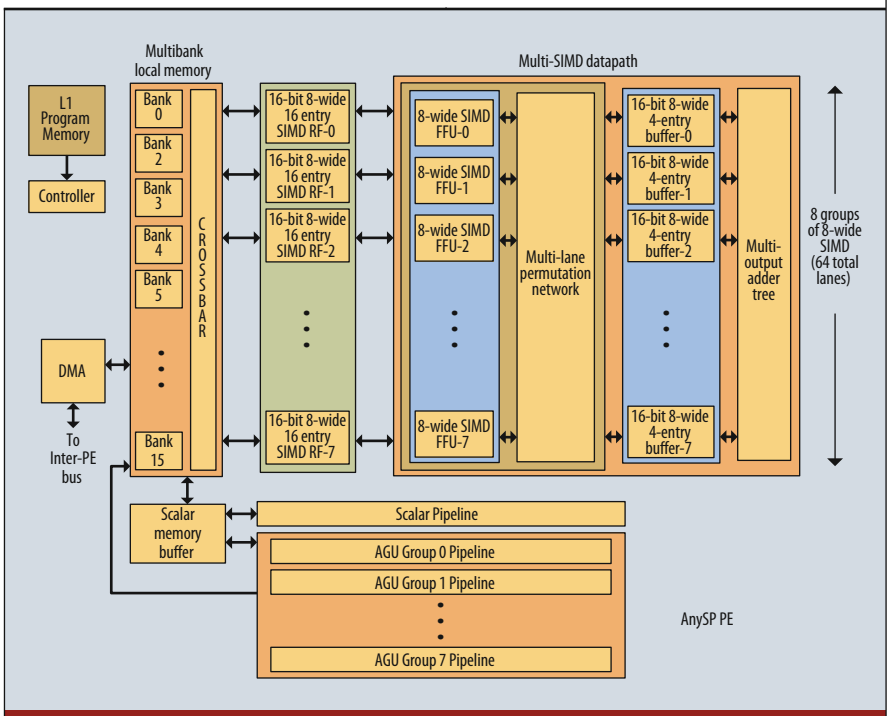

Figure 2. AnySP processing element. This processing element architecture consists of integrated SIMD and scalar datapaths.

CONFIGURABLE MULTI-SIMD WIDTH SUPPORT

The individual algorithms in the applications that we studied had varying SIMD widths. However, independent threads were not dominant. Rather, the system would run the same task many times for different sets of data. Each task was independent of others, running the exact same code and following almost the same control path with the only difference being the set of memory addresses accessed.

To support these types of kernel algorithms, AnySP was designed as a multi-SIMD-width architecture. Each group of 8-wide SIMD units has its own AGU to access a different data stream. The 8-wide groups can also be coalesced to create SIMD widths of 16, 32, or 64. This feature lets the system exploit data and thread parallelism together for large and small SIMD-width algorithms.

Small SIMD-width algorithms like intraprediction and motion compensation from h.264 video decoding can process multiple macroblocks at the same time while exploiting the 8-wide and 16-wide SIMD parallelism within the algorithms. Meanwhile, large SIMD-width algorithms like FFT and LDPC can use the full 64-width SIMD.

TEMPORARY BUFFER AND BYPASS NETWORK

AnySP implements temporary register buffers and a bypass network to reduce power consumption and the number of RF accesses. The temporary register buffers are implemented as a partitioned RF. The main RF contains 16 registers, but the design also adds a second partition containing four registers, making 20 registers total. This small, partitioned RF shields the main RF from accesses by storing values that have short lifetimes.

The bypass network is a modification to the writeback stage and forwarding logic. Typically, in processors, data forwarded to eliminate data hazards is also written back to the RF. In the bypass network, the compiler explicitly manages the forwarding and writing to the RF to eliminate unnecessary RF writes.

FLEXIBLE FUNCTIONAL UNITS

In typical SIMD architectures, power and performance are lost

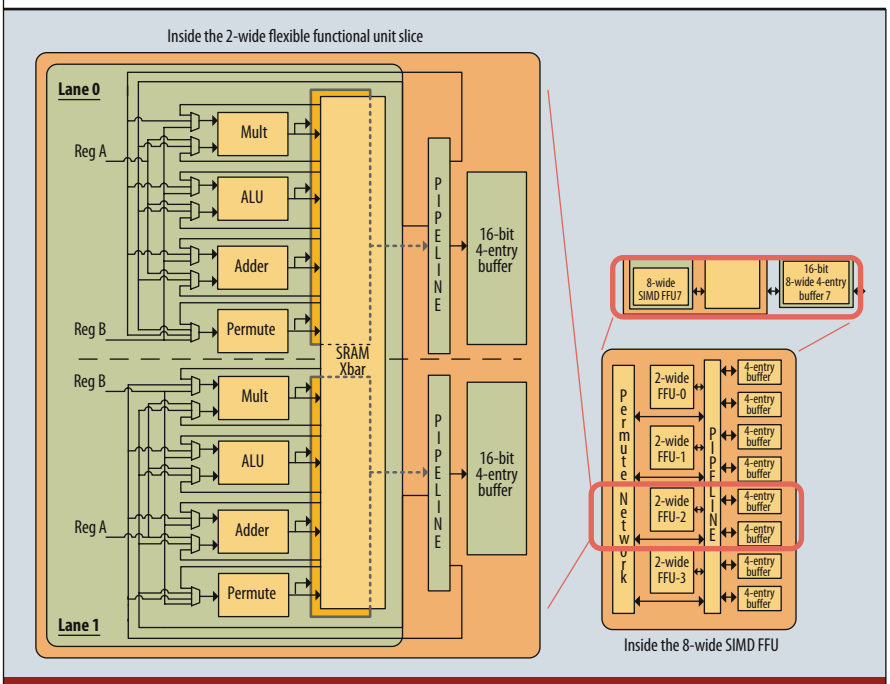

Figure 3. Design of a flexible functional unit that supports chaining of neighboring lanes to pipeline the execution of 2-deep computation subgraphs.

when the vector size is smaller than the SIMD width as a result of underutilized hardware. AnySP adds another level of configurability to the datapath by using FFUs as the core computation units. When SIMD utilization is low, the FFUs can chain back-to-back the FFUs between neighboring lanes. This effectively turns two SIMD lanes into a 2-deep execution pipeline. Two different instructions can be chained through the pipeline, and data is passed between them without writing back to the RF.

As Figure 3 shows, each 8-wide SIMD group is built from four 2-wide FFUs. Algorithms with SIMD widths smaller than 64 benefit from this structure. In chained-execution mode, the functional units among two internal lanes can be connected through a crossbar network. Overall, FFUs improve performance and reduce power by adding more flexibility. AnySP only chains pairs of lanes together, but the technique can be expanded to chain larger numbers of lanes.

SWIZZLE NETWORK

The number of distinct swizzle patterns needed for a specific algorithm is small, fixed, and known in advance. Previous research has explored building application-specific crossbars for SIMD processors (P. Raghavan et al., "A Customized Crossbar for Data-Shuffling in Domain-Specific SIMD Processors," *Proc. Architectures for Computing Systems* (ARCS 2007), LNCS 4415, Springer, 2007, pp. 57-68), but these lack flexibility because they cannot support new swizzle operations for applications that emerge postfabrication. AnySP proposes using an SRAM-based swizzle network that adds flexibility while maintaining the performance of a customized crossbar. The proposed network is similar to the work of N. Goel, A. Kumar, and P.R. Panda ("Power Reduction in VLIW Processor with Compiler Driven Bypass Network, *Proc. 20th Int'l Conf. VLSI Design* (VLSID 07), ACM Press, 2007, pp. 233-238) in that the X-Y style crossbar lays out the input buses horizontally and the outputs vertically.

Each point of intersection between the input and output buses contains a pass transistor controlled by a flip-flop. Multiple sets of swizzle configurations are stored in the SRAM cells, allowing zero-cycle delay for changing the swizzle pattern. By storing multiple configurations, many control wires can be removed, and the network's area and power consumption can be reduced while still operating within a single clock cycle. For crossbar sizes larger than 32 × 32, power is dramatically lower than the MUX-based alternative and can run at almost twice the frequency. For example, a 128 × 128 SRAM-based swizzle network consumes less than 30 percent of the power consumed by an equivalent MUX-based crossbar.

Though only a certain number of swizzle patterns can be loaded at a time without reconfiguration, this approach provides a viable solution because only a small set of swizzle patterns are needed for each algorithm. The swizzle network has lower power and provides more functionality than the permutation networks found in typical SIMD architectures by also supporting multicasting capabilities along with the swizzle patterns.

MULTIPLE OUTPUT ADDER TREE SUPPORT

Many SIMD architectures have special SIMD summation hardware to perform reduction-to-scalar operations. To compute this, adder trees sum up the values of all lanes and store the result in the scalar RF. While this worked for 3G algorithms, many of the video decoding algorithms needed sums shorter than the SIMD width. In the AnySP architecture, the adder tree allows for partial summations of 4 through 64 elements, which are then written back to the temporary buffer unit.

A 4-PE AnySP system running at 300 MHz with an ARM Cortex-M3 serving as the control processor met the throughput requirements of 100 Mbps 4G wireless while consuming 1.3 W at

90 nm. This falls short of the 1,000 Mops/mW efficiency target, but close enough to meet it in 45-nm process technology. H.264 video decoding at 30 fps is achieved with 60 mW at 90 nm, meeting the requirements for mobile HD video. The power breakdown of AnySP shows that the SIMD functional units dominate power consumption, followed by the register file and rest of the datapath. AnySP was designed to demonstrate that power efficiency can be achieved on a fully programmable processor in the context of a future mobile terminal supporting 4G wireless and high-definition video coding. Programmability is essential moving forward to provide a hardware substrate that allows the software to evolve naturally.

AnySP features a configurable SIMD datapath that supports wide and narrow vector lengths; flexible functional units, which can chain together narrow SIMD instructions using neighboring SIMD lanes; temporary buffers and a bypass network that reduce register and memory accesses; an SRAM-based swizzle network that reduces the power and latency of data shuffling operations; and a flexible multiple-output adder tree, which speeds up video applications.

Industry will continue to build heterogeneous systems consisting of programmable processors and hardwired ASICs, but it is already trying to reduce the number of distinct intellectual property blocks in designs to reduce cost and manage complexity. We expect features such as those in AnySP to slowly integrate into mainstream mobile architectures, achieving a fully programmable, mobile supercomputer. C

Mark Woh *is a PhD candidate in the Department of Electrical Engineering and Computer Science at the University of Michigan. Contact him at mwoh@umich.edu.*

Scott Mahlke *is an associate professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. Contact him at mahlke@umich.edu.*

Trevor Mudge *is Bredt Family Professor of Engineering in the Department of Electrical Engineering and Computer Science at the University of Michigan. Contact him at tnm@umich.edu.*

Chaitali Chakrabarti *is a professor of electrical engineering at Arizona State University. Contact her at chaitali@asu.edu.*

Editor: Tom Conte, College of Computing, Georgia Institute of Technology; conte@cc.gatech.edu