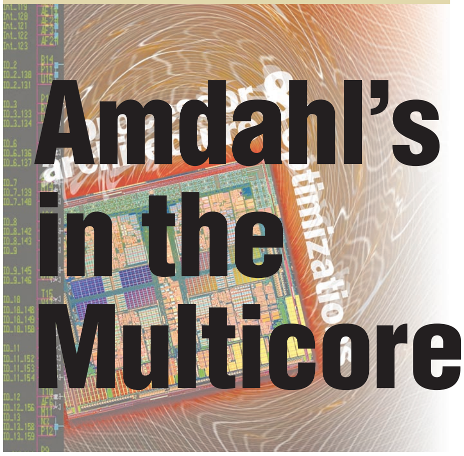
COVER FEATURE



Amdahl's Law in the Multicore Era

Mark D. Hill, University of Wisconsin-Madison
Michael R. Marty, Google

Augmenting Amdahl's law with a corollary for multicore hardware makes it relevant to future generations of chips with multiple processor cores. Obtaining optimal multicore performance will require further research in both extracting more parallelism and making sequential cores faster.

As we enter the multicore era, we're at an inflection point in the computing landscape. Computing vendors have announced chips with multiple processor cores. Moreover, vendor road maps promise to repeatedly double the number of cores per chip. These future chips are variously called chip multiprocessors, multicore chips, and many-core chips.

Designers must subdue more degrees of freedom for multicore chips than for single-core designs. They must address such questions as: How many cores? Should cores use simple pipelines or powerful multi-issue pipeline designs? Should cores use the same or different microarchitectures? In addition, designers must concurrently manage power from both dynamic and static sources.

Although answering these questions for today's multicore chip with two to eight cores is challenging now, it will become much more challenging in the future. Sources as varied as Intel and the University of California, Berkeley, predict a hundred,[1] if not a thousand,[2] cores.

As the "Amdahl's Law" sidebar describes, this model has important consequences for the multicore era. To complement Amdahl's software model, we offer a corollary of a simple model of multicore hardware resources. Our results should encourage multicore designers to view the entire chip's performance rather than focusing on core efficiencies. We also discuss several important limitations of our models to stimulate discussion and future work.

A COROLLARY FOR MULTICORE CHIP COST

To apply Amdahl's law to a multicore chip, we need a cost model for the number and performance of cores that the chip can support.

We first assume that a multicore chip of given size and technology generation can contain at most n base core equivalents, where a single BCE implements the baseline core. This limit comes from the resources a chip designer is willing to devote to processor cores (with L1 caches). It doesn't include chip resources expended on shared caches, interconnection networks, memory controllers, and so on. Rather, we simplistically assume that these nonprocessor resources are roughly constant in the multicore variations we consider.

We are agnostic on what limits a chip to n BCEs. It might be power, area, or some combination of power, area, and other factors.

Second, we assume that (micro-) architects have techniques for using the resources of multiple BCEs to create a core with greater sequential performance. Let the performance of a single-BCE core be 1. We assume that architects can expend the resources of r BCEs to create a powerful core with sequential performance $perf(r)$.

Architects should always increase core resources when $perf(r) > r$ because doing so speeds up both sequential and parallel execution. When $perf(r) < r$, however, the tradeoff begins. Increasing core performance aids sequential execution, but hurts parallel execution.

0018-9162/08/$25.00 © 2008 IEEE

Published by the IEEE Computer Society

Amdahl's Law

> Everyone knows Amdahl's law, but quickly forgets it.
> —Thomas Puzak, IBM, 2007

Most computer scientists learn Amdahl's law in school: Let speedup be the original execution time divided by an enhanced execution time.[1] The modern version of Amdahl's law states that if you enhance a fraction *f* of a computation by a speedup *S*, the overall speedup is:

$$\text{Speedup}_{\text{enhanced}}(f,S) = \frac{1}{(1-f) + \frac{f}{S}}$$

Amdahl's law applies broadly and has important corollaries such as:

- Attack the common case: When *f* is small, optimizations will have little effect.
- The aspects you ignore also limit speedup: As *S* approaches infinity, speedup is bound by 1/(1 – *f*).

Four decades ago, Gene Amdahl defined his law for the special case of using *n* processors (cores) in parallel when he argued for the single-processor approach's validity for achieving large-scale computing capabilities.[1] He used a limit argument to assume that a fraction *f* of a program's execution time was infinitely parallelizable with no scheduling overhead, while the remaining fraction, 1 – *f*, was totally sequential. Without presenting an equation, he noted that the speedup on *n* processors is governed by:

$$\text{Speedup}_{\text{parallel}}(f,n) = \frac{1}{(1-f) + \frac{f}{n}}$$

Finally, Amdahl argued that typical values of 1 – *f* were large enough to favor single processors.

Despite their simplicity, Amdahl's arguments held, and mainframes with one or a few processors dominated the computing landscape. They also largely held in the minicomputer and personal computer eras that followed. As recent technology trends usher us into the multicore era, Amdahl's law is still relevant.

Amdahl's equations assume, however, that the computation problem size doesn't change when running on enhanced machines. That is, the fraction of a program that is parallelizable remains fixed. John Gustafson argued that Amdahl's law doesn't do justice to massively parallel machines because they allow computations previously intractable in the given time constraints.[2] A machine with greater parallel computation ability lets computations operate on larger data sets in the same amount of time. When Gustafson's arguments apply, parallelism will be ample. In our view, however, robust general-purpose multicore designs should also operate well under Amdahl's more pessimistic assumptions.

References

1. G.M. Amdahl, "Validity of the Single-Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. Am. Federation of Information Processing Societies Conf.*, AFIPS Press, 1967, pp. 483-485.
2. J.L. Gustafson, "Reevaluating Amdahl's Law," *Comm. ACM*, May 1988, pp. 532-533.

Our equations allow *perf*(*r*) to be an arbitrary function, but all our graphs follow Shekhar Borkar[3] and assume $perf(r) = \sqrt{r}$. In other words, we assume efforts that devote *r* BCE resources will result in sequential performance $\sqrt{r}$. Thus, architectures can double performance at a cost of four BCEs, triple it for nine BCEs, and so on. We tried other similar functions (for example, $\sqrt[1.5]{r}$), but found no important changes to our results.

SYMMETRIC MULTICORE CHIPS

A symmetric multicore chip requires that all its cores have the same cost. A symmetric multicore chip with a resource budget of *n* = 16 BCEs, for example, can support 16 cores of one BCE each, four cores of four BCEs each, or, in general, *n*/*r* cores of *r* BCEs each (our equations and graphs use a continuous approximation instead of rounding down to an integer number of cores). Figures 1a and 1b show two hypothetical symmetric multicore chips for *n* = 16.

Under Amdahl's law, the speedup of a symmetric multicore chip (relative to using one single-BCE core) depends on the software fraction that is parallelizable (*f*), the total chip resources in BCEs (*n*), and the BCE resources (*r*) devoted to increase each core's performance. The chip uses one core to execute sequentially at performance *perf*(*r*). It uses all *n*/*r* cores to execute in parallel at performance *perf*(*r*) × *n*/*r*. Overall, we get:

$$\text{Speedup}_{\text{symmetric}}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}$$

Consider Figure 2a. It assumes a symmetric multicore chip of *n* = 16 BCEs and $perf(r) = \sqrt{r}$. The *x*-axis

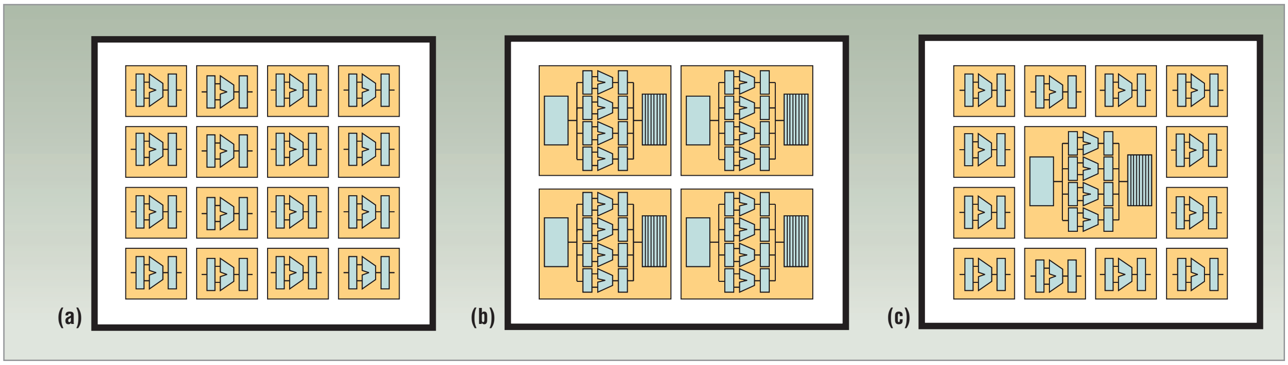


Figure 1. Varieties of multicore chips. (a) Symmetric multicore with 16 one-base core equivalent cores, (b) symmetric multicore with four four-BCE cores, and (c) asymmetric multicore with one four-BCE core and 12 one-BCE cores. These figures omit important structures such as memory interfaces, shared caches, and interconnects, and assume that area, not power, is a chip's limiting resource.

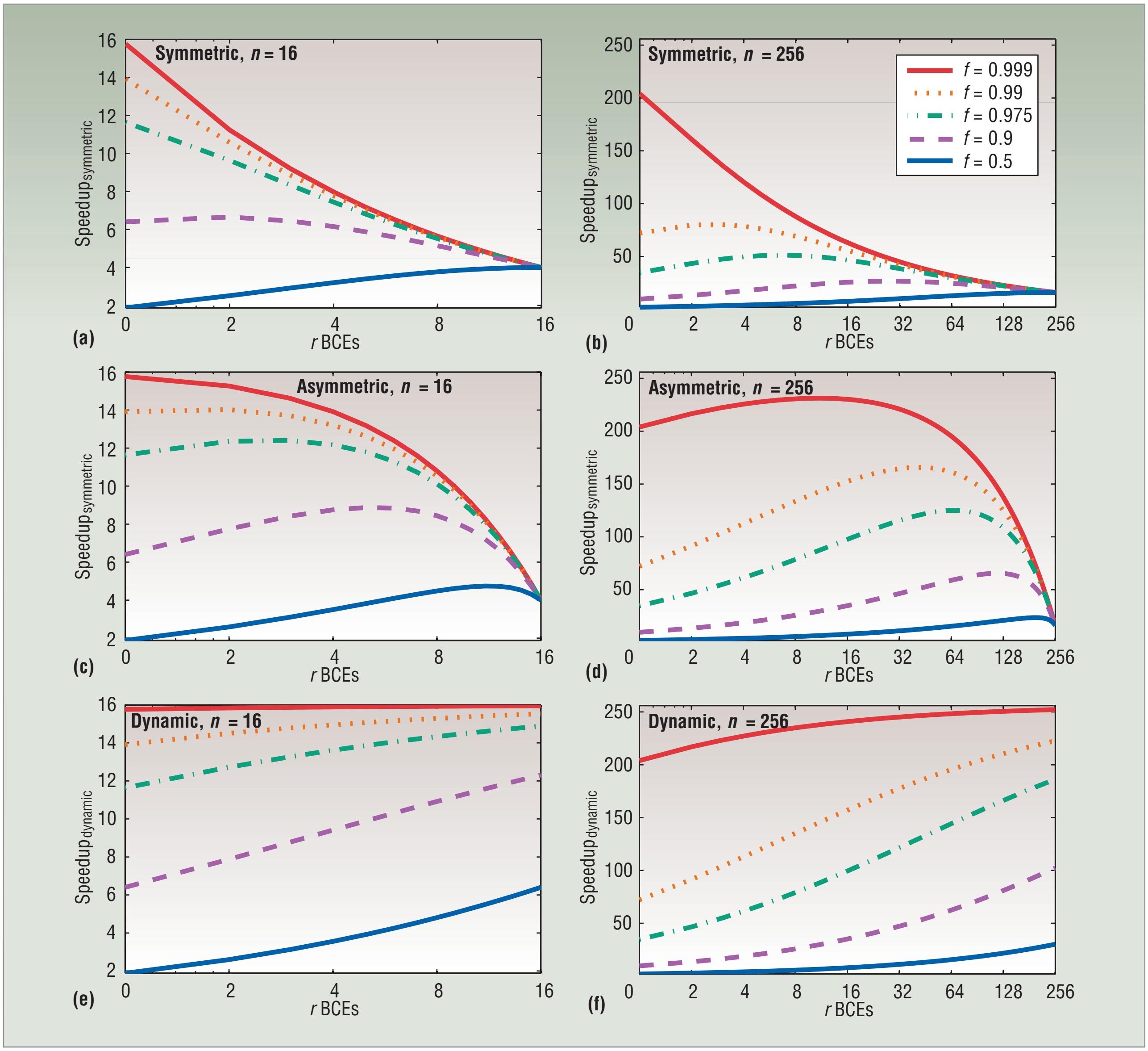


Figure 2. Speedup of (a, b) symmetric, (c, d) asymmetric, and (e, f) dynamic multicore chips with n = *16 BCEs (a, c, and e) or* n = *256 BCEs (b, d, and f).*

gives resources used to increase each core's performance: a value 1 says the chip has 16 base cores, while a value of $r = 16$ uses all resources for a single core. Lines assume different values for the parallel fraction ($f = 0.5, 0.9, \ldots, 0.999$). The y-axis gives the symmetric multicore chip's speedup relative to its running on one single-BCE base core. The maximum speedup for $f = 0.9$, for example, is 6.7 using eight cores at a cost of two BCEs each.

Similarly, Figure 2b illustrates how tradeoffs change when Moore's law allows $n = 256$ BCEs per chip. With $f = 0.975$, for example, the maximum speedup of 51.2 occurs with 36 cores of 7.1 BCEs each.

Result 1. Amdahl's law applies to multicore chips because achieving the best speedups requires fs that are near 1. Thus, finding parallelism is still critical.

Implication 1. Researchers should target increasing f through architectural support, compiler techniques, programming model improvements, and so on.

This implication is the most obvious and important. Recall, however, that a system is cost-effective if speedup exceeds its costup.[4] Multicore costup is the multicore system cost divided by the single-core system cost. Because this costup is often much less than n, speedups less than n can be cost-effective.

Result 2. Using more BCEs per core, $r > 1$, can be optimal, even when performance grows by only $\sqrt{r}$. For a given f, the maximum speedup can occur at one big core, n base cores, or with an intermediate number of middle-sized cores. Recall that for $n = 256$ and $f = 0.975$, the maximum speedup occurs using 7.1 BCEs per core.

Implication 2. Researchers should seek methods of increasing core performance even at a high cost.

Result 3. Moving to denser chips increases the likelihood that cores will be nonminimal. Even at $f = 0.99$, minimal base cores are optimal at chip size $n = 16$, but more powerful cores help at $n = 256$.

Implication 3. As Moore's law leads to larger multicore chips, researchers should look for ways to design more powerful cores.

ASYMMETRIC MULTICORE CHIPS

An alternative to a symmetric multicore chip is an asymmetric (or heterogeneous) multicore chip, in which one or more cores are more powerful than the others.[5-8] With the simplistic assumptions of Amdahl's law, it makes most sense to devote extra resources to increase only one core's capability, as Figure 1c shows.

With a resource budget of $n = 16$ BCEs, for example, an asymmetric multicore chip can have one four-BCE core and 12 one-BCE cores, one nine-BCE core and seven one-BCE cores, and so on. In general, the chip can have $1 + n - r$ cores because the single larger core uses r resources and leaves $n - r$ resources for the one-BCE cores.

Amdahl's law has a different effect on an asymmetric multicore chip. This chip uses the one core with more resources to execute sequentially at performance $perf(r)$. In the parallel fraction, however, it gets performance $perf(r)$ from the large core and performance 1 from each of the $n - r$ base cores. Overall, we get:

$$\text{Speedup}_{\text{asymmetric}}(f, n, r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}}$$

Figure 2c shows asymmetric speedup curves for $n = 16$ BCEs, while Figure 2d gives curves for $n = 256$ BCEs. These curves are markedly different from the corresponding symmetric speedups in Figures 2a and 2b. The symmetric curves typically show either immediate performance improvement or performance loss as the chip uses more powerful cores, depending on the level of parallelism. In contrast, asymmetric chips often reach a maximum speedup between the extremes.

Result 4. Asymmetric multicore chips can offer potential speedups that are much greater than symmetric multicore chips (and never worse). For $f = 0.975$ and $n = 256$, for example, the best asymmetric speedup is 125.0, whereas the best symmetric speedup is 51.2.

Implication 4. Researchers should continue to investigate asymmetric multicore chips, including dealing with the scheduling and overhead challenges that Amdahl's model doesn't capture.

Result 5. Denser multicore chips increase both the speedup benefit of going asymmetric and the optimal performance of the single large core. For $f = 0.975$ and $n = 1{,}024$, an example not shown in our graphs, the best speedup is at a hypothetical design with one core of 345 BCEs and 679 single-BCE cores.

Implication 5. Researchers should investigate methods of speeding sequential performance even if they appear locally inefficient—for example, $perf(r) = \sqrt{r}$. This is because these methods can be globally efficient as they reduce the sequential phase when the chip's other $n - r$ cores are idle.

DYNAMIC MULTICORE CHIPS

What if architects could have their cake and eat it too? Consider dynamically combining up to r cores to boost performance of only the sequential component, as Figure 3 shows. This could be possible with, for example, thread-level speculation or helper threads.[9-12] In sequential mode, this dynamic multicore chip can execute with performance $perf(r)$ when the dynamic techniques can use r BCEs. In parallel mode, a dynamic multicore gets performance n using all base cores in parallel. Overall, we get:

$$\text{Speedup}_{\text{dynamic}}(f, n, r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$

Figure 2e displays dynamic speedups when using r cores in sequential mode for $perf(r) = \sqrt{r}$ for $n = 16$

BCEs, while Figure 2f gives curves for $n = 256$ BCEs. As the graphs show, performance always gets better as the software can exploit more BCE resources to improve the sequential component. Practical considerations, however, might keep r much smaller than its maximum of n.

Result 6. Dynamic multicore chips can offer speedups that can be greater (and are never worse) than asymmetric chips with identical $perf(r)$ functions. With Amdahl's sequential-parallel assumption, however, achieving much greater speedup than asymmetric chips requires dynamic techniques that harness more cores for sequential mode than is possible today. For $f = 0.99$ and $n = 256$, for example, effectively harnessing all 256 cores would achieve a speedup of 223, which is much greater than the comparable asymmetric speedup of 165. This result follows because we assume that dynamic chips can both gang all resources together for sequential execution and free them for parallel execution.

Implication 6. Researchers should continue to investigate methods that approximate a dynamic multicore chip, such as thread-level speculation and helper threads. Even if the methods appear locally inefficient, as with asymmetric chips, the methods can be globally efficient. Although these methods can be difficult to apply under Amdahl's extreme assumptions, they could flourish for software with substantial phases of intermediate-level parallelism.

SIMPLE AS POSSIBLE, BUT NO SIMPLER

Amdahl's law and the corollary we offer for multicore hardware seek to provide insight to stimulate discussion and future work. Nevertheless, our specific quantitative results are suspect because the real world is much more complex.

Currently, hardware designers can't build cores that achieve arbitrary high performance by adding more resources, nor do they know how to dynamically harness many cores for sequential use without undue performance and hardware resource overhead. Moreover, our models ignore important effects of dynamic and static power, as well as on- and off-chip memory system and interconnect design.

Software is not just infinitely parallel and sequential. Software tasks and data movements add overhead. It's more costly to develop parallel software than sequential software. Furthermore, scheduling software tasks on asymmetric and dynamic multicore chips could be difficult and add overhead. To this end, Tomer Morad and his colleagues[13] and JoAnn Paul and Brett Meyer[14] developed sophisticated models that question the validity of Amdhal's law to future systems, especially embedded ones. On the other hand, more cores might advantageously allow greater parallelism from larger problem sizes, as John Gustafson envisioned.[15]

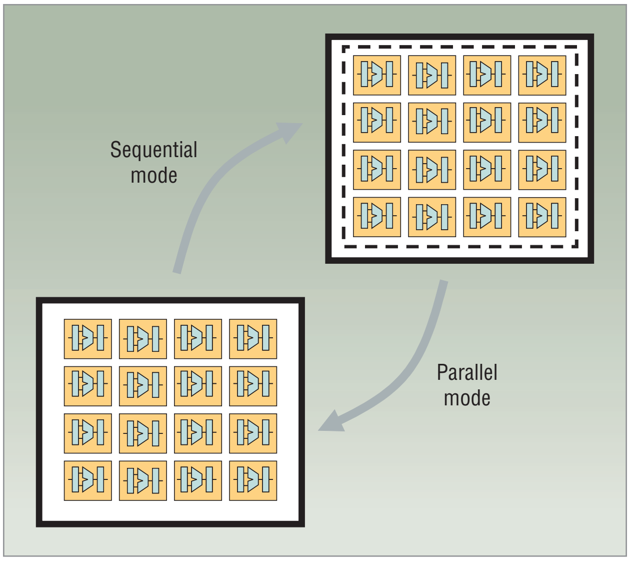


Figure 3. Dynamic multicore chip with 16 one-BCE cores.

Pessimists will bemoan our model's simplicity and lament that much of the design space we explore can't be built with known techniques. We charge you, the reader, to develop better models, and, more importantly, to invent new software and hardware designs that realize the speedup potentials this article displays. Moreover, research leaders should temper the current pendulum swing from the past's underemphasis on parallel research to a future with too little sequential research. To help you get started, we provide slides from a keynote talk as well as the code examples for this article's models at www.cs.wisc.edu/multifacet/amdahl. ■

Acknowledgments

We thank Shailender Chaudhry, Robert Cypher, Anders Landin, José F. Martínez, Kevin Moore, Andy Phelps, Thomas Puzak, Partha Ranganathan, Karu Sankaralingam, Mike Swift, Marc Tremblay, Sam Williams, David Wood, and the Wisconsin Multifacet group for their comments or proofreading. The US National Science Foundation supported this work in part through grants EIA/CNS-0205286, CCR-0324878, CNS-0551401, CNS-0720565, and CNS-0720565. Donations from Intel and Sun Microsystems also helped fund the work. Mark Hill has significant financial interest in Sun Microsystems. The views expressed herein aren't necessarily those of the NSF, Intel, Google, or Sun Microsystems.

References

1. "From a Few Cores to Many: A Tera-scale Computing Research Overview," white paper, Intel, 2006; ftp://download.intel.com/research/platform/terascale/terascale_overview_paper.pdf.

2. K. Asanovic et al., *The Landscape of Parallel Computing Research: A View from Berkeley*, tech. report UCB/EECS-2006-183, Dept. Electrical Eng. and Computer Science, Univ. of Calif., Berkeley, 2006.
3. S. Borkar, "Thousand Core Chips—A Technology Perspective," *Proc. ACM/IEEE 44th Design Automation Conf.* (DAC), ACM Press, 2007, pp. 746-749.
4. D.A. Wood and M.D. Hill, "Cost-Effective Parallel Computing," *Computer*, Feb. 1995, pp. 69-72.
5. S. Balakrishnan et al., "The Impact of Performance Asymmetry in Emerging Multicore Architectures," *Proc. 32nd Ann. Int'l Symp. Computer Architecture*, ACM Press, 2005, pp. 506-517.
6. J.A. Kahl et al., "Introduction to the Cell Multiprocessor," *IBM J. Research and Development*, vol. 49, no. 4, 2005, pp. 589-604.
7. R. Kumar et al., "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, IEEE CS Press, 2003, pp. 81-92.
8. M.A. Suleman et al., *ACMP: Balancing Hardware Efficiency and Programmer Efficiency*, HPS tech. report, TRHPS-2007-001, Univ. of Texas, Austin, 2007.
9. L. Hammond, M. Willey, and K. Olukotun, "Data Speculation Support for a Chip Multiprocessor," *Proc. 8th Int'l Conf. Architectural Support for Programming Languages and Operating Systems,* ACM Press, 1998, pp. 58-69.
10. E. Ipek et al., "Core Fusion: Accommodating Software Diversity in Chip Multiprocessors," *Proc. 34th Ann. Int'l Symp. Computer Architecture*, ACM Press, 2007, pp. 186-197.
11. J. Renau et al., "Energy-Efficient Thread-Level Speculation on a CMP," *IEEE Micro*, Jan./Feb. 2006, pp. 80-91.
12. G.S. Sohi, S. Breach, and T.N. Vijaykumar, "Multiscalar Processors," *Proc. 22nd Ann. Int'l Symp. Computer Architecture*, ACM Press, 1995, pp. 414-425.
13. T. Morad et al., "Performance, Power Efficiency, and Scalability of Asymetric Cluster Chip Multiprocessors," *Computer Architecture Letters*, vol. 4, July 2005; www.ee.technion.ac.il/people/morad/publications/accmp-computer-architecture-letters-jul2005.pdf.
14. J.M. Paul and B.H. Meyer, "Amdahl's Law Revisited for Single Chip Systems," *Int'l J. Parallel Programming*, vol. 35, no. 2, 2007, pp. 101-123.
15. J.L. Gustafson, "Reevaluating Amdahl's Law," *Comm. ACM*, May 1988, pp. 532-533.

***Mark D. Hill** is a professor in the Computer Sciences and the Electrical and Computer Engineering Departments at the University of Wisconsin-Madison. His research interests include parallel computer system design, memory system design, and computer simulation. Hill received a PhD in computer science from the University of California, Berkeley. He is a Fellow of the IEEE and the ACM. Contact him at markhill@cs.wisc.edu.*

***Michael R. Marty** is an engineer at Google currently working on its computing platform. His interests include parallel computer systems design, distributed software infrastructure, and simulation. Marty received a PhD in computer science from the University of Wisconsin-Madison. Contact him at mikemarty@google.com.*