*To appear in IEEE Computer*

# Amdahl's Law in the Multicore Era

Mark D. Hill and Michael R. Marty

Computer Sciences Department
University of Wisconsin—Madison
*{markhill, mikem}@cs.wisc.edu*

## Abstract (Separate From Paper)

*We apply Amdahl's Law to multicore chips using symmetric cores, asymmetric cores, and dynamic techniques that allow cores to work together on sequential execution. To Amdahl's simple software model, we add a simple hardware model based on fixed chip resources.*

*Our results encourage multicore designers to view performance of the entire chip rather than focusing on core efficiencies. Moreover, we observe that obtaining optimal multicore performance requires further research in both extracting more parallelism and making sequential cores faster. We seek to stimulate discussion and future work, as well as temper the current pendulum swing from the past's under-emphasis on parallel research to a future with too little sequential research.*

## Tag Lines (With Paper)

*Everyone knows Amdahl's Law, but quickly forgets it.*
*-Dr. Thomas Puzak, IBM, 2007*

*By adding a simple model of multicore hardware to Amdahl's simple software model, this paper shows that obtaining optimal multicore performance requires further research in both extracting more parallelism and making sequential cores faster.*

Today we are at an inflection point in the computing landscape as we enter the multicore era. All computing vendors have announced chips with multiple processor cores. Moreover, vendor roadmaps 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 of multicore chips must subdue more degrees of freedom than single-core designs. Questions include: How many cores? Should cores use simple pipelines or powerful multi-issue pipeline designs? Should cores use the same or different micro-architectures? In addition, designers must concurrently manage power from both dynamic and static sources.

While answers to these questions are challenges for today's multicore chip with 2-8 cores, they will get much more challenging in the future. Sources as varied as Intel and Berkeley predict a hundred [6] if not a thousand cores [2].

Our thesis is that *Amdahl's Law* has important consequences for the future of the multicore era. The Sidebar reviews Amdahl's Law, including his model that a fraction *f* of software execution time is infinitely parallelizable without overhead, while the remaining fraction *1-f* is totally sequential.

To complement Amdahl's simple software model, we develop a simple model of multicore hardware resources. Our results encourage multicore designers to view performance of the entire chip rather than focusing on core efficiencies. Optimizing multicore performance requires further research in both extracting more parallelism and making sequential cores faster. In particular, we find:

- Not surprisingly, research and development work should target increasing parallelism (increasing *f*) when possible, as multicore chips do not nullify Amdahl's original law.
- Surprising to some, work should also target increasing core performance, even if it appears locally inefficient (e.g., tripling sequential performance using nine times more hardware resources). Faster sequential cores reduce the time until the efficient parallel phase resumes
- As Moore's Law provides multicore chips with more resources, the optimal designs should contain more powerful cores.
- *Asymmetric* (or *heterogeneous*) multicore designs—that allow one or more powerful cores—offer greater speedup potential than *symmetric* designs.
- *Dynamic* designs—that temporarily harness cores together to speed sequential execution—have the potential to achieve the best of both worlds.

After developing our simple hardware model of symmetric, asymmetric, and dynamic multicore chips, we will discuss important limitations of our models to stimulate both discussion and future work.

## Amdahl's Law Background (Sidebar)

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

$$Speedup_{enhanced}(f, S) = \frac{1}{(1-f) + \frac{f}{S}}$$

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, Amdahl originally defined his law for the special case of using *n* processors (cores today) in parallel when he argued for the *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities* [1]. He simplistically assumed 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:

$$Speedup_{parallel}(f, n) = \frac{1}{(1-f) + \frac{f}{n}}$$

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

While Amdahl's arguments were simple, they held and mainframes with one or a few processors dominated the computing landscape. They also largely held in minicomputer and personal computer eras that followed. As recent technology trends usher us into the multicore era, this paper shows Amdahl's Law is still relevant.

Amdahl's equations assumed the computation problem size does not change when running it on enhanced machines. That is, the fraction of a program that is parallelizable remains fixed. Gustafson argued that Amdahl's law does not do justice to massively parallel machines because they enable computations previously intractable in given time constraints [4]. A machine with greater parallel computation ability allows computations to 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.

## A Simple Cost Model for Multicore Chips

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. Herein we develop a simple hardware model in the spirit of Amdahl's simple software model.

We first assume that a multicore chip of given size and technology generation can contain at most *n base core equivalents (BCEs)*, 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 does *not* include chip resources expended on shared caches, interconnection networks, memory controllers, etc. Rather we simplistically assume that these non-processor resources are roughly constant in the multicore variations we consider.

We are agnostic on what limits a chip to *n* BCEs. It may be **power**; it may be **area**; it may be some combination of power, area, and other factors.

We second 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 specifically 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.

Our equations allow *perf(r)* to be an arbitrary function, but all the graphs below assume *perf(r)* = $\sqrt{r}$. In other words, we assume efforts that devote *r* BCE resources will result in performance $\sqrt{r}$. Thus, architectures can double performance at a cost of 4 BCEs, triple it for 9 BCEs, etc. We tried other similar functions, e.g., $\sqrt[1.5]{r}$, 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 1 BCE each, 4 cores of 4 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 1(a) and 1(b) show two hypothetical symmetric multicore chips for *n* = 16. The figures illustrate area, not power, as the chip's limiting resource and omit important structures such as memory interfaces, shared caches, and interconnects.

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*), total chip resources in BCEs (*n*), and the BCE resources (*r*) devoted to increase the performance of each core. 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:

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

To understand this equation, consider Figure 3(a). It assumes a symmetric multicore chip of *n* = 16 BCEs and *perf(r)* = $\sqrt{r}$. The x-axis gives resources used to increase performance of each core: a value 1 says the chip has 16 base cores, while 16 uses all resources for a single core. Lines assume different values for the fraction parallel (*f*=0.5, 0.9, ..., 0.999). The y-axis gives the speedup of the symmetric multicore chip *relative to running on one single-BCE base core*. The maximum speedup for *f=0.9*, for example, is 6.7 using 8 cores of cost 2 BCEs each.

Similarly, Figure 3(b) illustrates how tradeoffs change when Moore's Law enables *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, as achieving the best speedups requires *f*'s that are very near 1. Thus, finding parallelism is still critical.

**Implication 1:** Researchers should target increasing *f* via architectural support, compiler techniques, programming model improvements, etc.

Implication 1 is both most obvious and most important. Recall, however, that speedups much less than *n* can still be cost

**Note: These figures omit important structures and assume area, not power, is a chip's limiting resource.**

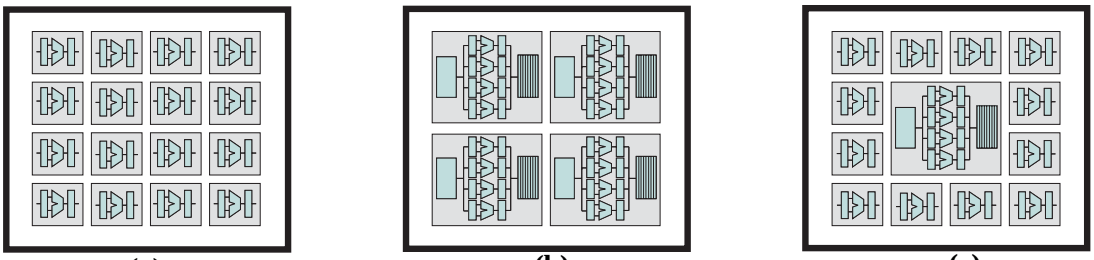


**Figure 1. (a) Symmetric Multicore with Sixteen 1-BCE cores, (b) Symmetric Multicore with Four 4-BCE cores, (c) Asymmetric Multicore with One 4-BCE core and 12 1-BCE cores**

effective. Recall that a system is *cost-effective* if speedup exceeds its *costup* [15]. Multicore costup is the multicore system cost divided by the single-core system cost. Since 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 1 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 should be non-minimal. 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 enables 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* where one or more cores are more powerful than the others [3, 8, 9, 10, 14]. With the simplistic assumptions of Amdahl's Law, it makes most sense to devote extra resources to increase the capability of only one core, as shown in Figure 1(c). With a resource budget of $n$=16 BCEs, for example, an asymmetric multicore chip can have one 4-BCE core and 12 1-BCE cores, one 9-BCE core and 7 1-BCE cores, etc. In general, the chip can have $1+n-r$ cores since the single larger core uses *r* resources and leaves $n-r$ resources for the 1-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:

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

Figure 3(c) shows asymmetric speedup curves for $n$=16 BCEs, while Figure 3(d) gives curves for $n$=256 BCEs. These curves are markedly different from the corresponding symmetric speedups in Figures 3(a) and 3(b). 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 in the middle of 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 51.2.

**Implication 4:** Researchers should continue to investigate asymmetric multicore chips, including dealing with scheduling and overhead challenges not captured by Amdahl's model.

**Result 5:** Denser multicore chips increase both the speedup benefit of going asymmetric (see above) and the optimal performance of the single large core. For f=0.975 and n=1024, 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*, e.g., $perf(r) = \sqrt{r}$. 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 together to boost performance of only the sequential component, as shown in Figure 2. This could be possible with thread-level speculation, helper threads, etc. [5,7,12,13]. 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:

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

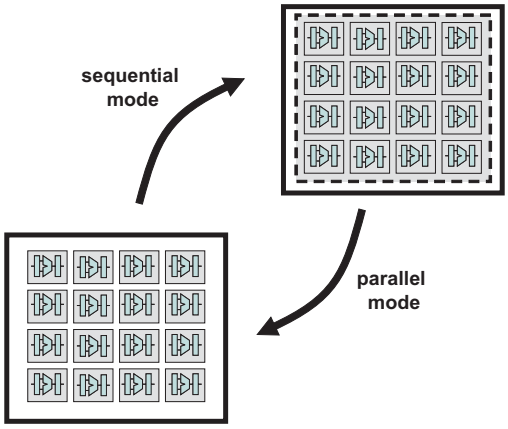


**Figure 2. Dynamic**
Sixteen 1-BCE cores

Figure 3(e) displays dynamic speedups when using *r* cores in sequential mode for *perf(r)* = $\sqrt{r}$ for *n*=16 BCEs, while Figure 3(f) gives curves for *n*=256 BCEs. The graphs show that performance always gets better as more BCE resources can be exploited to improve the sequential component. Practical considerations, however, may 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 larger numbers cores for sequential mode than can be done today. For *f=0.99* and *n=256*, for example, effectively harnessing all 256 cores would achieve a speedup of 223, which is much great than the comparable asymmetric speedup of 165. This result follows because we assume that dynamic chips can both gang together all resources 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. While these methods may 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 simple software model and our simple multicore hardware model 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.

Software is not just infinitely parallel and sequential. Software task and data movements add overhead. It is (currently) 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, Paul and Meyer [11] developed sophisticated models that question the validity of Amdhal's Law to future systems, especially embedded ones. On the other hand, more cores may advantageously enable greater parallelism from larger problem sizes, as envisioned by Gustafson [4].

Hardware designers cannot (currently) build cores that achieve arbitrary high performance by adding more and 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 ignored important effects due to dynamic and static power, as well as on- and off-chip memory system and interconnect design.

## Our Charge to You The Reader

Pessimists will bemoan our model's simplicity and lament that much of the design space we explore cannot 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 displayed in this paper. To help you get started, we provide the spreadsheet and code examples for this paper's models at

http://www.cs.wisc.edu/multifacet/ieeecomputer08_amdahl_multicore/

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 and/or proofreading. This work is supported in part by the National Science Foundation (NSF), with grants EIA/CNS-0205286, CCR-0324878, CNS-0551401, CNS-0720565, and CNS-0720565 as well as donations from Intel and Sun Microsystems. Hill has significant financial interest in Sun Microsystems. The views expressed herein are not necessarily those of the NSF, Intel, or Sun Microsystems.

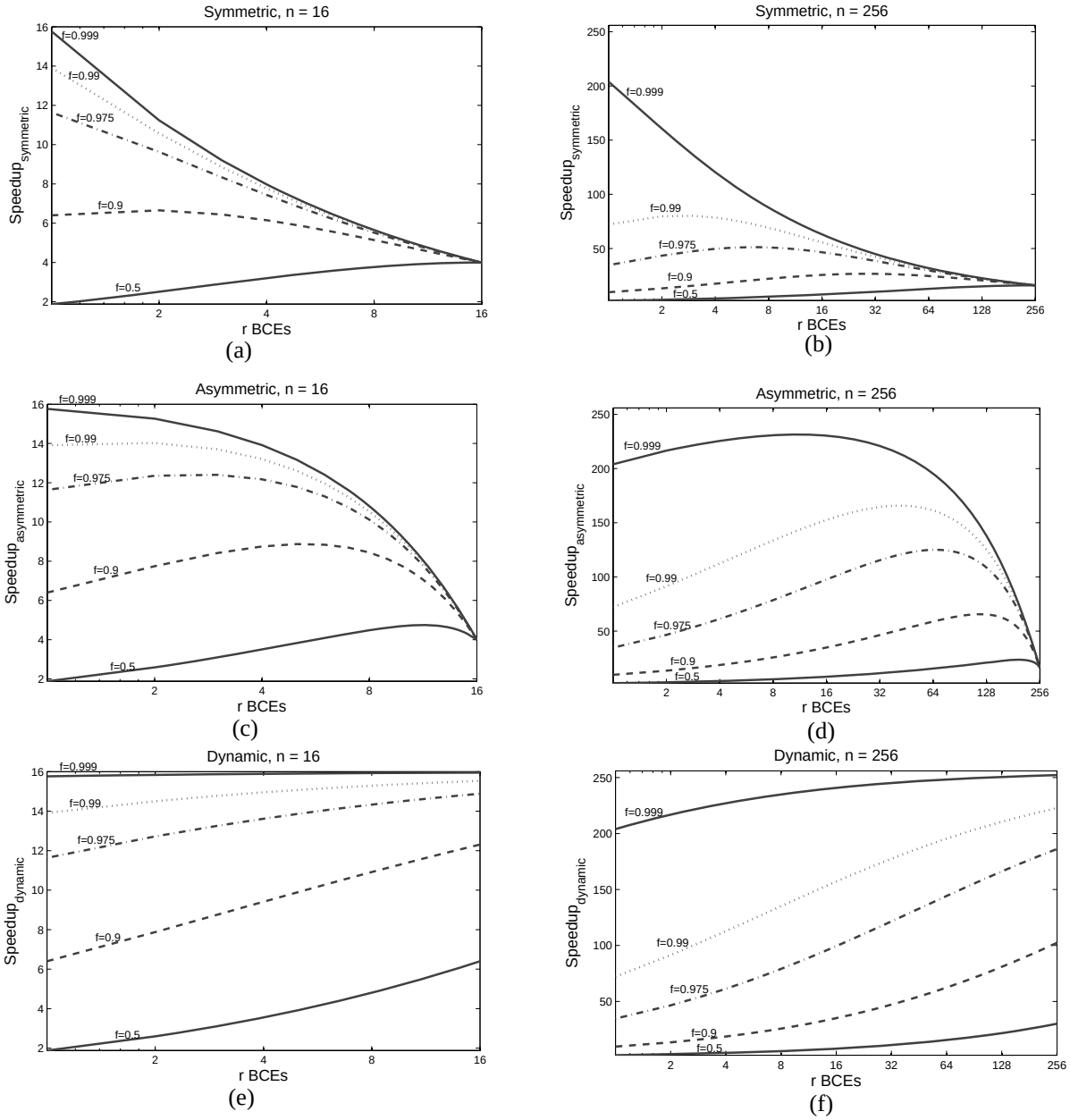


**Figure 3. Speedup of *Symmetric* (a & b), *Asymmetric* (c & d), and *Dynamic* (e & f) Multicore Chips with *n* = 16 BCEs (left) or *n* = 256 BCEs (right)**

## References


[1] G. M. Amdahl. Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS Conference Proceedings*, pages 483–485, April 1967.

[2] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samual Webb Williams, and Katherine A. Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report Technical Report No. UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[3] Saisanthosh Balakrishnan, Ravi Rajwar, Michael Upton, and Konrad Lai. The Impact of Performance Asymmetry in Emerging Multicore Architectures. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, June 2005.

[4] John L. Gustafson. Reevaluating Amdahl's Law. *Communications of the ACM*, 31(5):532–533, May 1988.

[5] Lance Hammond, Mark Willey, and Kunle Olukotun. Data Speculation Support for a Chip Multiprocessor. In *Proceedings of the Eighth International Conference on*

*Architectural Support for Programming Languages and Operating Systems*, pages 58–69, October 1998.

[6] Intel. From a Few Cores to Many: A Tera-scale Computing Research Overview. ftp://download.intel.com/research/platform/terascale/terascale_overview_pap% er.pdf, 2006.

[7] Engin Ipek, Meyrem Kirman, Nevin Kirman, and Jose F. Martinez. Core Fusion: Accomodating Software Diversity in Chip Multiprocessors. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, June 2007.

[8] J.A. Kahl, M.N. Day, H.P. Hofstee, C.R. Johns, T.R. Maeurer, and D. Shippy. Introduction to the Cell Multiprocessor. *IBM Journal of Research and Development*, 49(4), 2005.

[9] Rakesh Kumar, Keith I. Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, December 2003.

[10] Rakesh Kumar, Dean Tullsen, Norman Jouppi, and Partha Ranganathan. Heterogeneous Chip Multiprocessors. *IEEE Computer*, 38(11):32–38, November 2005.

[11] JoAnn M. Paul and Brett H. Meyer. Amdahl's Law Revisited for Single Chip Systems. *International Journal of Parallel Programming*, 35(2):101–123, April 2007.

[12] Jose Renau, Karin Strauss, Luis Ceze, Wei Liu, Smruti Sarangi, James Tuck, and Josep Torrellas. Energy-Efficient Thread-Level Speculation on a CMP. *IEEE Micro*, 26(1), Jan/Feb 2006.

[13] G.S. Sohi, S. Breach, and T.N. Vijaykumar. Multiscalar Processors. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 414–425, June 1995.

[14] M. Aater Suleman, Yale N. Patt, Eric A. Sprangle, Anwar Rohillah, Anwar Ghuloum, and Doug Carmean. ACMP: Balancing Hardware Efficiency and Programmer Efficiency. Technical Report HPS Technical Report, TR-HPS-2007-001, University of Texas, Austin, February 2007.

[15] David A. Wood and Mark D. Hill. Cost-Effective Parallel Computing. *IEEE Computer*, pages 69–72, February 1995.