Amdahl's Law in the Multicore Era

Mark D. Hill and Michael R. Marty

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

Abstract

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

A key result we find is that, even as we enter the multicore era, researchers should still seek methods of speeding sequential execution. Moreover, methods that appear locally inefficient *(e.g., tripling sequential performance with a 9x resource cost) can still be* globally efficient *as they reduce the sequential phase when the rest of the chip's resources are idle.*

To reviewers: This paper's accessible form is between a research contribution and a perspective. It seeks to stimulate discussion, controversy, and future work. In addition, it seeks to temper the current pendulum swing from the past's underemphasis on parallel research to a future with too little sequential research.

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 ones? 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. Source as varied as Intel and Berkeley predict a hundred [6] if not a thousand cores [2].

It is our thesis that *Amdahl's Law* has important consequences for the future of our multicore era. Since most of us learned Amdahl's Law in school, all of our points are "known" at some level. Our goal is ensure we remember their implications and avoid the pitfalls that Puzak fears.

Table 1 foreshadows the results we develop for applications that are 99% parallelizable. For varying number of base cores, the second column gives the upper bounds on speedup as predicted by Amdahl's Law. In this paper, we develop a simple hardware model that reflects potential tradeoffs in devoting chip resources towards either parallel or sequential execution.The right-most three columns show the larger upper bounds on speedup enabled using richer cores deployed in symmetric, asymmetric, and dynamic multicore designs.

TABLE 1. Upper Bound on Speedup, *f*=0.99

# Base Core Equivalents	Base Amdahl	Symmetric	Asymmetric	Dynamic
16	14	14	14	< 16
64	39	39	49	< 60
256	72	80	166	< 223
1024	91	161	531	< 782

Implications of our work include:

- Not surprisingly, researchers and architects should aggressively attack serial bottlenecks in the multicore era.
- Increasing core performance, even if it appears *locally inefficient*, can be *globally efficient* by reducing the idle time of the rest of the chip's resources.
- Chips with more resources tend to encourage designs with richer cores.
- *Asymmetric* multicore designs offer greater potential speedup than *symmetric* designs, provided challenges (e.g., scheduling) can be addressed.
- *Dynamic* designs—that temporarily harness cores together to speed sequential execution—have the potential to achieve the best of both worlds.

Overall, we show that Amdahl's Law beckons multicore designers to view performance of the entire chip rather than zeroing in on core efficiencies. In the following sections, we first review Amdahl's Law. We then present simple hardware models for symmetric, asymmetric, and dynamic multicore chips.

Amdahl's Law Background

Most computer scientists learned Amdahl Law's in school [5]. 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:* If *f* is small, your optimizations will have little effect.
- *But the aspects you ignore also limit speedup:* As *S* goes to infinity, Speedup goes to *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 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, he 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, we investigate whether Amdahl's Law is still relevant.

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, interconnections, 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 richer 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 rich 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* = 64 BCEs, for example, can support 64 cores of 1 BCE each, 16 cores of 4 BCEs each, or, in general, *n/r* cores of *r* BCEs each (rounded down to an integer number of cores). Figures 1 and 2 show cartoons of two possible 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, let's begin with the upper-left graph of Figure 4. 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. The remaining left-hand graphs give speedups for symmetric multicore chips with chip resources of *n* = 64, 256, and 1024 BCEs.

Result 1: Amdahl's Law applies to multicore chips, as achieving good speedups requires *f*'s that are very near 1. Thus, finding parallelism is 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 effective.[1]

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 maxi-

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

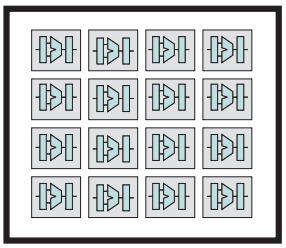

Figure 1. Symmetric
Sixteen 1-BCE cores

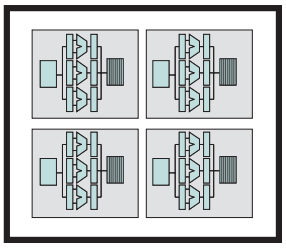

Figure 2. Symmetric
Four 4-BCE cores

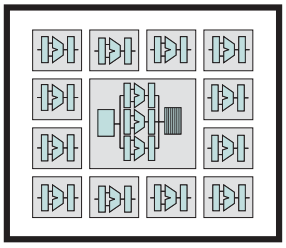

Figure 3. Asymmetric
One 4-BCE core, 12 1-BCE cores

mum speedup can occur at 1 big core, *n* base cores, or with an intermediate number of middle-sized cores. Consider, *n*=16. With *f*=0.5, one core (of cost 16 BCEs) gives the best speedup of 4. With *f*=0.975, 16 single-BCE cores provide a speedup of 11.6. With *n*=64, *f*=0.9, 9 cores of 7.1 BCEs each provides an overall speedup of 13.3.

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 sizes *n*=16 and 64, but more powerful cores help at *n*=256 and 1024.

Implication 3: Even as Moore's Law allows larger chips, researchers should look for ways to design more powerful cores.

Asymmetric Multicore Chips

An alternative to a symmetric multicore chip is an *asymmetric multicore chip* where one or more cores are more powerful than the others [3, 8, 9, 12]. 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 3. With a resource budget of *n*=64 BCEs, for example, an asymmetric multicore chip can have one 4-BCE core and 60 1-BCE cores, one 9-BCE core and 55 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:

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

The asymmetric speedup curves are shown in Figure 4. These curves are markedly different from the corresponding symmetric speedups. 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 reach a maximum speedup in the middle of the extremes.

Result 4: Asymmetric multicore chips can offer maximum 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. For *n=1024* and the same *f*, the difference increases to 364.5 versus 102.5. This result follows from Amdahl's idealized software assumptions, wherein software is either completely sequential or completely parallel.

Implication 4: Researchers should continue to investigate asymmetric multicore chips. However, real chips must deal with many challenges, such as scheduling different phases of parallelism with real overheads. Furthermore, chips may need multiple larger cores for multiprogramming and workloads that exhibit overheads 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, for example, best speedup is obtained 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.

1. A system is *cost-effective* if its speedup exceeds its *costup* [13]. 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.

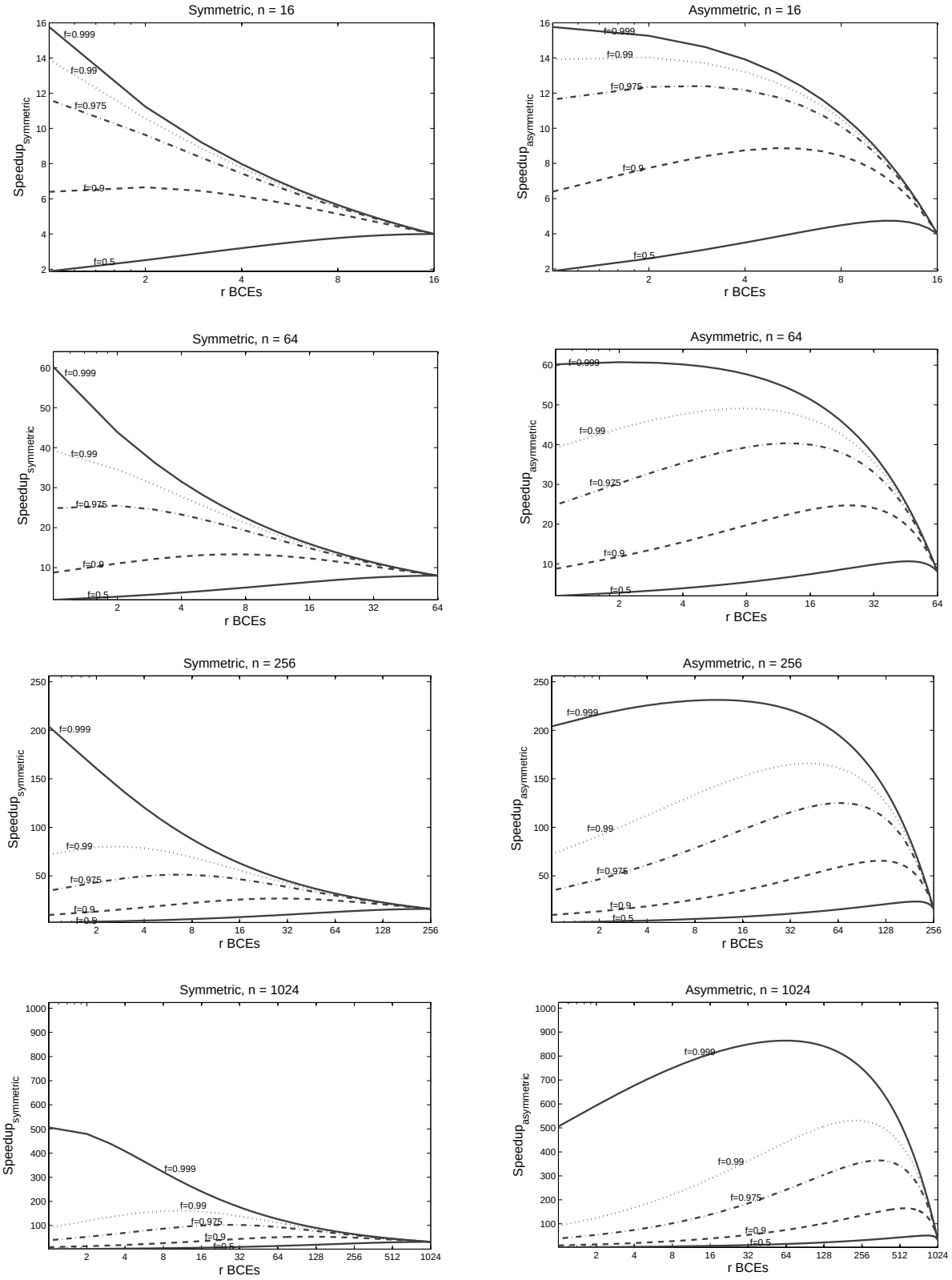

Figure 4. Speedup of Symmetric and Asymmetric Multicore chips.

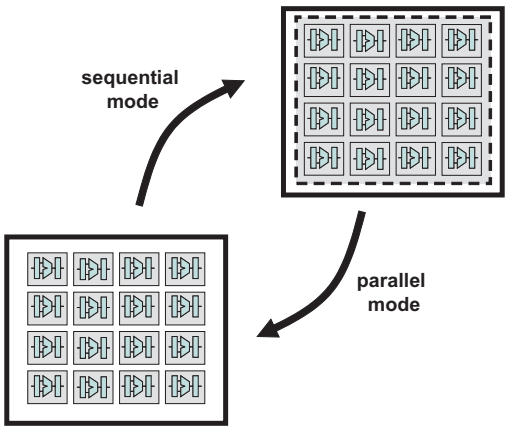

Figure 5. Dynamic
Sixteen 1-BCE cores

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 5. This could be possible with thread-level speculation, helper threads, etc. [4, 7, 10, 11]. 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:

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

Of course dynamic techniques may have some additional resource overhead (e.g., area) not reflected in the equation as well as additional runtime overhead when combining and splitting cores. Figure 6 displays dynamic speedups when using *r* cores in sequential mode for $perf(r) = \sqrt{r}$. Light grey lines give the corresponding values for asymmetric speedup. 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 Amdahl's sequential-parallel assumption, however, achieving much greater speedup than asymmetric chips requires that dynamic techniques harness large numbers of BCEs in sequential mode. For *f=0.999* and *n=256*, for example, the dynamic speedup attained when 256 BCEs can be utilized in sequential mode is 963. However the comparable asymmetric speedup is 748. 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.

Conclusions

To Amdahl's simple software model, we add a simple hardware model and compute speedups for *symmetric, asymmetric,* and *dynamic* multicore chips:

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

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

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

Results first reaffirm that seeking greater parallelism is critical to obtaining good speedups. We then show how core designs that are locally inefficient can be globally efficient.

Of course, our model seeks insight by making many simplifying assumptions. The real world is much more complex. Amdahl's simple software model may not hold reasonably for future software. Future mainstream parallel software may also behave differently from today's highly tuned parallel scientific and database codes. Our simple hardware model does not account for the complex tradeoffs between cores, cache capacity, interconnect resources, and off-chip bandwidth. Nevertheless, we find value in the controversy and discussion that drafts of this paper have already stimulated.

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, and CNS-0551401, 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.

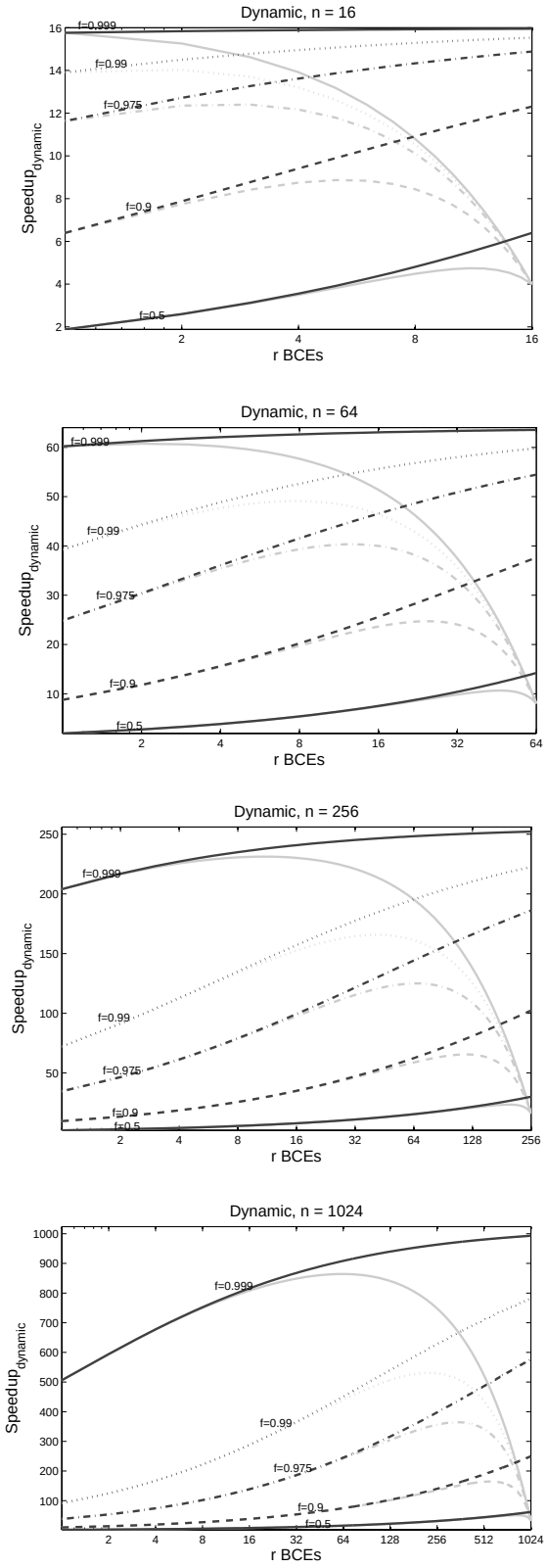

Figure 6. Speedup of Dynamic Multicore Chips (light lines show asymmetric speedup)

References

[1] G. M. Amdahl. Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS Conference Proceedings*, pages 483–485, 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 *ISCA 32*, June 2005.

[4] Lance Hammond, Mark Willey, and Kunle Olukotun. Data Speculation Support for a Chip Multiprocessor. In *ASPLOS 8*, pages 58–69, October 1998.

[5] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, third edition, 2003.

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

[7] Engin Ipek, Meyrem Kirman, Nevin Kirman, and Jose F. Martinez. Core Fusion: Accomodating Software Diversity in Chip Multiprocessors. In *ISCA 34*, 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 *MICRO 36*, December 2003.

[10] 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.

[11] G.S. Sohi, S. Breach, and T.N. Vijaykumar. Multiscalar Processors. In *ISCA 22*, pages 414–425, June 1995.

[12] 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.

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