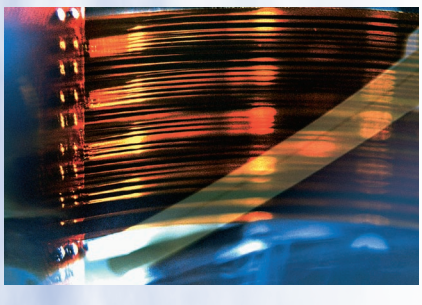
# Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era



*Dong Hyuk Woo and Hsien-Hsin S. Lee*
*Georgia Institute of Technology*

**An updated take on Amdahl's analytical model uses modern design constraints to analyze many-core design alternatives. The revised models provide computer architects with a better understanding of many-core design types, enabling them to make more informed tradeoffs.**

Unsustainable power consumption and ever-increasing design and verification complexity have driven the microprocessor industry to integrate multiple cores on a single die, or *multicore*, as an architectural solution sustaining Moore's law.[1] With dual-core and quad-core processors on the market and oct-core on the horizon, researchers already are a step ahead. They're investigating architectures, compilers, and programming models for a *many-core* processor with hundreds or even 1,000 cores on a single platform.[2,3]

In 1967, Gene Amdahl proposed an often overlooked law of scaling: A program's sequential computation largely limits the maximum achievable speedup.[4] This implies that any nonparallel execution or intercore communication will rapidly diminish the performance scalability for parallel applications regardless of the amount of additional computation resources. A simple, yet insightful, observation, Amdahl's law continues to serve as a guideline for parallel programmers to assess the upper bounds of attainable performance.

Unfortunately, beyond performance, computer architects face another Grand Challenge: energy efficiency. Architects should carefully design a future many-core processor so that its power consumption doesn't exceed its power budget.[5] For example, a 16-core processor with each core consuming an average of 20 watts will lead to 320 watts total power when all cores are active. This level of consumption can easily exceed a single processor die's power budget. In other words, the amount of power each core consumes will dictate the number of cores architects can integrate on-die. Apparently, power is becoming more critical than performance in scaling up many-core processors. Thus, before integrating a large number of cores on-chip to provide desired performance and throughput, architects must maximize each core's power efficiency.

Tackling these new design challenges requires extending Amdahl's law to account for power scalability's implications in the coming many-core era. As the original Amdahl's law demonstrates, a simple analytical model can provide computer architects with useful insights. By using simple analytical models at the early design phase, we aim to provide a better understanding of energy-efficiency's limits, some feasible many-core design options, and future directions for making many-core more scalable.


Published by the IEEE Computer Society
0018-9162/08/$25.00 © 2008 IEEE

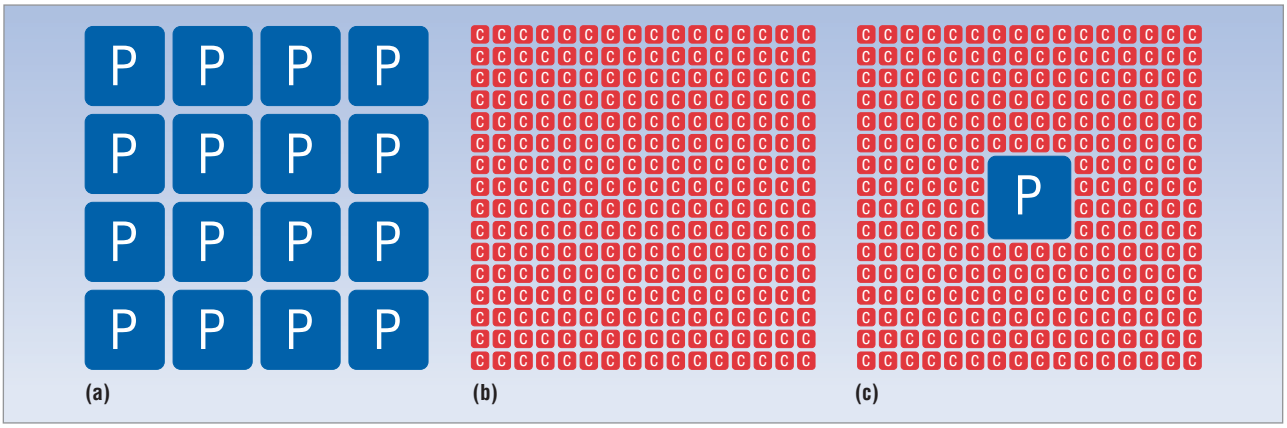



*Figure 1. Many-core design styles. (a) A symmetric many-core processor that replicates a state-of-the-art superscalar processor on a die, and (b) a symmetric many-core processor that replicates a smaller, more power-efficient core on a die. (c) An asymmetric many-core processor with numerous efficient cores and one full-blown processor as the host processor.*

## MANY-CORE DESIGN STYLES

For our study, we broadly classify future many-core architectures into three types. The first is a symmetric many-core processor that simply replicates a state-of-the-art superscalar processor on a die, as in Figure 1a. High-end multicore processor vendors such as Intel and AMD use this approach. It's flexible and general enough to run different processes simultaneously while providing the best single-thread performance. Additionally, it can run independent threads spawned from one process to improve a single application's performance. We use $P$ to represent a single state-of-the-art superscalar processor and $P^*$ to represent this type of many-core design style.

As Figure 1b shows, the second design style is a symmetric many-core processor that replicates a smaller, yet more power-efficient, core on a die. Embedded many-core processors, such as picoChip,[6] Connex Machine,[7] and TILE64 (www.tilera.com/products/processors.php), use this approach. The performance of a processing core using this approach isn't as high as that of a state-of-the-art superscalar processor. However, architects can integrate more processing cores on a die using this approach, thus the aggregate on-chip performance might be comparable to $P^*$. We use $c$ to denote a smaller, more power-efficient processing core and $c^*$ to represent this many-core design style.

The third design style, shown in Figure 1c, is an asymmetric many-core processor that contains many efficient cores ($c^*$) and one full-blown processor ($P$) as the host. The Sony-Toshiba-IBM (STI) Cell Broadband Engine[8] and a recent proposed research project, POD,[9] are examples of such an asymmetric many-core processor. This design style lacks the flexibility to run different processes simultaneously. Nevertheless, the single-thread performance on the host processor should be high, because it guarantees state-of-the-art sequential performance for certain applications. Moreover, it provides highly parallel performance when the efficient cores are in use. We use $P + c^*$ to represent this design style.

## AUGMENTING AMDAHL'S LAW

While Amdahl mainly focused on performance scalability back in the 1960s, we're more interested in the power scalability or energy efficiency of future many-core processors. Here, we develop analytical power models of each design and formulate metrics to evaluate energy efficiency on the basis of performance and power models.

### Models for $P^*$

According to Amdahl's law, the formula for computing the theoretical maximum speedup (or performance) achievable through parallelization is as follows:

$$Perf = \frac{1}{(1-f)+\frac{f}{n}} \quad (1)$$

where $n$ is the number of processors, and $f$ is the fraction of computation that programmers can parallelize ($0 \le f \le 1$).

To model the power consumption for a $P^*$ many-core processor, we introduce a new variable, $k$, to represent the fraction of power the processor consumes in idle state ($0 \le k \le 1$). We assume that one superscalar processor in active state consumes a power of 1. By definition, the amount of power one full-blown processor consumes during the sequential computation phase is 1, while the remaining $(n - 1)$ full-blown processors consume $(n-1)k$. Thus, during the sequential computation phase, $P^*$ consumes $1 + (n - 1)k$. For the parallel computation phase, $n$ full-blown processors consume $n$ amount of power. Because it takes $(1 - f)$ and $f/n$ to execute the sequential and parallel code, respectively, the formula for average power consumption (denoted by $W$) for a $P^*$ is as follows:

$$W = \frac{(1-f)\times\{1+(n-1)k\}+\frac{f}{n}\times n}{(1-f)+\frac{f}{n}} = \frac{1+(n-1)k(1-f)}{(1-f)+\frac{f}{n}} \quad (2)$$

Now, we can model *performance per watt* (*Perf*/*W*), which represents the performance achievable at the same cooling capacity, based on the average power (*W*) in Equation 2. This metric is essentially the reciprocal of energy, because the definition of *performance* is the reciprocal of execution time. Because *Perf*/*W* of single-core execution is 1, the *Perf*/*W* benefit of a *P** is expressed as

$$\frac{Perf}{W} = \frac{1}{(1-f)+\frac{f}{n}}\times\frac{(1-f)+\frac{f}{n}}{1+(n-1)k(1-f)} = \frac{1}{1+(n-1)k(1-f)} \quad (3)$$

In addition to *Perf*/*W*, we can also model *performance per joule* (*Perf*/*J*), a metric for evaluating the performance achievable in the same battery life cycle or, more specifically, energy. *Perf*/*J* is equivalent to the reciprocal of energy-delay product.[10] Using Equation 1 and Equation 3, the formula for performance per joule is as follows:

$$\frac{Perf}{J} = \frac{1}{(1-f)+\frac{f}{n}}\times\frac{1}{1+(n-1)k(1-f)}$$

## Models for *c**

The performance model of a *c** many-core processor has been a topic of Mark Hill and Michael Marty's recent research.[11] This model assumes that one larger core consumes the same amount of die area that several smaller cores consume.

We slightly modified this performance model to accommodate arbitrarily sized cores. To model the performance difference between a full-blown processor (*P*) and an efficient core (*c*), we introduce the variable $s_c$. This variable represents an efficient core's performance normalized to that of a full-blown processor ($0 \le s_c \le 1$). Because each efficient core's performance is $s_c$, the formula for calculating *c**'s performance model is as follows:

$$Perf = \frac{s_c}{(1-f)+\frac{f}{n}}$$

To model *c**'s power consumption, we need two new variables: $w_c$ and $k_c$. The first variable represents an active efficient core's power consumption relative to that of an active full-blown processor ($0 \le w_c \le 1$); the second represents the fraction of an efficient core's idle power normalized to the same core's overall power consumption ($0 \le k_c \le 1$). During the sequential computation phase, one efficient core in active state consumes $w_c$, and all idle cores consume $(n-1)\times w_c \times k_c$. During the parallel computation phase, all efficient cores consume $n \times w_c$. Because it takes $(1-f)/s_c$ and $f/(n \times s_c)$ to perform sequential and parallel computation, respectively, the average power consumption by a *c** is

$$W = \frac{\frac{1-f}{s_c}\times\{w_c+(n-1)w_c k_c\}+\frac{f}{ns_c}\times nw_c}{\frac{1-f}{s_c}+\frac{f}{ns_c}} = \frac{w_c+(n-1)w_c k_c(1-f)}{(1-f)+\frac{f}{n}}$$

Thus, the following equations can represent *Perf*/*W* and *Perf*/*J*:

$$\frac{Perf}{W} = \frac{s_c}{w_c+(n-1)w_c k_c(1-f)} \quad \text{and}$$

$$\frac{Perf}{J} = \frac{s_c}{(1-f)+\frac{f}{n}}\times\frac{s_c}{w_c+(n-1)w_c k_c(1-f)}$$

## Models for *P* + c**

Hill and Marty have also studied the performance model of a *P* + *c** many-core processor.[11] We slightly modify this performance model. Executing the sequential code at the host processor (one *P*) takes $(1-f)$, whereas executing the parallel code using the efficient cores takes $f/\{(n-1)s_c\}$. (A *P* + *c** many-core processor contains one *P* and $(n-1)$ *c* cores.) Note that we assume the host processor to be idle while the efficient cores are executing the parallel code. Thus, the formula for computing performance improvement using a *P* + *c** is as follows:

$$Perf = \frac{1}{(1-f)+\frac{f}{(n-1)s_c}}$$

During the sequential computation phase, the amount of power the full-blown processor consumes is 1, and the amount the efficient cores consume is $(n-1)w_c k_c$. During the parallel computation phase, its full-blown processor consumes *k*, while the efficient cores consume $(n-1)\,w_c$. Because executing sequential and parallel code takes $(1-f)$ and $f/(n-1)s_c$, the average power is

$$W = \frac{(1-f)\{1+(n-1)w_c k_c\}+\frac{f}{s_c}\left\{\frac{k}{n-1}+w_c\right\}}{(1-f)+\frac{f}{(n-1)s_c}}$$

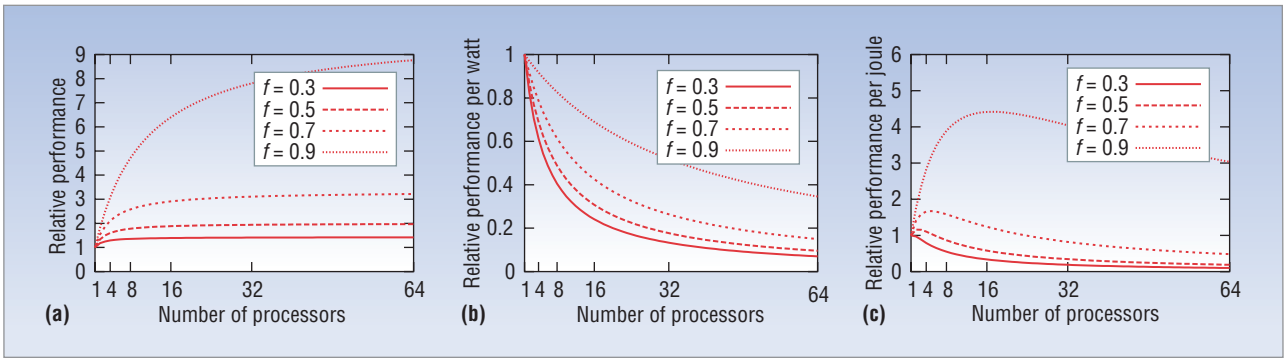



***Figure 2. P* scalability. P*, a symmetric many-core processor that replicates a state-of-the-art superscalar processor on a die, consumes a high amount of energy to complete the task: (a) performance, (b) performance per watt, and (c) performance per joule, where k = 0.3.***

Consequently, *Perf*/*W* of a $P + c^*$ is expressed as

$$\frac{Perf}{W} = \frac{1}{(1-f)\{1+(n-1)w_c k_c\} + \frac{f}{s_c}\left\{\frac{k}{n-1} + w_c\right\}}$$

and *Perf*/*J* of a $P + c^*$ as

$$\frac{Perf}{J} = \frac{1}{(1-f) + \frac{f}{(n-1)s_c}} \times \frac{1}{(1-f)\{1+(n-1)w_c k_c\} + \frac{f}{s_c}\left\{\frac{k}{n-1} + w_c\right\}}$$

## Power-equivalent models

Because the limited power budget is one of the most critical design constraints, comparing different designs without considering the single-chip power budget is meaningless.

Two main factors limit power growth on a single chip: power supply and power density. Power supply is proportional to the energy cost for sustaining machines in data centers, as well as a concern for portable devices' battery life. Power density pertains to thermal control mechanisms' extra complexity and cost. From the power budget perspective, take, for example, a full-blown processor and an efficient core that consume 20 W and 5 W, respectively. Given a 160-W maximum power budget, we can integrate only eight full-blown processors or 32 efficient cores on a single die. Thus, to perform an apples-to-apples comparison for a given power budget, we developed power-equivalent models by converting the number of cores of a $c^*$ or $P + c^*$ to an equivalent number of full-blown processors of a $P^*$.

Let $W_{budget}$ be the single-chip power budget and $n_{P^*}$ be the maximum number of full-blown processors we can implement on a $P^*$ die. Because a full-blown processor's power consumption is modeled as 1, $n_{P^*}$ full-blown processors on a die can consume up to $n_{P^*}$. Therefore, the maximum number of full-blown processors on a $P^*$ is $n_{P^*} = W_{budget}$.

Conversely, $n_{c^*}$ cores of a $c^*$ consume power up to $n_{c^*} \times w_c$, which should be less than or equal to $W_{budget}$. So, the maximum number of efficient cores on a $c^*$ is $n_{c^*} = W_{budget}/w_c$.

Similarly, $n_{P+c^*}$ cores of a $P + c^*$ consume power up to $1 + (n_{P+c^*} - 1)w_c$. Again, a single-chip power budget, $W_{budget}$, constrains the number of cores an architect can implement on a chip. Consequently, the maximum $n_{P+c^*}$ is

$$n_{P+c^*} = 1 + \frac{W_{budget} - 1}{w_c}$$

Using these equations, we can uniformly represent and compare performance, *Perf*/*W*, and *Perf*/*J* of each many-core style with respect to a single-chip power budget.

## EVALUATION

To thoroughly compare the design styles, we evaluated within $P^*$, $c^*$, and $P + c^*$ as well as across designs.

### Evaluating a *P**

Figure 2b shows *Perf*/*W* of a $P^*$. Unfortunately, parallel execution on a $P^*$ consumes much more energy than sequential execution to complete the task. In the ideal case of $f = 1$, in which we can parallelize the entire code, we can achieve the maximum *Perf*/*W*—that is, 1. In other words, a sequential execution and its parallel execution version will consume the same amount of energy only when the performance improvement through parallelization scales linearly. Otherwise, a $P^*$ must dissipate more energy to finish the same task. This occurs because performance doesn't scale linearly, as Figure 2a shows, but the amount of idle power does scale linearly with the number of cores.

Another interesting implication of this outcome addresses battery life. If we want to optimize the system for a longer battery life, it's better to run several processes on different cores rather than parallelize each

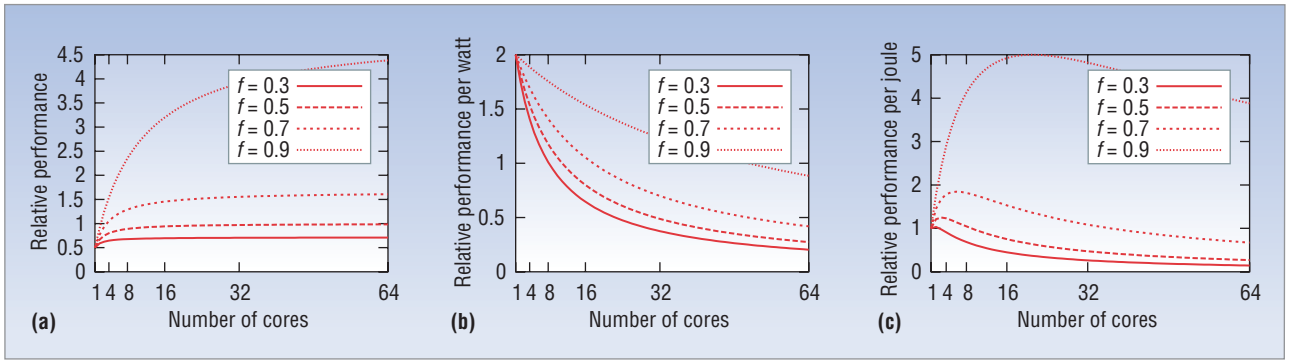



***Figure 3. c* scalability. The maximum speedup of this c*—a symmetric many-core processor that replicates a smaller, more power-efficient core on a die—isn't as high as that of P*: (a) performance, (b) performance per watt, and (c) performance per joule, where $s_c = 0.5$, $w_c = 0.25$, and $k_c = 0.2$.***

process and time-multiplex multitask them. Although the number of processes is fewer than the number of cores, spawning as few threads as possible so that different processes can run simultaneously is more power efficient. This improved efficiency is because *Perf/W* becomes worse as the number of cores increases. Furthermore, this result implies that maximizing and balancing parallelization among processors is also important, not only for higher performance but also for power-supply efficiency and extended battery life. However, no matter how well the code is parallelized or its performance scales, parallelization on a *P** many-core will always consume more energy unless the parallel performance scales perfectly linearly.

Figure 2c shows the *Perf/J* of a *P**. The evaluation result demonstrates that, if the performance of a parallelized application scales well, we can expect performance improvement at the same energy budget. In other words, a *P** can extract greater performance when running embarrassingly parallel applications given the same amount of energy. For example, when $f = 0.9$ and $k = 0.3$, a 16-core *P** can achieve a speedup more than four times that of a single-core processor using the same amount of energy.

However, parallelization on a *P** doesn't always lead to better *Perf/J*, as Figure 2c shows. For example, an application, half of which we can parallelize ($f = 0.5$), loses energy efficiency if we parallelize it with eight full-blown processors. This means that, from both the *Perf/W* and *Perf/J* perspectives, efforts to parallelize applications that can't be parallelized well might not be useful at all.

Another interesting observation is the existence of an optimal number of cores to achieve the best possible *Perf/J*. So, if we're particularly interested in tuning a system for this metric, dynamic monitoring and adaptively adjusting the system will be helpful. For example, given a 32-core *P**, it's wise to enable only 17 full-blown processors when running an application with $f = 0.9$—that is, 90 percent of it can be parallelized. In this case, it's best to completely shut off the remaining 15 full-blown processors to suppress unnecessary idle energy consumption.

## Evaluating a *c**

To evaluate a *c**'s performance and power consumption, we must model the relationship between a core's performance and size. To do this, we use Fred Pollack's performance efficiency rule.[12] It states that, given the same process technology, the state-of-the-art processor provides 1.5 to 1.7 times higher performance and consumes 2 to 3 times the die area compared with its previous-generation counterpart. This means that a processor that consumes $T$ times more transistors can provide only $\sqrt{T}$ times higher performance. On the other hand, the rule also implies that the processor is $\sqrt{T}$ times less efficient in terms of area. Another rule of thumb used in this evaluation is that a core's power consumption is proportional to the number of transistors it contains.

Figure 3 shows the analytical results of a *c**. In this analysis, we assume that each efficient core *c* has one-fourth the number of transistors of a full-blown processor *P*. We then model this efficient core's power consumption as one-fourth that of a full-blown processor ($w_c = 0.25$). We also assume the efficient core's performance to be one-half that of a full-blown processor ($s_c = 0.5$) and its fraction of power to be 20 percent ($k_c = 0.2$).

Figure 3a shows that the maximum speedup of this *c** isn't as high as that of *P**. The primary reason is that an efficient core's sequential performance is lower. As Amdahl's law says, sequential performance strictly limits the maximum speedup, and a *c** design quickly levels off the speedup. Figure 3b shows that, when the number of cores is small, a *c** consumes less energy than a single-core, full-blown processor baseline. This occurs mainly because the performance-to-power ratio of an efficient core is better than that of a full-blown processor. Unfortunately, as the number of cores increases, the amount of energy consumption becomes higher than that of a single-core full-blown processor baseline. Furthermore, Figure

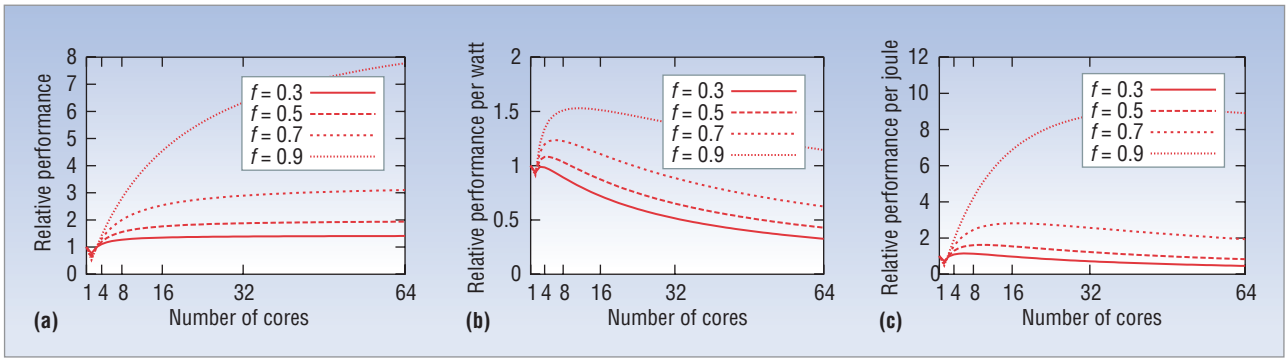



*Figure 4. P + c* scalability. P + c* is an asymmetric many-core processor with numerous efficient cores and one full-blown processor as the host processor: (a) performance, (b) performance per watt, and (c) performance per joule, where* $k = 0.3$, $s_c = 0.5$, $w_c = 0.25$, *and* $l_c = 0.2$.

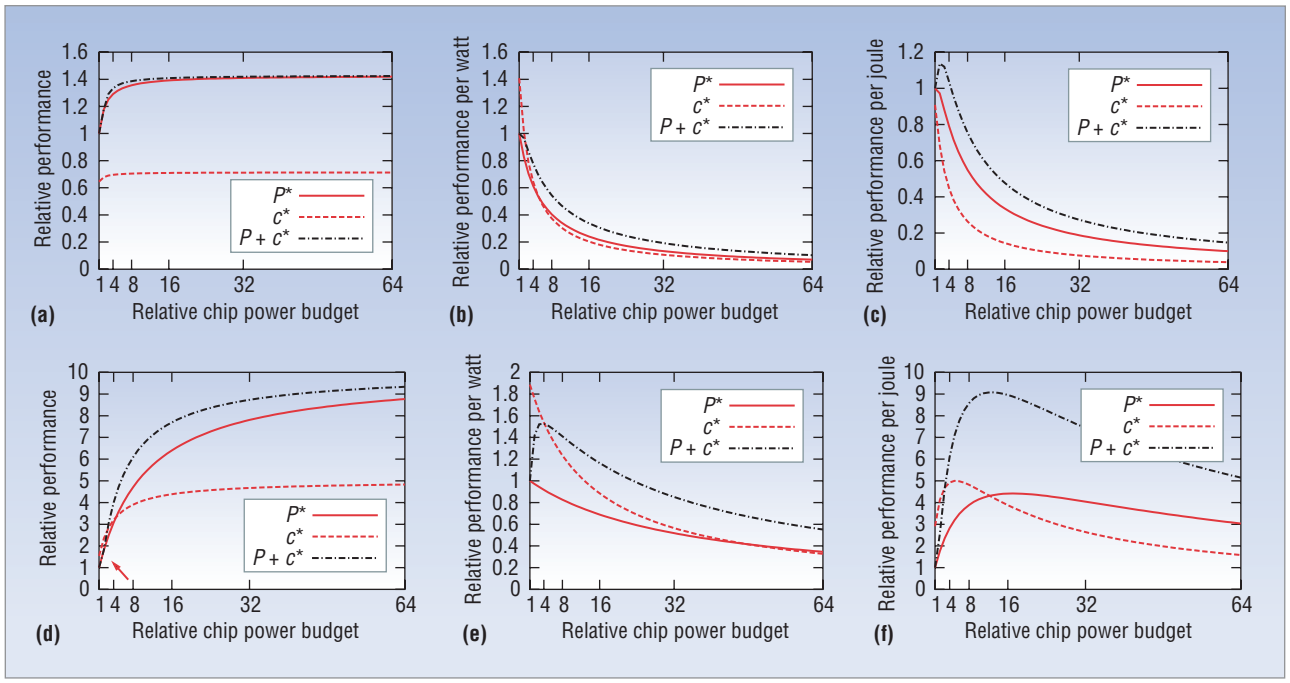



*Figure 5. Power-equivalent models. We used power-equivalent models to perform cross-design comparisons. Given* $f = 0.3$, *we measured (a) performance, (b) performance per watt, and (c) performance per joule. Given* $f = 0.9$, *we measured (d) performance, (e) performance per watt, and (f) performance per joule.*

3c shows that *Perf/J* of a *c** isn't good either, unless the application is embarrassingly parallel—that is, it has high *f* values. This means that performance saturation is the major contributor that leads to a low *Perf/J*.

## Evaluating a *P* + *c**

Figure 4b shows the *Perf/W* of a *P* + *c**, where $s_c$, $w_c$, and $k_c$ are modeled as 0.5, 0.25, and 0.2, respectively. Unlike a *P** or *c**, whose *Perf/W* monotonically decreases, an optimal number of cores exists that consumes the least amount of energy to execute an application. For example, we can improve the *Perf/W* of an embarrassingly parallel application ($f = 0.9$) by about 50 percent, when eight cores execute it.

However, *Perf/W* becomes worse than that of a one-core baseline processor when the number of cores exceeds a certain peak. There are two reasons for this result: efficient cores' relative power efficiency and performance saturation. When the number of cores is small, the additional performance benefit gained by adding one efficient core to the host processor dominates additional power overhead, so *Perf/W* increases. However, once performance improvement starts to saturate, as Figure 4a shows, additional power overhead dominates. Thus, *Perf/W* decreases, as in Figure 4b. In an energy-constrained environment such as embedded systems, how to spawn the optimal number of threads and turn off unused cores will be an interesting topic of investigation.

Figure 4c shows the *Perf/J* of a $P + c^*$. Because of its low-latency sequential execution and energy-efficient parallel execution, a $P + c^*$ achieves the best *Perf/J* compared with the two previous designs.

## Evaluating power-equivalent models

In addition to evaluating each many-core design style on its own, we use power-equivalent models to perform cross-design comparisons. Because the power budget is the major design constraint, the amount of power one core consumes determines the number of cores architects can implement on a single die. So, to compare different many-core designs, it's better to study performance and energy efficiency with the same power budget, rather than with the same number of cores.

Figure 5 shows the evaluation results with power-equivalent models. We assume each efficient core to consume one-fourth the power of a full-blown processor ($w_c = 0.25$) and its performance to be half that of a full-blown processor ($s_c = 0.5$). As Figures 5a and 5d show, the power-equivalent performance of a $P + c^*$ is found to be highest in most cases. The power-equivalent performance of a $P^*$ approaches that of a $P + c^*$ when $f$ is small. As $f$ increases, the difference between them grows, because a $P + c^*$ can have more cores at the same power budget. The power-equivalent performance of a $c^*$ improves as $f$ increases, as Figures 5a and 5d show, but it's still the lowest among the three in most cases.

When $f = 0.9$ and the relative power budget is very low, the power-equivalent performance of a $c^*$ is the highest (a pointer highlights this area in Figure 5d). In other words, in terms of performance itself, a $c^*$ is preferable only when applications contain a huge amount of parallelism, and the system is extremely power limited. Embedded devices designed for multimedia or data-streaming applications fall into this category.

Figures 5b and 5e show power-equivalent *Perf/W*. When the relative chip power budget is small, a $c^*$ consumes the least amount of energy to finish a task. However, when the budget is reasonably large, a $P + c^*$ always consumes the least amount of energy. We explain these relationships as follows: When the power budget is small, a $c^*$ can finish the task quickly owing to more processing power. As the power budget increases, this benefit diminishes because of the performance saturation resulting from its low sequential performance. This effect continues to degrade the $c^*$ as the budget increases and eventually causes the *Perf/W* of a $c^*$ to become even worse than that of a $P^*$.

Similarly, Figures 5c and 5f show that the *Perf/J* of a $c^*$ is the highest only when the power budget is low and the task is embarrassingly parallel ($f = 0.9$). However, as the power budget increases, the *Perf/J* of a $c^*$ many-core is worse than that of the other designs. Instead, a $P + c^*$ is the most power scalable. Due to its high sequential performance along with energy-efficient parallel computation capability, it achieved the highest *Perf/J*. To better understand the design spectrum, we also performed several sensitivity studies with different sizes of $c$ and with different relationships between the performance and the power using these models. These studies showed similar trends.

Extending Amdahl's law to take power and energy into account, our analysis clearly demonstrates that a symmetric many-core processor can easily lose its energy efficiency as the number of cores increases. To achieve the best possible energy efficiency, our work suggests a many-core alternative, featuring many small, energy-efficient cores integrated with a full-blown processor. Our analytical models also show that by knowing the amount of parallelism available in an application prior to execution, we can find the optimal number of active cores for maximizing performance for a given cooling capacity and energy in a system. To further optimally control the number of active cores adaptively, future many-core runtime must be capable of dynamic per-core power profiling and have a feedback mechanism to manage thread dispatch.


### Acknowledgments

We thank Mark Hill of the University of Wisconsin for his feedback and encouragement on an early version of this article. This work was sponsored in part by National Science Foundation CAREER Award CNS-0644096.


### References


1. L. Hammond, B.A. Nayfeh, and K. Olukotun, "A Single-Chip Multiprocessor," *Computer*, Sept. 1997, pp. 79-85.
2. J. Held, J. Bautista, and S. Koehl, "From a Few Cores to Many: A Tera-Scale Computing Research Overview," white paper, Intel; http://download.intel.com/research/platform/terascale/terascale_overview_paper.pdf.
3. W.-M. Hwu et al., "Implicitly Parallel Programming Models for Thousand-Core Microprocessors," *Proc. 44th Design Automation Conf.* (DAC 07), ACM Press, 2007, pp. 754-759.
4. G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. Am. Federation of Information Processing Soc. Spring Joint Computer Conf.* (AFIPS 07), AFIPS Press, 1967, pp. 483-485.
5. T. Mudge, "Power: A First-Class Architectural Design Constraint," *Computer*, Apr. 2001, pp. 52-58.
6. A. Duller, G. Panesar, and D. Towner, "Parallel Processing: The picoChip Way!" *Proc. Communicating Process Architectures*, 2003, IOS Press, pp. 125-138.

7. T.R. Halfhill, "Massively Parallel Digital Video," *Microprocessor Report*, 9 Jan. 2006.
8. H.P. Hofstee, "Power-Efficient Processor Architecture and the Cell Processor," *Proc. 11th Ann. Symp. High-Performance Computer Architecture* (HPCA 05), IEEE CS Press, 2005, pp. 258-262.
9. D.H. Woo et al., "POD: A 3D-Integrated Broad-Purpose Acceleration Layer," *IEEE Micro*, vol. 28, no. 4, 2008, pp. 28-40.
10. R. Gonzalez and M. Horowitz, "Energy Dissipation in General-Purpose Microprocessors," *IEEE J. Solid-State Circuits*, vol. 31, no. 9, 1996, pp. 1277-1284.
11. M.D. Hill and M.R. Marty, "Amdahl's Law in the Multicore Era," *Computer*, July 2008, pp. 33-38.
12. F.J. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," *Proc. IEEE/ACM 32nd Int'l Symp. Microarchitecture* (MICRO 32), keynote address, IEEE CS Press, 1999, p. 2.

***Dong Hyuk Woo*** *is a PhD student in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include energy-efficient many-core architectures. Woo received an MS in electrical and computer engineering from the Georgia Institute of Technology. He is a student member of the IEEE and the ACM. Contact him at dhwoo@ece.gatech.edu.*

***Hsien-Hsin S. Lee*** *is an associate professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include computer architecture, cybersecurity, and 3D integration. Lee received a PhD in computer science and engineering from the University of Michigan at Ann Arbor. He is a senior member of the IEEE. Contact him at leehs@gatech.edu.*