Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

ENERGY COROLLARIES TO AMDAHL'S LAW

Analyzing the Interactions Between Parallel Execution and Energy Consumption

By Sangyeun Cho, Michael Moeng, and Rami Melhem {10/6/08-01}

Department of Computer Science, University of Pittsburgh

A key objective of today's multicore processor designs is to increase performance without a proportional increase in energy consumption. Single-core processors became overly complex, and their per-watt performance deteriorated. Multicore processors try to ameliorate the problem by speeding up a workload with parallel processing at a lower clock frequency (and lower voltage). But what is the precise relationship between parallel processing and energy consumption?

In 2004, Intel's Shekhar Borkar suggested that a perfect two-way parallelization would lead to half the clock frequency (and voltage), one-quarter the energy consumption, and one-eighth the power density, when compared with sequential execution of the same program in the same execution time. (See "Microarchitecture and Design Challenges for Gigascale Integration.")[1]

In this article, we explore how much energy savings is possible with parallel processing if processors can dynamically change their voltage and frequency.[2] This article is based on our paper, "Corollaries to Amdahl's Law for Energy,"[3] from *IEEE Computer Architecture Letters* in January 2008.

We will address several questions: What is the maximum energy improvement to be gained from parallelization? How can we determine the processor speed to achieve that improvement? How does static power affect the energy-optimal program speedup and energy consumption? Given a target speedup, how do we set the processor speeds to minimize energy? Our exploration uses the same simple application model as the well-known Amdahl's law—parallel applications having a serial section and a parallel section whose ratio is known.

Reviewing Amdahl's Law

Amdahl's law provides a simple yet extremely useful method for predicting the potential performance of a parallel computer, given the ratio of the serial and parallel work in a program and the number of processors available. It has been widely applied in determining speedups, even for single-processor systems. It's also known as the law of diminishing returns. The following equation succinctly describes Amdahl's law:

$$Speedup = \frac{1}{s + p/N} \quad (1)$$

where, $(s + p) = 1$, s (p) is the ratio of the serial (parallel) work in the program, and N is the number of processors. We begin with the same input parameters as in Amdahl's law, namely $s(p)$ and N. Then we derive the minimum energy consumption one would get with optimal frequency allocation to the serial and parallel regions in a program while the execution time is unchanged. We obtain:

$$Improvement\ in\ Dynamic\ Energy = \frac{1}{\left(s + \frac{p}{N^{(\alpha-1)/\alpha}}\right)^{\alpha}} \quad (2)$$

when the dynamic power consumption of a processor running at clock frequency f is proportional to f^{α}. In literature describing dynamic voltage and frequency scaling, α is

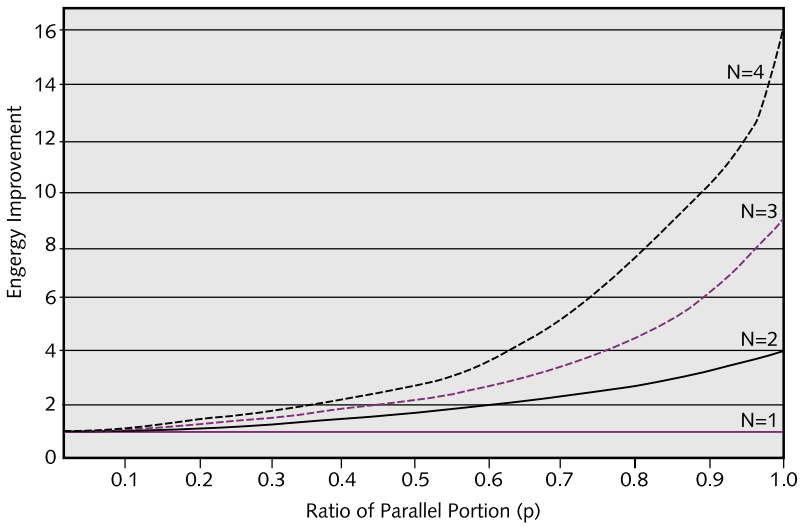

Figure 1. Achievable dynamic energy improvement assuming $\alpha = 3$ and using 1, 2, 3, and 4 processors, given the ratio of serial and parallel work in a program.

between 2 and 3—typically 3. Equation (2) suggests that more parallelism (larger p) and more processors (larger N) help reduce energy consumption. Figure 1 is a plot of equation (2).

Formulating the Problem

For the purposes of these calculations, we assume that processors can run at arbitrary clock frequencies, subject to a maximum frequency, F_{max}. Using Amdahl's law in equation (1) as a basis, the speedup (x) one would achieve with parallelization and frequency scaling is subject to the following:

$$x \le \frac{1}{s + p/N} \tag{3}$$

For the sake of simplicity, we normalize the sequential execution time of the program as 1. Similarly, we normalize the amount of work (i.e., number of cycles) in the program as 1. Therefore, the maximum clock frequency (F_{max}) has a relative speed of 1. The amount of work in the serial portion of the program is represented by s, and the parallel portion by p (or $1-s$). Figure 2 shows this arrangement.

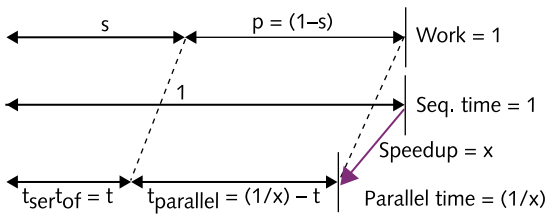

Figure 2. Normalized "work" and "time." The "parallel time" is partitioned into serial and parallel regions. The time for the serial region is t, and the time for the parallel region is the parallel time $(1/x)$, less the time for the serial region, t.

We also assume that the dynamic power consumption of a processor running at F_{max} is normalized to 1, and that static power consumption is λ. That is, the ratio of static power to dynamic power at F_{max} is λ. Our simple assumption about static power consumption allows us to reveal its effect in closed-form derivations.

Clock frequencies for the two regions in the work, namely s and p, are calculated as follows:

$$f_s = \frac{s}{t} \tag{4}$$

$$f_p = \frac{1-s}{(\frac{1}{x}) - t \cdot N} \tag{5}$$

In these equations, we assume, for simplicity, that the execution time of a program region is determined by the amount of work (e.g., s) and the processor speed (e.g., f_s). In reality, some program operations (such as main memory access) have latencies unrelated to the processor speed.

For a given problem, s is fixed, and for a given architecture, N and λ are fixed. Hence, the energy consumption, E, is a function of t and x. Specifically:

$$E(t,x) = t \cdot f_s^{\alpha} + N \cdot \left(\frac{1}{x} - t\right) \cdot f_p^{\alpha} + N \cdot \lambda \cdot \frac{1}{x} \tag{6}$$

In equation (6), the three terms represent energy for the serial portion, energy for the parallel portion, and energy for static power consumption during the whole execution time, respectively. We assume that dynamic power consumption of a processor running at f is f^{α}. We do not consider the processor temperature as a factor. Hence, the term for static energy is the product of the per-processor power consumption rate, λ, the number of processors, N, and the total execution time.

Energy Improvement With Parallelization

Let's consider the question of maximum energy improvement with parallel processing, and which clock frequency achieves it. We start with a special case, the problem of obtaining the minimum energy consumption when x is 1. That is, the program execution time is identical to that of sequential execution at the maximum processor speed. The condition $x = 1$ is similar to setting a deadline (= sequential execution time) by which to finish the computation. Of course, one may get larger energy savings by further decreasing x. With the condition $x = 1$, we can rewrite equation (6) as:

$$E(t) = t \cdot \left(\frac{s}{t}\right)^{\alpha} + N \cdot (1-t) \cdot \left(\frac{1-s}{(1-t) \cdot N}\right)^{\alpha} + N \cdot \lambda \tag{7}$$

From equation (7), we can derive t^*, the value of t that minimizes energy consumption, by setting $dE(t)/dt$ to 0. We get:

$$t^* = \frac{s}{s + p/N^{(\alpha-1)/\alpha}} \tag{8}$$

Now we obtain the values of f_s and f_p to minimize $E(t)$, using equations (4), (5), and (8). Specifically:

$$f_s^* = \frac{s}{t^*} = s + \frac{p}{N^{(a-1)/a}} \quad (9)$$

$$f_p^* = \left(s + \frac{p}{N^{(a-1)/a}}\right) \cdot N^{\frac{1}{a}} \quad (10)$$

$$= f_s^* / N^{\frac{1}{a}} \quad (11)$$

Both f_s^* and f_p^* are a function of s and N in equations (9) and (10). Equation (11) shows the relationship between f_s and f_p when $E(t)$ is minimized. Interestingly, the ratio between the two frequencies, f_s^* / f_p^*, is a function of N, but not of s. Equation (11) suggests that to achieve minimum energy consumption, we need to slow the clock frequency in the parallel region of the program by $N^{1/\alpha}$ compared with the frequency in the serial region. Figure 3 illustrates this relationship.

Finally, from equations (7) and (8), we obtain the minimum energy consumption:

$$E_{min} = E(t^*) = \left(s + \frac{p}{N^{\frac{(\alpha-1)}{\alpha}}}\right)^{\alpha} + N \cdot \lambda \quad (12)$$

Here, the first term shows dynamic energy consumption, and the second term expresses static energy consumption. Equation (2) is simply taken from equation (12). Figure 1 depicts the maximum energy improvement owing to parallelization (E_{min}^{-1}) when the number of processors varies from 1 to 4, $\alpha = 3$ and $\lambda = 0$. It's clear that energy improvement is a function monotonically increasing with p and N. Figure 3 shows how the overall energy ($E(t)$) changes as we adjust t. It also presents t^*, the value of t that minimizes $E(t)$. Note that the optimal solution obtained for f_s^* and f_p^* is feasible, because both clock frequencies are less than the processor's maximum frequency, $F_{max} = 1$.

Determining the Effect of Static Power

Amdahl's law explores the effect of parallelization on speedup, and we have described the effect of parallelization on energy consumption when the program's execution time is unchanged (i.e., $x = 1$). However, depending on the rate of static power consumption, the minimum amount of total energy consumption may not occur at $x = 1$. If static power consumption is high, the processor will use the least energy at a higher clock frequency, possibly resulting in $x > 1$. On the other hand, if static power consumption is low, the processor will use the minimum energy at a slower clock frequency, $x < 1$.

Let's revisit the problem of minimizing total energy consumption without restricting x. For this, we set the derivatives of equation (6) with respect to both t and x to zero. We obtain the following:

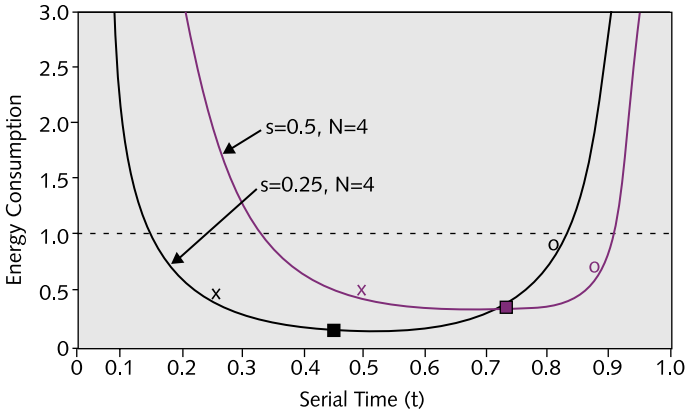

Figure 3. Dynamic energy consumption vs. t (serial time) for two cases, $s = 0.25$ and $s = 0.5$ when $N = 4$. The bound of t is marked with X (when $f_s = F_{max} = 1$) and O (when $f_p = F_{max} = 1$). The minimum energy point in each curve (at $t = t^*$) is marked with a filled rectangle.

$$t^* = \left(\frac{\alpha - 1}{\lambda N}\right)^{\frac{1}{\alpha}} \cdot s \quad (13)$$

$$x^* = \left(\frac{\lambda N}{\alpha - 1}\right)^{\frac{1}{\alpha}} \cdot \left(\frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}}\right) \quad (14)$$

With t^* and x^*, we can use equations (4) and (5) to calculate the optimum frequencies:

$$f_s^* = \left(\frac{\lambda N}{\alpha - 1}\right)^{\frac{1}{\alpha}} \quad (15)$$

$$f_p^* = \left(\frac{\lambda}{\alpha - 1}\right)^{\frac{1}{\alpha}} = f_s^* / N^{\frac{1}{\alpha}} \quad (16)$$

from which we can compute the minimum energy. An interesting observation is that at f_s^* and f_p^*, the dynamic energy is given by the following:

$$E_{dynamic} = t^* \cdot f_s^{\alpha} + N \cdot \left(\frac{1}{x^*} - t\right) \cdot f_p^{\alpha} \quad (17)$$

$$= \frac{1}{\alpha - 1} \cdot \frac{N\lambda}{x^*} \quad (18)$$

which is equal to $\frac{1}{\alpha - 1}$ of the static energy, $E_{static} = \frac{N\lambda}{x^*}$. In other words, total energy consumption is minimized when the dynamic energy consumption is $\frac{1}{\alpha - 1}$ times the static energy consumption. This relation holds during the execution of both the serial and parallel sections of the program.

The above solution is applicable only if both f_s^* and f_p^* are less than than F_{max}, however, necessitating that $\lambda N \le \alpha - 1$. If the ratio between static and dynamic power (λ) is large, we can't maintain the aforementioned relationship between static and dynamic energy. In that case, we should set $f_s = 1$ and

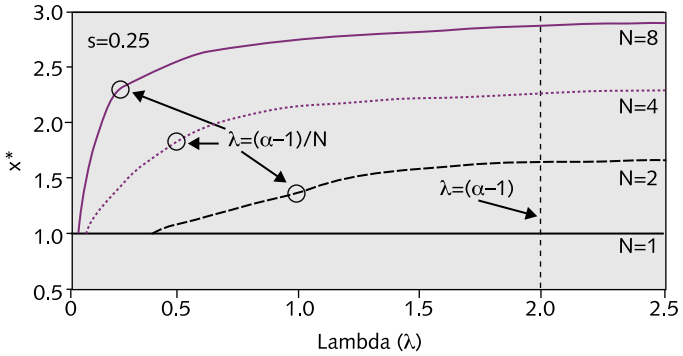

Figure 4. λ changes x^*, the speedup of a program when its energy consumption is minimized. x^* saturates at the maximum speedup that Amdahl's law dictates when $\lambda > \alpha - 1$. We assume that $\alpha = 3$.

find the values of x and f_p that minimize total energy consumption. Denoting these values by x^{**} and f_p^{**}, we obtain the following:

$$x^{**} = \frac{1}{s + (\frac{p}{N}) \cdot (\frac{\alpha - 1}{\lambda})^{\frac{1}{\alpha}}} \quad (19)$$

$$f_p^{**} = (\frac{\lambda}{\alpha - 1})^{\frac{1}{\alpha}} \quad (20)$$

Again, these values result in dynamic power consumption's being $\frac{1}{\alpha - 1}$ times the static power consumption during execution of the parallel portion of the program.

Finally, if static power consumption is so high that $\lambda > \alpha - 1$, then the minimum energy is obtained when $f_s = f_p = 1$. That is, energy consumption is at the minimum when the processors run at their maximum speed to finish the task as quickly as possible. Figure 4 summarizes the relationship between λ and the speedup that results in minimum energy consumption. In this figure, the values of λ are divided into three regions. When $\lambda \le \frac{\alpha - 1}{N}$, the solution for the optimum energy consumption problem is given by equations (14), (15), and (16).

When $\frac{\alpha - 1}{N} < \lambda \le \alpha - 1$, the solution is given by $f_s = 1$, in equations (19) and (20). When $\lambda > \alpha - 1$, the solution is given by $f_s = f_p = 1$, and the speedup is that given by Amdahl's law in equation (1).

Figure 5 depicts the improvement ratio of the minimum energy at different program speedups, relative to the baseline sequential execution of a given application. This plot clearly demonstrates that a smaller λ leads to a larger energy-improvement ratio at any selected program speedup. Moreover, the greatest energy-improvement ratio occurs at a lesser program speedup. In other words, one can slow the clock speed further to benefit from reducing dynamic energy to a greater degree before static energy starts to offset the benefit, if λ is small.

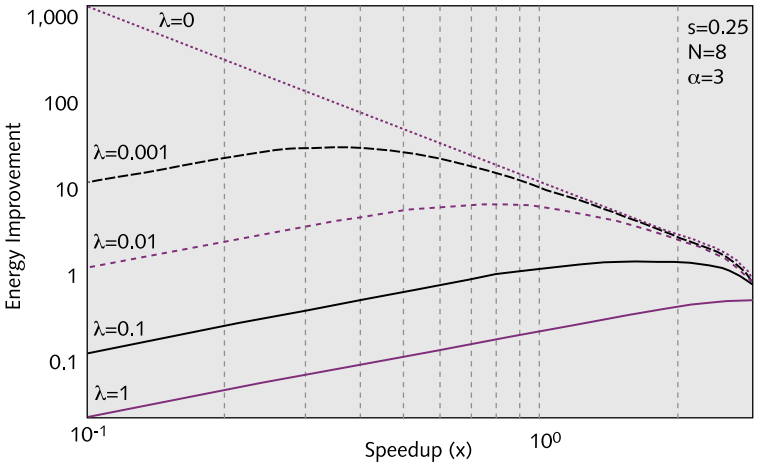

Figure 5. Energy improvement at different speedups, compared with sequential execution.

Energy-Performance Trade-offs

So far, we have focused on the problem of obtaining the minimum energy consumption with specific processor speeds (hence program speedup) given p, λ, and N. We have largely ignored the program's performance. In this section, we will consider the trade-offs between program performance and energy consumption. The main question is how to set the clock frequency f_s and f_p for a specified degree of performance (x).

Because the static energy $N \cdot \lambda \cdot \frac{1}{x}$ is immediately determined, given x, we need only minimize the dynamic energy while meeting the program speedup requirement. Our solution is derived from equations (4), (5), (14), (15), and (16) as follows:

$$If\ x \le \frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}},\ f_s^* = x f_{s,x=1}^*, f_p^* = x f_{p,x=1}^* \quad (21)$$

$$If\ \frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}} < x \le \frac{1}{s + \frac{p}{N}},\ f_s^* = 1, f_p^* = \frac{px}{N(1 - sx)} \quad (22)$$

where $f_{s,x=1}^*$ and $f_{p,x=1}^*$ are the optimal frequencies when $x = 1$ in equations (9) and (10). We call the speedup interval in equation (21) the *linear frequency scaling interval*, because the energy-optimal f_s and f_p can be obtained by simply scaling $f_{s,x=1}^*$ and $f_{p,x=1}^*$ by a factor of x. Note that the upper bound of the condition in equation (21) is equivalent to $\lambda N \le \alpha - 1$.

Figure 6 shows how the minimum energy consumption changes as we target a different program speedup. This figure also shows the contributions of dynamic and static energy consumption. Notice that the dynamic energy of the sequential region saturates at around $x = 2.3$. That's because f_s cannot scale

beyond F_{max}. Finally, when $f_s = f_p = 1$ (i.e., the maximum speedup), dynamic energy is 1—the same as for sequential execution.

Another way to make the trade-off between energy consumption and program performance is to minimize the energy-delay product rather than the total energy. The energy-delay product ED is as follows:

$$ED(t,x) = (t \cdot f_s^{\alpha} + N \cdot \left(\frac{1}{x} - t\right) \cdot f_p^{\alpha} + N \cdot \lambda \cdot \frac{1}{x}) \cdot \frac{1}{x} \quad (23)$$

Through similar analysis we get the following:

$$t^* = \left(\frac{\alpha - 2}{2N\lambda}\right)^{1/\alpha} \cdot s \quad (24)$$

$$x^* = \left(\frac{2N\lambda}{\alpha - 2}\right)^{1/\alpha} \cdot \left(\frac{1}{s + \frac{p}{N^{(\alpha-1)/\alpha}}}\right) \quad (25)$$

$$f_s^* = \left(\frac{2N\lambda}{\alpha - 2}\right)^{1/\alpha} \quad (26)$$

$$f_p^* = \left(\frac{2\lambda}{\alpha - 2}\right)^{1/\alpha} \quad (27)$$

It is interesting to find that $f_p^* = f_s^* / N^{\frac{1}{\alpha}}$ as in equation (16). In fact, by comparing the values obtained for t^*, y^*, f_s^*, and f_p^* with equations (13), (14), (15), and (16), we observe that they are the same equations, replacing $(\alpha - 1)$ with $(\alpha - 2)/2$. This similarity also appears in the calculation of energy. Specifically, we can compute the dynamic energy when ED is minimized:

$$E_{dynamic} = \frac{2}{\alpha - 2} \cdot N\lambda \cdot \frac{1}{x^*} \quad (28)$$

which is equal to $2/(\alpha - 2)$ of the static energy, $E_{static} = N\lambda \cdot \frac{1}{x^*} \cdot$

Adding the Overhead of Parallelization

Analytical models of computer performance sacrifice accuracy for simplicity. Some aspects of a computer system are not modeled, and one might wonder about the effect of these missing properties. In this section, we examine the impact of one such property—synchronization overhead.

Synchronization overhead is a broad term we use to describe any overhead incurred when using multiple processors together. It includes time spent communicating by using locks or barriers, waiting because of uneven task completion, or stalling because of contention for shared resources. This overhead has been examined in the context of scientific computing using supercomputers. The importance of synchronization overhead in conventional computers grows as core counts increase.

The model we use to represent synchronization overhead adds a synchronization function to the parallel portion: $p \Rightarrow p(1 + \sigma(n))$. This is in addition to the regular work the processor does, and it increases with the number of cores. Although the exact form of synchronization depends on both the CPU or system architecture and the program executed, prior work has found $\sigma(n) = c \cdot \log(n)$ to be close for many architectures. (See H.P. Flatt's "A Simple Model for Parallel Processing."[4]) Figure 7 is a plot similar to that of Figure 1 using that estimation, with: $c = 0.08$.

After using the same derivation process described earlier, we discovered something unintuitive: the relation between f_s and f_p when minimizing energy consumption is independent of the synchronization function. Although this might seem surprising, it makes sense when considering that the relative serial and parallel portions of work do not factor into the relation between f_s and f_p either—the synchronization cost effectively raises the portion of parallel work (along with the total work).

Conclusions

We have considered the problem of minimizing total energy consumption for a given architecture (values of $N, \alpha,$ and λ) and a given problem with a known ratio of parallelizable work (value of p). We have analytically derived the formula for the processor speeds that minimize energy consumption

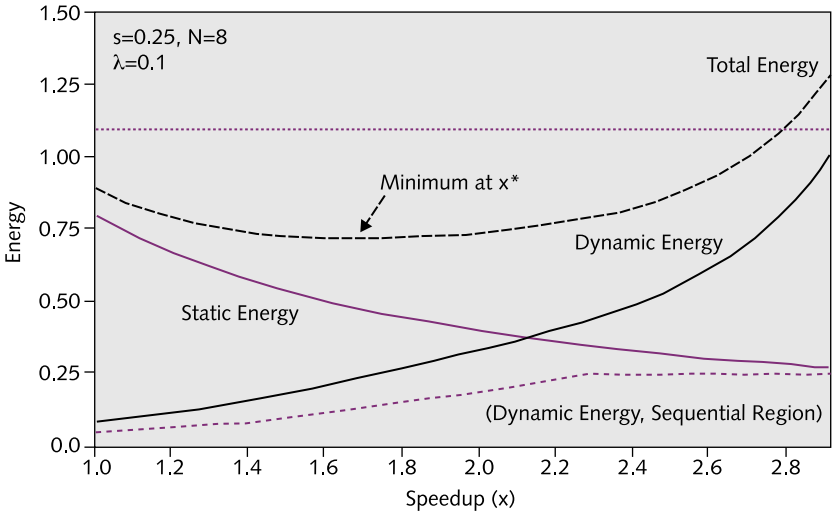

Figure 6. Optimal energy, given the speedup of x, with $\alpha = 3$. Total energy is the sum of dynamic and static energy. This plot also shows dynamic energy for the sequential region. The thick dotted line shows the sequential machine's energy consumption. Given these parameters, the maximum speedup (Amdahl's law) is 2.909, and $x^* = 1.684$, according to equation (14). The dynamic energy of the sequential region saturates at 2.286, according to equation (21).

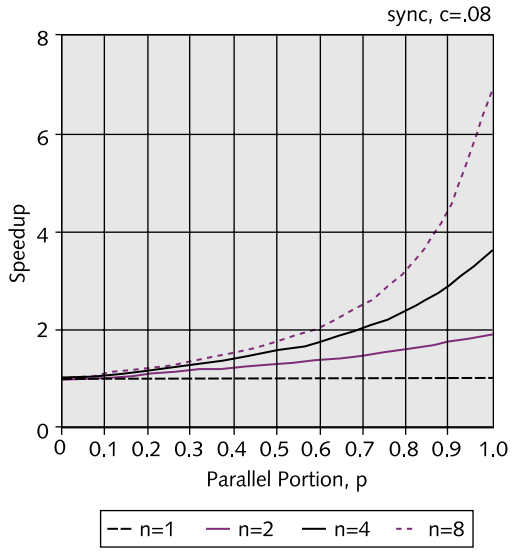

Figure 7. Achievable dynamic energy improvement, after accounting for a synchronization overhead of $0.08 \log n$. As in Figure 1, $\alpha = 3$.

and have shown that at those speeds, dynamic energy consumption is equal to $\frac{1}{\alpha - 1}$, the static energy consumption. Hence, to minimize energy, this relation between static and dynamic energy should be maintained—as long as the processor does not exceed its maximum allowable clock frequency. In that case, the maximum speed should be used.

In many systems, it is desirable to strike a trade-off between energy consumption and performance by minimizing the energy-delay product rather than the total energy. Our results show that the optimal energy-delay is obtained when $f_s = N^{\frac{1}{\alpha}} f_p$ and $f_p = (\frac{2\lambda}{\alpha - 2})^{\frac{1}{\alpha}}$, and $\alpha > 2$. Our results also show that the frequency relation of $f_p^* = f_s^* / N^{\frac{1}{\alpha}}$ allows us to optimize both energy and the energy-delay product.

Our formulas also show that, for a given processor implementation (λ and α), the minimum total energy is a monotonically decreasing function of the number of processors, N, as long as the parallel section of code can be executed on N processors. Hence, from the viewpoint of total energy consumption, all available processors should be running. However, note that this result—and all results in this article—assume that the N processors in the system consume static power, even when executing serial code. If individual processors can be turned off and back on with low overhead, then the formula for total energy in equation (6) should be changed such that $N \cdot \lambda \cdot \frac{1}{x}$ is replaced by $(t + (\frac{1}{x} - t) \cdot N) \cdot \lambda$.

In this case, our analysis indicates that the minimum total energy is independent of the number of processors used for executing the parallel section of a program. The energy-delay product is minimized when the maximum number of available processors executes the parallel code. The minimum amount of energy is consumed when clock speeds are equal during the serial and parallel sections, which again results in static energy equaling $(\alpha - 1)$, the dynamic energy.

[Editor's note: Sangyeun Cho is an assistant professor in the Department of Computer Science at the University of Pittsburgh in Pennsylvania. Michael Moeng is a graduate student in that department. Dr. Rami Melhem is the department chairman and a professor of computer science.]

References

1. S. Borkar. "Microarchitecture and Design Challenges for Gigascale Integration," keynote at the International Symposium on Microarchitecture (MICRO), December 2004.
2. F. Yao, A. Demers, and S. Shenker. "A Scheduling Model for Reduced CPU Energy," *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pp. 374–382, October 1995.
3. S. Cho and R. Melhem. "Corollaries to Amdahl's Law for Energy," *IEEE Computer Architecture Letters* (CAL), 7(1):25–28, January 2008.
4. H. P. Flatt. "A Simple Model for Parallel Processing," *IEEE Computer*, 17(11):95, November 1984.