Appears in Intl. Symp. on Performance Analysis of Systems and Software (ISPASS), Austin, TX, Mar. 2005

# Power-Performance Implications of Thread-level Parallelism on Chip Multiprocessors

Jian Li and José F. Martínez

Computer Systems Laboratory
Cornell University
Ithaca, NY 14853 USA

http://m3.csl.cornell.edu/

## ABSTRACT

We discuss power-performance implications of running parallel applications on chip multiprocessors (CMPs). First, we develop an analytical model that, for the first time, puts together parallel efficiency, granularity, and voltage/frequency scaling, to quantify the performance and power consumption delivered by a CMP running a parallel code. Then, we conduct detailed simulations of parallel applications running on a power-performance CMP model. Our experiments confirm that our analytical model predicts power-performance behavior reasonably well. Both analytical and experimental models show that parallel computing can bring significant power savings and still meet a given performance target, by choosing granularity and voltage/frequency levels judiciously. The particular choice, however, is dependent on the application's parallel efficiency curve and the process technology utilized, which our model captures. Likewise, analytical model and experiments show the effect of a limited power budget on the application's scalability curve. In particular, we show that a limited power budget can cause a rapid performance degradation beyond a number of cores, even in the case of applications with excellent scalability properties. On the other hand, our experiments show that power-thrifty memory-bound applications can actually enjoy better scalability than more "nominally scalable" applications (i.e., without regard to power) when a limited power budget is in place.

## 1 INTRODUCTION

Low-power computing has long been an important design objective for mobile, battery-operated devices. More recently, however, power consumption in high-performance microprocessors has drawn considerable attention from industry and researchers as well. Traditionally, power dissipation in CMOS technology has been significantly lower than other technologies, such as TTL or ECL. However, at current speeds and feature sizes, CMOS power consumption has increased dramatically. This makes microprocessor cooling increasingly difficult and expensive [2, 14]. As a result, over the last few years, power has become a first-priority concern to microprocessor designers/manufacturers [1, 39].

In light of this mounting problem, industry and researchers are eyeing chip multiprocessor architectures (CMPs). CMPs can attain higher performance by running multiple threads in parallel. By integrating multiple cores on a chip, designers hope to deliver performance growth while depending less on raw circuit speed, and thus power [1].

Earlier VLSI works have discussed the trade-offs that sequential vs. parallel circuits present in silicon area and power consumption [4, 32]. There is also rich literature on power/thermal-aware simultaneous multithreading (SMT) and CMP designs (or similar architecture configurations), most of which focuses on multiprogrammed workloads [8, 12, 24, 28, 36, 37]. But so far, very little work has been done on the power-performance issues involving *parallel applications* executing on multiprocessors in general, and on multicore chips in particular.

In this paper, we investigate the power-performance issues of running parallel applications on a CMP. First, we develop an analytical model to study the effect of the number of processors used, the parallel efficiency, and the voltage/frequency scaling applied, on the performance and power consumption delivered by a CMP. Specifically, we look at (1) optimizing power consumption given a performance target, and (2) optimizing performance given a certain power budget. Then, to confirm the insights developed from the analytical model and assess its limitations, we conduct detailed simulations of parallel applications running on a power-performance model of a CMP.

## 2 ANALYTICAL STUDY

In this section, we develop an analytical model as a first approximation to studying the power-performance trade-offs specific to parallel computation on CMPs. First, we lay out the basic power and performance equations used in the model. Then, we study two important scenarios:

(1) power optimization given a performance target; and (2) performance optimization under a budget constraint.

## 2.1 Basic Equations

### Power

Our power model is based on two well-known equations for CMOS logic [23, 31]:

$$f_{\text{max}} = \eta \, \frac{(V - V_{\text{th}})^\alpha}{V} \tag{1}$$

$$P = P_{\text{D}} + P_{\text{S}} = A\,C\,V^2\,f + V I_{\text{leak}} \tag{2}$$

Eq. 1 establishes the relationship between the supply voltage $V$ and the maximum operating frequency $f_{\text{max}}$, where $V_{\text{th}}$ is the threshold voltage, and $\eta$ and $\alpha$ are experimentally derived constants. Eq. 2 defines power consumption $P$ as the sum of dynamic and static components, $P_{\text{D}}$ and $P_{\text{S}}$, respectively. In the dynamic term, $A$ is the gate activity factor, $C$ is the total capacitance, $V$ is the supply voltage, and $f$ is the operating frequency. In the static term, $V$ is again the supply voltage, and $I_{\text{leak}}$ is the leakage current.

Leakage current mainly consists of subthreshold and gate-oxide leakage [23], $I_{\text{leak}} = I_{\text{sub}} + I_{\text{ox}}$. We comment on each one in turn.

$$I_{\text{sub}} = K_1\,W\,e^{\frac{-V_{\text{th}}\,q}{n\,k\,(T+273)}}\,(1 - e^{\frac{-V\,q}{k\,(T+273)}})$$

where $K_1$ and $n$ are experimentally derived constants, $W$ is the gate width, $V_{\text{th}}$ is threshold voltage, $k = 1.38 \times 10^{-23}\text{J/K}$ is Boltzmann's constant, $T$ is the temperature in degrees Celsius, $q = 1.6 \times 10^{-19}\text{C}$ is the electron unit charge, and V is the supply voltage.

$$I_{\text{ox}} = K_2\,W\,(\frac{V}{H_{\text{ox}}})^2\,e^{\frac{-\gamma\,H_{\text{ox}}}{V}}$$

where $K_2$ and $\gamma$ are experimentally derived constants, $H_{\text{ox}}$ is the oxide thickness, and $V$ is the supply voltage.

Thus, leakage current is largely exponentially dependent on $V$ and $T$. In our leakage current model, we approximate this dependency using a curve-fitted formula as follows:

$$\begin{aligned} I_{\text{leak}} &= I_{\text{leak, Vn, Tstd}}\,e^{a_1 + a_2\,V + a_3\,T + a_4\,V\,T} \\ &= I_{\text{leak, Vn, Tstd}}\,e^{\phi(V,T)} \end{aligned} \tag{3}$$

where $I_{\text{leak, Vn, Tstd}}$ is the leakage current at nominal supply voltage $V_{\text{n}}$ and room temperature $T_{\text{std}}$ (25°C), and $a_1$, $a_2$, $a_3$ and $a_4$ are curve-fitting constants. We use $\phi(V,T)$ to abbreviate the dependency of this curve-fitted formula on supply voltage and temperature.

We validate our curve-fitted formula using HSpice simulations on a chain of inverters. In the HSpice simulations, we independently vary the supply voltage from a minimum of $2.3\,V_{\text{th}}$ (to maintain enough noise margin [19]) to $V_{\text{n}}$, and the operating temperature from 25°C to 110°C for two process technologies: 130nm and 65nm. The results show that the maximum error is within 9.5% and 7.5% for 130nm and 65nm, respectively (0.25% and 0.05% average error, respectively).

Finally, we use the curve-fitted expression in Eq. 3 to replace $I_{\text{leak}}$ in Eq. 2 as follows:

$$P = A\,C\,V^2\,f + V I_{\text{leak}, V_{\text{n}}, T_{\text{std}}}\,e^{\phi(V,T)} \tag{4}$$

Our analytical model assumes a fixed CMP configuration built out of identical cores. Each core would comprise a microprocessor, some private cache(s), and a proportional part of the chip's interconnect. In this analytical model, in order to keep the discussion manageable, we assume a constant activity factor across the chip. This effectively excludes resources that have significantly lower switching activity and may occupy a large chip area, such as a L2 cache. In the experimental study we make no such simplifications (Section 3).

### Performance

To model performance, we use the formula for execution time proposed in [15]:

$$t = IC \cdot CPI \cdot f^{-1} \tag{5}$$

where $IC$ is the dynamic instruction count, $CPI$ is the average number of cycles per instruction, and $f$ is the operating frequency. Applications may run sequentially on one processor, or in parallel on $N$ processors (up to the maximum number of cores on the chip). When running in parallel on $N$ processors, we assume that all threads exhibit identical instruction counts and average number of clock cycles per instruction, $IC_N$ and $CPI_N$, respectively. Furthermore, we assume all processors use the same supply voltage $V_N$ and operating frequency $f_N$.

The *parallel efficiency* [7] of an application running on $N$ processors $\varepsilon(N)$ can be written as:

$$\varepsilon(N) = \frac{t_1}{N\,t_N} = \frac{IC_1\,CPI_1\,f_1^{-1}}{N\,IC_N\,CPI_N\,f_N^{-1}}$$

If $f_N = f_1$ (i.e., no frequency scaling is applied), we can rewrite the parallel efficiency as:

$$\varepsilon_{\text{n}}(N) = \frac{IC_1\,CPI_1}{N\,IC_N\,CPI_N} \tag{6}$$

We call this *nominal parallel efficiency* $\varepsilon_{\text{n}}(N)$. It is a useful characterization of the application's parallel behavior on the CMP architecture, independent of power considerations.[1] On the one hand, it captures the over-

[1]We assume that $CPI_N$ does not depend on the clock frequency. In a system with multiple clocks (e.g., off-chip memory), this would require that any frequency scaling be applied to all clocks proportionally. Our experimental study does not make this simplification (Section 3).

heads of a parallel configuration, such as the communication overheads between threads, which can result in suboptimal performance gains ($\varepsilon_n(N) < 1$). On the other hand, it also captures the beneficial effects of a parallel setup, such as increased aggregate caching capacity, which may result in superlinear performance gains ($\varepsilon_n(N) > 1$).

Using these basic equations, we present models for two important scenarios: power optimization given a performance target (Section 2.2), and performance optimization under a power budget constraint (Section 2.3).

## 2.2 Scenario I: Power Optimization

In this scenario, the goal is to find the configuration that maximizes power savings while delivering a prespecified level of performance. In particular, we require that all configurations deliver the performance of a sequential execution on one processor at full throttle, $t_1 = t_N$. Thus, using Eq. 5, and for any number of processors $N$, we can write:

$$IC_1\ CPI_1\ f_1^{-1} = IC_N\ CPI_N\ f_N^{-1}$$

Using the definition of nominal parallel efficiency (Eq. 6), we can rewrite this equality as follows:

$$f_N = \frac{f_1}{N\,\varepsilon_n(N)} \tag{7}$$

We can also rewrite Eq. 4 for the parallel case as follows:

$$P_N = A\,N\,C\,V_N^2\,f_N + V_N\,N\,I_{\text{leak},V_n,T_{\text{std}}}\,e^{\phi(V_N,T_N)} \tag{8}$$

where $N\,C$ and $N\,I_{\text{leak},T_{\text{std}}}$ represent the aggregate capacitance and standard leakage current of the $N$-processor configuration, respectively. If we define the *voltage scaling ratio* $\vartheta = V_N/V_1$, we can rewrite Eq. 8 using $\vartheta$ and Eq. 7 as follows:

$$\begin{aligned} P_N &= A\,N\,C\,(\vartheta\,V_1)^2\frac{f_1}{N\,\varepsilon_n(N)} \\ &\quad +(\vartheta\,V_1)\,N\,I_{\text{leak},T_{\text{std}}}\,e^{\phi(\vartheta\,V_1,T_N)} \\ &= \frac{\vartheta^2}{\varepsilon_n(N)}P_{D,1} + \vartheta\,N\,e^{\phi(\vartheta\,V_1,T_N)}\,P_{S,1,T_{\text{std}}} \end{aligned} \tag{9}$$

where $P_{D,1}$ and $P_{S,1,T_{\text{std}}}$ are the dynamic and static power consumption at $N = 1$, nominal voltage and frequency, and room temperature. We resort to published data from the International Technology Roadmap for Semiconductors (ITRS) [19] to obtain $V_1$, $V_{\text{th}}$, $f_1$, $P_{D,1}$, and $P_{S,1,T_{\text{std}}}$. We contemplate two process technologies, 130nm and 65nm, and set the operating temperature of the single-core configuration at $T_1 = 100°$C. We assume voltage can scale continuously from the nominal supply voltage $V_1$ down to $2.3V_{\text{th}}$. Frequency can scale continuously from $f_1$ without a lower bound (always positive, of course). We set $A = 1$, $\alpha = 2$ [31]. We can obtain $\vartheta$ by solving the following equality derived from Eqs. 1 and 7:

$$\frac{(\vartheta\,V_1 - V_{\text{th}})^2}{\vartheta} = \frac{(V_1 - V_{\text{th}})^2}{N\,\varepsilon_n(N)}$$

Using a 32-way CMP baseline, we study configurations running on different numbers of processor cores, assuming that unused processor cores in each case are shut down. In each configuration, we approximate the operating temperature $T_N$ using the HotSpot thermal model [38] for its default Alpha EV6 floorplan, similarly to how we use it in our experimental study (Section 3).

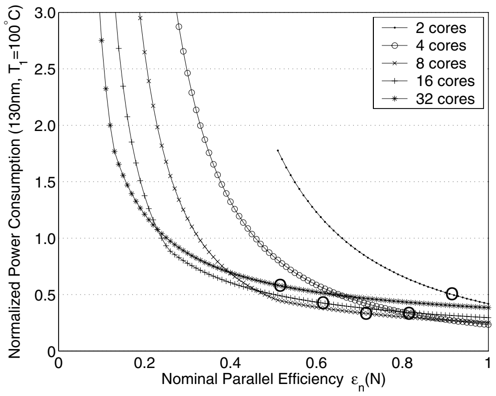
We plot the normalized power consumption $P_N/P_1$ for each combination of process technology and operating temperature $T_1$, using $N = \{2, 4, 8, 16, 32\}$ processors, and nominal parallel efficiency ranging from $1/N$ to 1 in each case.

Notice that, when $\varepsilon_n(N) < \frac{1}{N}$, there is no way for the $N$-processor configuration to achieve the same performance as the single-processor one without raising $V_N$ over $V_1$. We do not allow this in the model. Also, although we do not plot regions with superlinear speedup ($\varepsilon_n(N) > 1$), the results would be in line with the plots.

The curves (Fig. 1) show that, for any $N$, higher nominal parallel efficiency $\varepsilon_n(N)$ allows for greater power savings. Indeed, Eq. 7 shows that, as efficiency goes up, a lower frequency $f_N$ is required to maintain the performance of the single-core configuration. Generally, this also allows a lower supply voltage $V_N$ (Eq. 1). As a result, for a fixed $N$, higher efficiency results in lower dynamic and static power consumption (Eq. 8). Furthermore, a reduction in the dynamic activity results in lower die temperature $T_N$, which further reduces static power consumption (Eq. 8). This is captured in our analysis through the HotSpot thermal model [38].

Recall, however, that to maintain acceptable noise margin, $V_N$ may not decrease below $2.3V_{\text{th}}$. Thus, for a given $N$, there is an $\varepsilon_n(N)$ beyond which the decrease in frequency is not accompanied by a decrease in supply voltage. This results in diminished returns on both dynamic and static power savings (Eq. 8). Moreover, the returns on static power savings are also limited by another lower bound: The die temperature can never be lower than the ambient temperature. In the plots, this is reflected by a change in the curvature.

In any case, for the configurations used, all curves show power savings with respect to the single-core configuration beyond a certain $\varepsilon_n(N)$. Moreover, in general, the plots show that configurations with higher $N$ require a lower level of efficiency to reach their power breakeven points. This is in agreement with Eq. 7: a higher



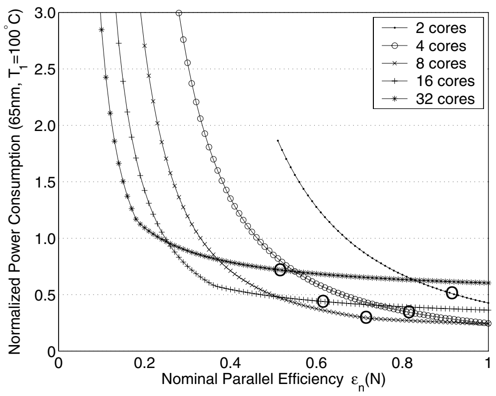


Figure 1: Normalized power consumption using 130nm (top) and 65nm (bottom) technologies at $T_1 = 100°$C, varying $N = \{2, 4, 8, 16, 32\}$ and $\varepsilon_n(N) \in [1/N, 1]$, and forcing that all configurations yield the performance of $N = 1$ at full throttle. The marks in each plot indicate working points for a sample application with $\varepsilon_n = \{0.9, 0.8, 0.7, 0.6, 0.5\}$.

$N$ makes $f_N$ (and thus power consumption) drop more rapidly as $\varepsilon_n(N)$ increases.

Notice that, because $f_N$ drops more rapidly with larger $N$ (since we maintain the same performance target), voltage scaling can be more aggressive. This, however, also implies that the lower bound in supply voltage (and temperature) is reached (approached) earlier in the efficiency scale. As a result, scaling may not compensate for the extra power consumption introduced by the additional cores. This is reflected in the plots by the fact that high-$N$ curves run above low-$N$ ones at high efficiency points. Overall, applications that exhibit a high degree of parallel efficiency do not necessarily save more power by choosing a larger $N$.

In any case, recall that $\varepsilon_n(N)$ depends on $N$, and applications typically exhibit different levels of parallel efficiency with different $N$. (Usually, efficiency is lower with higher $N$ due to communication overhead.) Thus, for a given application, it generally does not make sense to directly compare the curves at a fixed point on the efficiency scale. To illustrate this, we mark in the plots the operating points of an imaginary application that exhibits $\varepsilon_n = \{0.9, 0.8, 0.7, 0.6, 0.5\}$ for $N = \{2, 4, 8, 16, 32\}$, respectively. As we can see, the configuration that yields the maximum power savings is not necessarily the one with the highest number of processors, largely because of the decrease in parallel efficiency. This observation holds for both feature sizes.

Finally, it is important to note that, because absolute performance and power consumption is different in each of the two feature sizes, caution is advised when making comparisons across the plots.

## 2.3 Scenario II: Performance Optimization

In this scenario, the goal is to find the configuration that maximizes performance under a constrained power budget. In particular, we set the maximum power budget to that of executing on one processor at full throttle, $P_1 = P_N$.

The performance gain or *speedup* $S$ on $N$ processors [7] can be expressed as:

$$S = \frac{t_1}{t_N} = \frac{IC_1\ CPI_1\ f_1^{-1}}{IC_N\ CPI_N\ f_N^{-1}} = N\,\varepsilon_n(N)\,\frac{f_1^{-1}}{f_N^{-1}}$$

Using Eq. 1 and the definition of voltage ratio $\vartheta$ (Section 2.2), we can rewrite the speedup as:

$$S = N\,\varepsilon_n(N)\,\frac{(\vartheta\,V_1 - V_{th})^\alpha}{\vartheta\,(V_1 - V_{th})^\alpha} \tag{10}$$

To compute $\vartheta$, we introduce the problem restriction $P_1 = P_N$. Using Eqs. 1, 4, and 8, we can express this restriction as:

$$\begin{aligned} & A\,C\,V_1^2\,\eta\,\frac{(V_1 - V_{th})^\alpha}{V_1} + V_1\,I_{\text{leak},V_n,T_{std}}\,e^{\phi(V_1,T_1)} \\ = \quad & A\,N\,C\,(\vartheta\,V_1)^2\,\eta\,\frac{(\vartheta\,V_1 - V_{th})^\alpha}{\vartheta\,V_1} \\ & +(\vartheta\,V_1)\,N\,I_{\text{leak},V_n,T_{std}}\,e^{\phi(\vartheta\,V_1,T_N)} \end{aligned}$$

By using transformations similar to those used in Eq. 9, we obtain the equality:

$$\begin{aligned} & P_{D,1} + e^{\phi(V_1,T_1)}\,P_{S,1,T_{std}} \\ = \quad & N\,\vartheta\,\frac{(\vartheta\,V_1 - V_{th})^\alpha}{(V_1 - V_{th})^\alpha}\,P_{D,1} + \vartheta\,N\,e^{\phi(\vartheta\,V_1,T_N)}\,P_{S,1,T_{std}} \end{aligned} \tag{11}$$

After obtaining $\vartheta$ from Eq. 11, we can resolve speedup $S$ in Eq. 10.

Fig. 2 shows speedups for up to 32 cores under our constant power budget constraint. In the plot, we assume the application's nominal parallel efficiency $\varepsilon_n(N)$ is 1 for any $N$. As before, we use two process technologies, 130nm and 65nm.

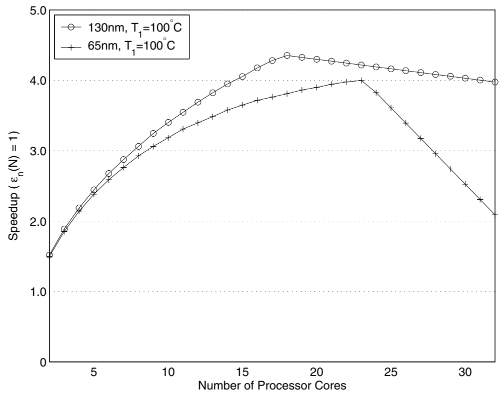


Figure 2: Speedup of $N$-processor configurations with $\varepsilon_{\rm n}(N) = 1$, and power budget in all cases equal to the power consumption of the single-processor configuration at full throttle.

It is interesting to observe that, even if the application's nominal parallel efficiency $\varepsilon_{\rm n}(N)$ is 1, the maximum speedup achieved across all configurations is only a little over 4, which corresponds to the configuration with $N = 18$ and 130nm process technology. Indeed, as the number of cores increases, the total power budget ought to remain constant. As a result, voltage/frequency scaling must be applied to the cores, which slows them down. The result is a suboptimal speedup with respect to the sequential execution.

Still, the speedup grows with the number of cores for relatively small configurations. For larger $N$, however, both technologies eventually show *decreasing* speedups. This is the case beyond $N = 18$ and $N = 23$ for 130nm and 65nm, respectively. It means that, for even a perfectly scalable application, obtaining the optimum performance within a certain power budget may require a number of processors *lower* than the maximum number available. Indeed, due to the limited range of voltage scaling (Section 2.2), once the supply voltage reaches $2.3V_{\rm th}$ in our model, only frequency scaling can be applied further as the number of processors goes up. Unfortunately, dynamic power consumption is only linearly dependent on $f$, and thus the frequency reduction needed for each processor added is significant. This results in a significant performance degradation, most notably in the 65nm case, where the ITRS data attributes a higher fraction of the total power consumption to static power [19].

Note that, despite the speedup curve for 65nm being below that of 130nm at all times, the absolute performance of the 65nm case might well be above that of 130nm one. This is because each curve is relative to the performance of its respective sequential case. Also, note that, in this analysis, we tacitly assume that the application runs at maximum power in the sequential case—in other words, the power consumption by the sequential setup *is* the power budget. In reality, however, applications may consume less than that when running sequentially, for example, if they stall often on memory accesses. In that case, as we assign more processors to the application, it would be possible to consume *more* than the power consumed by the sequential setup, and still be within budget. This may result in an extra performance "boost" beyond the gain predicted by the model. Thus, depending on the application characteristics, the speedups shown in the plots may be somewhat pessimistic. We do consider this effect in the experimental study (Section 3).

| | |
|---|---|
| CMP Size | 16-way |
| Processor Core | Alpha 21264 [6] |
| Process Technology | 65nm |
| Nominal Frequency | 3.2GHz |
| Nominal $V_{\rm dd}$ | 1.1v [19] |
| $V_{\rm th}$ | 0.18v [19] |
| Ambient Temperature | 45°C |
| Die Size | 244.5mm$^2$ (15.6mm × 15.6mm) |
| L1 I-, D-Cache | 64KB, 64B line, 2-way, 2-cycle RT |
| Unified L2 Cache | Shared on chip, 4MB, 128B line, 8-way, 12-cycle RT |
| Memory | 75ns RT |

Table 1: The CMP configuration modeled in the experiments. In the table, RT stands for round-trip.

## 3 EXPERIMENTAL SETUP

To confirm the insights developed from the analytical model in Section 2 and assess its limitations, we conduct detailed simulations of parallel applications running on a detailed power-performance model of a CMP. In this section, we discuss the architecture modeled, the applications, and the power model.

### 3.1 Architecture

Our study uses a detailed model of a 16-processor CMP. CMP cores are modeled after the Alpha 21264 (EV6) processor [6]. (While it is conceivable to consider a heterogeneous CMP model, its applicability and performance impact in the context of a parallel execution is unclear and beyond the scope of this paper.) Each processor core has private L1 instruction and data caches. All cores share a 4MB on-chip L2 cache through a common bus, and implement a MESI cache coherence protocol [7]. Table 1 lists relevant cache and memory parameters.

We choose a 65nm process technology. The original EV6 ran at 600MHz on a 350nm process technology; by proceeding similarly to [24], we determine the clock frequency of our 65nm EV6 cores to be 3.2GHz. We set nominal supply and threshold voltages at 1.1v and 0.18v, respectively [19], and in-box ambient air temperature at 45°C [29, 38]. Using CACTI [40], we obtain an estimated chip area of 244.5mm$^2$ (15.6mm × 15.6mm), using a scaling method similar to [25].

For the sake of simplicity, we assume global voltage/frequency scaling for the entire chip. (While it is

| *Application* | *Problem Size* |
|---|---|
| Barnes-Hut | 16K particles |
| Cholesky | tk15.O |
| FFT | 64K points |
| FMM | 16K particles |
| LU | $512 \times 512$ matrix, $16 \times 16$ blocks |
| Ocean | $514 \times 514$ ocean |
| Radiosity | room -ae 5000.0 -en 0.05 -bf 0.1 |
| Radix | 1M integers, radix 1024 |
| Raytrace | car |
| Volrend | head |
| Water-Nsq | 512 molecules |
| Water-Sp | 512 molecules |

Table 2: Applications from the SPLASH-2 suite used in the experiments.

conceivable to allow each core to run at a different frequency, the applicability and performance impact in the context of a parallel execution is nontrivial and beyond the scope of this paper.) Frequency can scale from 3.2GHz down to 200MHz, and we resort to [18] to establish the relationship between frequency and supply voltage. Notice that, because voltage/frequency scaling is applied at the chip level, on-chip latencies (e.g., on-chip cache hit time) do not vary in terms of cycles. However, a round trip to (off-chip) memory takes the same amount of time regardless of the voltage/frequency scaling applied on chip, and thus the round-trip memory latency in processor cycles goes down as we downscale voltage and/or frequency. This is unlike the analytical model, where a system-level voltage/frequency scaling is assumed (Section 2).

## 3.2 Applications

We use all twelve applications from the SPLASH-2 suite [41]. The problem size of each application is at least as large as the suggested size in [41], and does not change with the number of cores. Table 2 lists the applications and their execution parameters. For all applications, we skip initialization and then simulate to completion.

## 3.3 Power Model

We use Wattch to model the switching activity and dynamic power consumption of the on-chip functional blocks. This is different from our analytical model (Section 2), where we assume a constant activity factor $A = 1$ for all on-chip circuitry (Eq. 2). As for static power consumption, we model it as a fraction of the dynamic power consumption [5, 38]. In our model, this fraction is exponentially dependent on the temperature [5]. The average operating temperature (over the chip area) in our model ranges from in-box ambient air temperature (45°C) to a maximum operating temperature of 100°C, in agreement with multiple contemporary processor chip designs. We use the HotSpot thermal model [38] for chip temperature estimation.

Wattch is reasonably accurate in relative terms; however, the absolute power values can be off by a nontrivial amount [24]. Because we use power values to communicate across two different tools (Wattch and HotSpot), we ought to ensure we do so in a meaningful way. We achieve this by renormalizing power values as follows.

We use HotSpot to determine the maximum operational power consumption (dynamic+static), which is the one that yields the maximum operating temperature of 100°C. Then, using the dynamic/static ratio that corresponds to that temperature [5], we derive the dynamic component.

We now need to establish the connection with Wattch. To do so, we use a compute-intensive microbenchmark to recreate a quasi-maximum power consumption scenario at nominal voltage and frequency levels in our simulation model, and obtain Wattch's dynamic power value. This number is often different from the one obtained through HotSpot using the method explained above. To overcome this gap, we calculate the ratio between Wattch and HotSpot's dynamic power values, and use it throughout the experiments to renormalize wattage obtained with Wattch in our simulations as needed. This makes it possible for both tools to work together. While the absolute power may again not be exact, the results should be meaningful in relative terms. Using both tools, plus the power ratio/temperature curve, we are able to connect dynamic and static power consumption with temperature for any voltage and frequency scaling point.

Finally, we notice that the temperature and power density of the shared L2 cache is significantly lower than the rest of the chip across all the applications studied. Reasons include: much less switching activity; aggressive clock gating in the model [3]; and large L2 cache dissipation area. This observation is in agreement with published work by others [5, 8]. To obtain meaningful figures of power density and temperature, we exclude L2 from the calculations. However, we do include the power consumption of L2 in the results for total power consumption.

## 4 EVALUATION

In this section we evaluate two scenarios, which correspond to the two scenarios discussed in the analytical model (Section 2).

## 4.1 Scenario I: Power Optimization

This scenario is analogous to Scenario I of the analytical model (Section 2.2). The goal is to find the configuration that maximizes power savings while delivering the same performance as the sequential execution at nominal voltage and frequency levels.

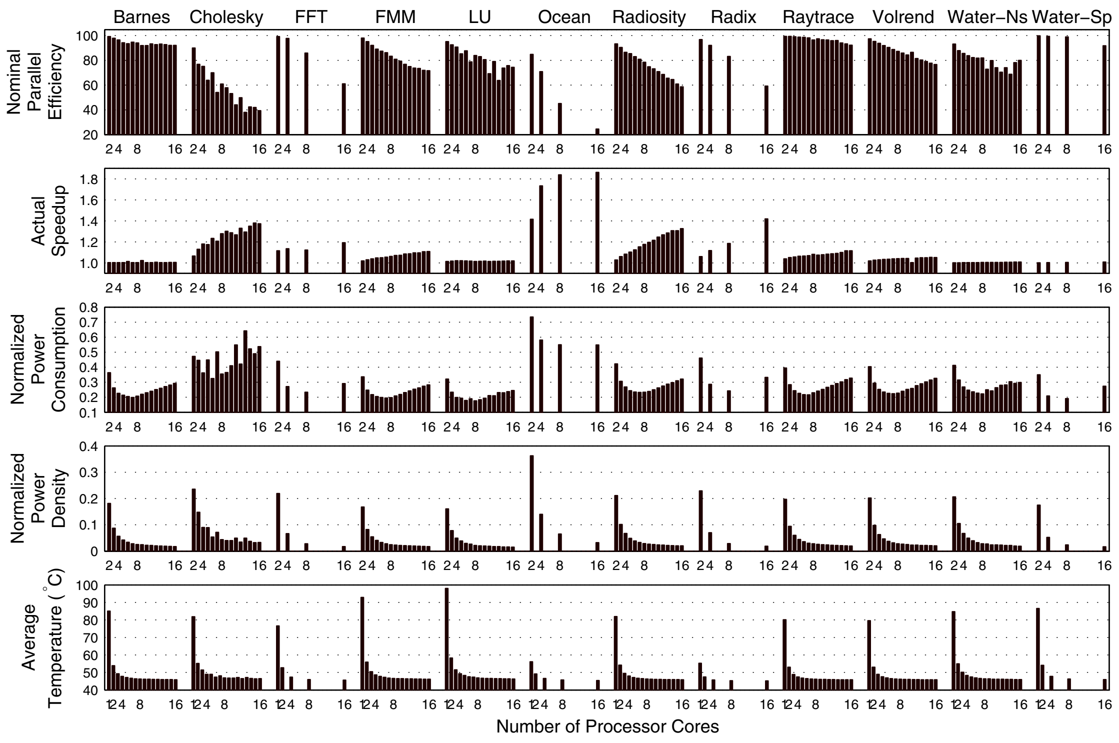


Figure 3: Performance, power, and thermal characteristics of a 16-way CMP running the SPLASH-2 applications [41], according to the restrictions discussed in Scenario I (Section 4.1). Missing bars indicate that the corresponding application does not run with such a number of processors.

The simulation experiment is conducted conceptually similarly to Section 2.2: We first simulate the execution of all twelve SPLASH-2 applications on a number of processor cores ranging from one to sixteen, at nominal frequency and voltage levels (without regard to any power budget). Notice that some SPLASH-2 applications only work for a number of cores power of two, and in that case we only study configurations with one, two, four, eight, and sixteen cores. From this, we obtain: (1) the nominal parallel efficiency curve for each application; and (2) the power consumption of each application on the single-core configuration. The nominal parallel efficiency is then used to calculate the target frequency of each configuration (Eq. 7). The target voltage supply is extrapolated from [18] in each case. Then, we recalculate the simulation parameters that are sensitive to voltage/frequency scaling, and run simulations again to collect power and performance statistics for processor cores. Figure 3 shows the simulation results. As in Section 2, we assume unused processor cores are turned off.

The first plot shows the nominal parallel efficiency values that we obtain during the profiling phase. We observe wide changes in nominal parallel efficiency levels, both within (as we change $N$) and across applications. In general, within an application, nominal parallel efficiency goes down as the number of processor cores goes up.

The second plot shows the actual speedups attained by the applications on the different configurations studied. That applications experience speedups seems counter-intuitive, since the fundamental premise of this scenario is that all configurations are tuned to yield equal performance (that of the single-core configuration in each case). This is most noticeable with Ocean and, to a lesser extent, Cholesky and Radiosity. One reason why this is the case is that, as the number of processors increases and voltage/frequency scaling is applied to the chip (but not to off-chip memory), the processor-memory speed gap narrows, which benefits memory-bound applications. This is not captured by the analytical model in Section 2.2, which assumes system-wide voltage/frequency scaling.

The third plot shows, for each application, the power consumption in configurations of various $N$, normalized to the power consumption of the single-core configuration in each case. Given sufficient parallel efficiency, power consumption can be effectively reduced as the number of participating cores increases. Poor scalability, however, can make using more cores counter-productive. Indeed, the plot shows that, in general, a diminishing parallel efficiency resulting from increasing the number of processors eventually causes power savings to stagnate and, if even more cores are used, to actually recede. Our analytical model correctly predicts this behavior (Figure 1).

There are three main reasons behind this: First, the limited voltage range constrains the potential for power savings in relatively large configurations (Section 2). Second, as the parallel efficiency diminishes with increasing number of participating cores, the architecture cannot afford too aggressive voltage/frequency levels if it is to meet the baseline performance. Third, the total static power consumption may increase as the number of processor cores increases (Eq. 9).

Another way to examine the effect of parallelization on power is to look at the power density. The fourth plot shows, for each application, the average power density for each $N$, normalized to the average power density for $N = 1$ in each case. As we increase $N$, power density decreases rapidly as a result of aggressive voltage scaling (and consequent temperature drop). For example, in our experiments, we observe an across-the-board power density reduction of around 95% at $N = 8$. As we approach the lower bounds of voltage and temperature, however, the reduction in power density becomes significantly slower.

Finally, the fifth plot shows the average operating temperature of the applications running on different configurations, including the single-core configurations. (Notice that, unlike the other four plots, this one is not normalized.) As a result of lower power density, the average operating temperature decreases as $N$ increases. Larger temperature reduction is observed in applications that consume more power at nominal operating voltage and frequency levels (FMM and LU). This is because of the exponential relation between temperature and static power. Again, the temperature reduction rate, initially very sharp, quickly slows down as $N$ increases.

Overall, we see that parallel computing can be an effective way to reduce chip power consumption, power density and operating temperature on CMP given a performance target, although its effectiveness is very dependent on factors such as parallel efficiency, range of voltage/frequency scaling, and total leakage power, all of which tend to worsen as the number of participating cores goes up.

## 4.2 Scenario II: Performance Optimization

This scenario is analogous to the one in Section 2.3. In this case, we want to see the maximum speedup a $N$-processor configuration can achieve within the power budget of a single core, which we derive using microbenchmarking (Section 3.3). The simulation experiment is set up with off-line profiling, similarly to Section 4.1. In addition to obtaining profile information at the nominal frequency (3.2GHz), we obtain the power

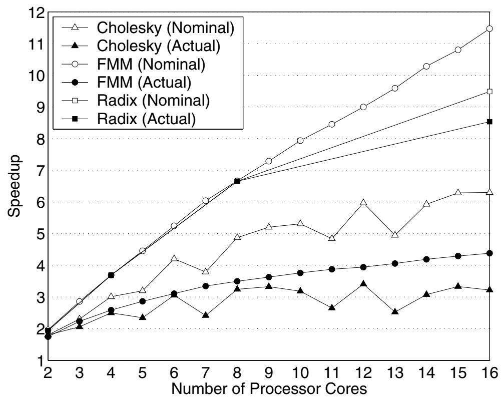


Figure 4: Comparison of the nominal and actual speedup of three SPLASH-2 applications. The power budget is the maximum nominal power consumption of a single core (Section 3.3).

and performance statistics with frequencies ranging from 200MHz to 3.0GHz, with a step of 200MHz. Then we use the profile information to calculate the optimal voltage and frequency in order to achieve the maximum speedup under the fixed power dissipation budget for each $N$-processor configuration. The configuration values that fall between any two profiled values are approximated by linearly scaling between the two. Then, using this information, we run simulations again to get the real speedup.

We select three applications in this case study: FMM, Cholesky, and Radix, in descending order of computational intensity and power consumption. Both the nominal and the actual speedups of the three applications are presented in Figure 4. (The nominal speedup is derived from the nominal parallel efficiency values calculated in Section 4.1.)

As predicted by the analytical model (Section 2.3), we see a performance gap between the nominal configuration without considering power dissipation budget (nominal speedup) and the configuration that abides by the power budget constraint (actual speedup). The gap is most significant in the compute-intensive application (FMM), and least so for Radix, which is memory-bound. In general, two situations help the performance scaling of memory-bound applications: First, as we apply voltage/frequency scaling to the chip—but not to off-chip memory—, the processor-memory speed gap narrows, resulting in less memory stalls. Second, due to stalling on memory accesses, these applications usually do not reach the chip's power budget when running sequentially. Thus, as we increase $N$, these applications can benefit from higher voltage/frequency levels, and still abide by the power budget. As discussed previously, none of these effects is captured by the analytical model (Section 2). In fact, for up to eight processor cores, the actual speedup of Radix does match the nominal speedup (Fig. 4). When looking at our data for these configurations, we find that they operate at nominal voltage and frequency levels. Indeed, the nominal power consumption of Radix is low enough that it allows up to eight-core configurations to run at nominal voltage and frequency without exceeding our power budget. However, as we continue increasing $N$, a gap between actual and nominal speedup eventually appears, as the configurations can no longer meet power budget constraints at nominal voltage and frequency levels.

Finally, notice that, for these memory-bound applications and low $N$, one could seek higher performance by *overclocking* the chip, and still abide by the power budget. However, unless the memory subsystem is also overclocked, the resulting increase in the processor-memory speed gap could partially offset the potential performance gain.

## 5 RELATED WORK

Earlier VLSI works have discussed the trade-offs that sequential vs. parallel circuits present in silicon area and power consumption [4, 32]. But so far, very little work has been done on the power-performance issues involving parallel applications executing on multiprocessors in general, and on multicore chips in particular.

There is rich literature on power/thermal-aware simultaneous multithreading (SMT) and CMP designs (or similar architecture configurations), most of which focuses on multiprogrammed workloads [8, 12, 24, 28, 27, 36, 37]. In contrast, our work focuses on the power-performance issues of CMPs in the context of parallel applications.

Huh et al. [17] conduct an in-depth exploration of the design space of CMPs. However, they do not address power. More recently, Ekman and Stenström [9] conduct a design-space study of CMPs in which they address some power issues. Assuming a certain silicon budget, they compare chips with different numbers of cores, and correspondingly different core sizes. They argue that parallel applications with limited scalability but some instruction-level parallelism may run better on CMPs with few, wide-issue cores. They also argue that CMPs with few, wide-issue cores and with many, narrow-issue cores consume roughly the same power, as cache activity offsets savings at the cores. Our work assumes a given chip design, and explores the issues of assigning different numbers of cores to a parallel application, given certain power-performance constraints. Also, their work uses $0.18\mu$m process technology, and does not consider voltage/frequency scaling, which is a fundamental component in our work.

Grochowski et al. [13] discuss trade-offs between microprocessor processing speed vs. throughput in a power-constrained environment. They postulate that a microprocessor that can achieve both high scalar performance and high throughput performance ought to be able to dynamically vary the amount of energy expended to process each instruction, according to the amount of parallelism available in the software. To achieve this, they survey four techniques: dynamic voltage/frequency scaling (DVFS), asymmetric cores, variable-sized cores, and speculation control, and conclude that a combination of DVFS and asymmetric cores is best. In our work, we formally connect granularity of parallelism, application's parallel efficiency, and DVFS to work two scenarios on a symmetric CMP: performance optimization under a power budget constraint, and also power optimization given a performance target.

Kaxiras et al. [22] compare the power consumption of an SMT and a CMP digital signal processing chip for mobile phone applications. They do not explicitly study parallel applications in the "traditional" sense. For example, they approximate a parallel encoder with four independent MPEG encoder threads, each thread processing one quarter of the original image size. A speech encoder and a speech decoder are connected in a pipelined fashion to a channel encoder and decoder, respectively. The issues of granularity vs. parallel efficiency and DVFS that we address cannot be easily conveyed in this context.

Kadayif et al. [20] propose to shut down idle processors in order to save energy when running nested loops on a CMP. The authors also study a pre-activation strategy based on compiler analysis to reduce the wake-up overhead of powered-off processors. Although they address program granularity and power, they do not exploit DVFS in their solution, which is fundamental in our work.

In a different work, Kadayif et al. [21] propose to use DVFS to slow down lightly loaded threads, to compensate for load imbalance in a program and save power and energy. They use the compiler to estimate the load imbalance of array-based loops on single-issue processor cores. The authors also mention the opportunity of further energy savings by using less than the number of available processor cores using profile information. However, the connection of DVFS to granularity and parallel efficiency of the code is not fleshed out, and the processor model is too simple for our purpose.

In the context of cache-coherent shared-memory multiprocessors, Moshovos, et al. [30] reduce energy consumption by filtering snoop requests in a bus-based parallel system. Saldanha and Lipasti [35] observe significant potential of energy savings by using serial snooping for load misses. Li et al. [26] propose to save energy wasted in barrier spin-waiting, by predicting a processor's stall time and, if warranted, forcing it into an appropriate ACPI-like low-power sleep state. This work is complementary to ours in that it does not consider the number of processors, and does not attack power consumption during useful activity by the application.

In an environment of loosely-coupled web servers running independent workloads, several studies evaluate different policies to control the number of active servers (and thus their performance level) to preserve power while maintaining acceptable quality of service [10, 11, 33, 34].

In the context of micro-architectures, Heo and Asanović [16] study the effectiveness of pipelining as a power-saving tool in a uniprocessor. They examine the relationship between the logic depth per stage and the supply voltage in deep submicron technology under different conditions. This is complementary to our work, since we study power-performance issues of using multiple cores on a CMP.

## 6 CONCLUDING REMARKS

In this paper, we have explored power-performance issues of running parallel applications on a CMP. We have developed an analytical model to study the effect of combining granularity, parallel efficiency, and voltage/frequency scaling on the performance and power consumption delivered by a CMP. To confirm the insights developed from the analytical model and assess its limitations, we have conducted detailed simulations of parallel applications running on a power-performance model of a CMP. The experiments confirm that the analytical model captures the power-performance behavior reasonably well.

Both analytical and experimental models show that, through judicious choice of granularity and voltage/frequency scaling, parallel computing can bring significant power savings while meeting a given performance target. The particular choice, however, is dependent on the application's parallel efficiency curve and the process technology utilized, which our model captures.

Similarly, analytical model and experiments show the effect of a limited power budget on the application's scalability curve. In particular, we have shown that a limited power budget can cause a rapid performance degradation beyond a number of cores, even in the case of applications with excellent scalability properties. On the other hand, our experiments have shown that power-thrifty memory-bound applications can actually enjoy better scalability than more nominally scalable applications when a limited power budget is in place.

Overall, our study concludes that, under the right circumstances, parallel computing may bring significant power savings over a uniprocessor setup of similar performance. It also illustrates the dependency on the application's parallel efficiency curve and process technology when pursuing the configuration that maximizes performance within a certain power budget, which makes the choice nontrivial.

## ACKNOWLEDGMENTS

We thank David Albonesi, Rajit Manohar, and the anonymous reviewers for useful feedback. This work was supported in part by NSF grant CCF-0429922 and gifts from Intel. Jian Li's support was provided in part by NSF grants ACI-0406345 and ACI-0121401.

## REFERENCES

[1] T. Agerwala. Computer architecture: Challenges and opportunities for the next decade. In *International Symposium on Computer Architecture*, München, Germany, June 2004. (Keynote presentation).

[2] S. Borkar. Design challenges for technology scaling. *IEEE Micro*, 19(4):23–29, July–Aug. 1999.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *International Symposium on Computer Architecture*, pages 83–94, Vancouver, Canada, June 2000.

[4] A. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, Apr. 1992.

[5] P. Chaparro, J. González, and A. González. Thermal-effective clustered microarchitectures. In *Workshop on Temperature-Aware Computer Systems*, München, Germany, June 2004.

[6] Compaq Computer Corporation, Shrewsbury, Massachusetts. *Alpha 21264 Microprocessor Hardware Reference Manual*, July 1999.

[7] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1999.

[8] J. Donald and M. Martonosi. Temperature-aware design issues for SMT and CMP architectures. In *Workshop on Complexity-Effective Design*, München, Germany, June 2004.

[9] M. Ekman and P. Stenström. Performance and power impact of issue-width in chip-multiprocessor cores. In *International Conference on Parallel Processing*, pages 359–368, Kaohsiung, Taiwan, Oct. 2003.

[10] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Workshop on Power Aware Computing Systems*, pages 179–196, Cambridge, MA, Feb. 2002.

[11] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy consevation policies for web servers. In *USENIX Symposium on Internet Technologies and Systems*, Seattle, WA, Mar. 2003.

[12] S. Ghiasi and D. Grunwald. Design choices for thermal control in dual-core processors. In *Workshop on Complexity-Effective Design*, München, Germany, June 2004.

[13] E. Grochowski, R. Ronen, J. Shen, and H. Wang. Best of both latency and throughput. In *International Conference*

*on Computer Design*, pages 236–243, San Jose, CA, Oct. 2004.

[14] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, Q1 2001.

[15] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier Science Pte Ltd., third edition, 2003.

[16] S. Heo and K. Asanović. Power-optimal pipelining in deep submicron technology. In *International Symposium on Low Power Electronics and Design*, Newport Beach, CA, Aug. 2004.

[17] J. Huh, D. Burger, and S. W. Keckler. Exploring the design space of future CMPs. In *International Conference on Parallel Architectures and Compilation Techniques*, pages 199–210, Barcelona, Spain, Sept. 2001.

[18] Intel Corporation. *Intel Pentium M Processor on 90nm Process with 2-MB L2 Cache Datasheet*, June 2004.

[19] The ITRS Technology Working Groups. *International Technology Roadmap for Semiconductors (ITRS)*, http://public.itrs.net.

[20] I. Kadayif, M. Kandemir, and U. Sezer. An integer linear programming based approach for parallelizing applications in on-chip multiprocessors. In *IEEE/ACM Design Automation Conference*, pages 703–708, New Orleans, LA, June 2002.

[21] I. Kadayif, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Exploiting processor workload heterogeneity for reducing energy consumption in chip multiprocessors. In *Design, Automation and Test in Europe*, pages 1158–1163, Paris, France, Feb. 2004.

[22] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing power consumption of an SMT and a CMP DSP for mobile phone workloads. In *International Conference on Compilers, Architecture, and Systhesis for Embedded Systems*, pages 211–220, Atlanta, Georgia, Nov. 2001.

[23] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *IEEE Computer*, 36(12):68–75, Dec. 2003.

[24] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *International Symposium on Microarchitecture*, pages 81–92, San Diego, CA, Dec. 2003.

[25] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *International Symposium on Computer Architecture*, pages 64–75, München, Germany, June 2004.

[26] J. Li, J. F. Martínez, and M. C. Huang. The Thrifty Barrier: Energy-aware synchronization in shared-memory multiprocessors. In *International Symposium on High-Performance Computer Architecture*, pages 14–23, Madrid, Spain, Feb. 2004.

[27] Y. Li, D. Brooks, Z. Hu, and K. Skadron. Performance, energy, and temperature considerations for SMT and CMP architectures. In *International Symposium on High-Performance Computer Architecture*, San Francisco, CA, Feb. 2005.

[28] Y. Li, K. Skadron, D. Brooks, and Z. Hu. Understanding the energy efficiency of simultaneous multithreading. In *International Symposium on Low Power Electronics and Design*, pages 207–212, Newport Beach, CA, Aug. 2004.

[29] R. Majan. Thermal management of CPUs: A perspective on trends, needs and opportunities. In *International Workshop on Thermal Investigations of ICs and Systems*, Madrid, Spain, Oct. 2002. Keynote presentation.

[30] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary. JETTY: Filtering snoops for reduced energy consumption in SMP servers. In *International Symposium on High-Performance Computer Architecture*, pages 85–96, Nuevo Leone, Mexico, Jan. 2001.

[31] T. Mudge. Power: A first-class architectural design constraint. *IEEE Computer*, 34(4):52–58, Apr. 2001.

[32] K. K. Parhi. *VLSI Digital Signal Processing Systems*. John Wiley and Sons, Inc., New York, NY, 1999.

[33] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *International Workshop on Compilers and Operating Systems for Low Power*, Barcelona, Spain, Sept. 2001.

[34] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *International Symposium on Performance Analysis of Systems and Software*, pages 111–122, Austin, TX, Mar. 2003.

[35] C. Saldanha and M. Lipasti. Power efficient cache coherence. In *Workshop on Memory Performance Issues*, Göteborg, Sweden, June 2001.

[36] R. Sasanka, S. V. Adve, Y. Chen, and E. Debes. Comparing the energy efficiency of CMP and SMT architectures for multimedia workloads. In *International Conference on Supercomputing*, pages 196–206, Malo, France, June–July 2004.

[37] J. S. Seng, D. M. Tullsen, and G. Z. N. Cai. Power-sensitive multithreaded architecture. In *International Conference on Computer Design*, pages 199–208, Austin, Texas, Sept. 2000.

[38] K. Skadron, M. Stan, W. Huang, and S. Velusamy. Temperature-aware microarchitecture: Extended discussion and results. Technical Report CS-2003-08, University of Virginia, Apr. 2003.

[39] U. Weiser. Microprocessors: Bypass the power wall. In *Intel Academic Forum*, Barcelona, Spain, Apr. 2004. (Keynote presentation).

[40] S. Wilton and N. Jouppi. CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, May 1996.

[41] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *International Symposium on Computer Architecture*, pages 24–36, Santa Margherita Ligure, Italy, June 1995.