RC24086 (W0610-124) October 23, 2006
Computer Science

IBM Research Report

Evaluating Design Tradeoffs in On-Chip Power Management for CMPs

Joseph Sharkey
SUNY Binghamton
Binghamton, NY

Alper Buyuktosunoglu, Pradip Bose
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Evaluating Design Tradeoffs in On-Chip Power Management for CMPs

Abstract

The continual scaling of process technology has resulted in a larger number of transistors available on a single chip and has thus enabled the dawn of Chip Multiprocessors (CMPs), where multiple processor cores are placed on a single chip. Additionally, power consumption remains a concern due to the thermal and reliability issues that result from high power dissipations. Dynamic power management techniques employed in current processors are designed for individual processor cores. With the shift towards multi-core chips, it is necessary to augment such per-core techniques with a larger chip-level control.

In this paper, we first model such a centralized power management unit in a full-system simulation environment. We implement asynchronous interfaces where needed to allow the individual cores of the CMP to operate at different voltages/frequencies and we also introduce the ability to adjust the fetch bandwidth of each processor core. Our model accounts for the cost of transition between power modes, as well as the control sampling time granularity constraints. To show the utility of our model, we then explore the design-tradeoffs associated with CMP power management solutions. We show that global power management solutions outperform the solutions that locally manage power per-core without any global chip-wide control. We then show that global power management is most effective at relatively fine granularities that allow it to adapt to changing workload behavior quickly and thus conclude that on-chip hardware solutions for CMP power management are important for future CMP microprocessors.

1. Introduction

The power consumption of modern microprocessors is of increasing concern due to the thermal and reliability issues that arise when the power consumptions become excessively large. In order to mitigate these power concerns, contemporary microprocessors typically employ some form of dynamic power management within the processor core. These power management techniques have largely been designed and optimized for single-core microprocessors.

Recently, however, there has been a shift towards the designs that place multiple processor cores on a single chip [21, 23]. Many modern industrial processors make use of such Chip Multiprocessor (CMP) designs [10, 11, 13, 18]. In light of this trend towards more aggressively scaled-out chips, it is necessary to explore the use of an additional layer of power management at the chip-level. This additional layer of control has a global view of the power and performance demands of the individual

processor cores as well as those of the entire chip. Using this information, the chip-wide power budget can be distributed among the processor cores so as to maximize the overall performance while meeting the chip-wide power budget.

To this end, we examine the need for a global on-chip power manager that redistributes power among multiple processors cores and explore the design-tradeoffs associated with such a manager. To explore the design-tradeoffs we choose to use simple power-control knobs, such as fetch throttling [17], as well as complex power-control knobs such as Dynamic Voltage/Frequency Scaling (DVFS) [4]. To evaluate the complex control knobs that result in on-chip asynchrony, we develop a methodology that allows such asynchrony to be modeled in the IBM Mambo [2] full-system simulator. The ability to model such complex systems is important in order to evaluate design-tradeoffs and to determine whether the development of such complex systems is indeed worthwhile. Furthermore, knowledge gained through such early stage evaluation is crucial in driving the direction of future microprocessor systems. We show the usefulness of our framework by evaluating the CMP power management solutions.

Specifically, we make the following contributions:

- We develop a modeling infrastructure that permits us to examine the use of various power management solutions in a full-system simulation environment.
- We compare global chip-wide power management solutions to those that manage power locally within each core.
- We compare the power management techniques that treat all cores homogeneously, forcing the same power controls to be applied to all cores at the same time, to those that treat the cores heterogeneously and allow the cores to operate at different power modes.
- We examine the impact of varying the control sampling time between the cores and the global power manager.

For both simple and complex control knobs, we show that global power management solutions outperform the solutions that locally manage power per-core without any global chip-wide control. We then show that global power management is most effective at relatively fine granularities that allow it to adapt to changing workload behavior quickly and thus conclude that on-chip hardware solutions for CMP power management are important for future CMP microprocessors.

The rest of the paper is organized as follows. Section 2 provides an overview of possible power management solutions for CMPs. In section 3, we describe the details of our simulation infrastructure. Section 4 describes the details of the power management variations examined in the paper. Section 5 presents the results, section 6 discusses the related work, and we offer our concluding remarks in section 7.

2. Overview

With the steady increase in the number of cores and active threads, it is clear from existing or prior studies that built-in mechanisms (like clock-gating [25]) alone will not suffice to mitigate the power concerns. This is especially true, when the objective is to reduce not just the average power, but also to ensure that the maximum power consumption is within affordable limits, without significantly compromising the overall performance. The modern power management problem is really a multi-variate optimization exercise, since the parameters involved are (at least) the power, temperature and performance levels at the local (core and non-core components) and global (chip) levels. Power-aware, *static* (or built-in) design techniques for each core and non-core component are of course necessary to increase the baseline energy-efficiency. Fine-grain clock-gating, power-efficient clocking and latch designs are examples of such static techniques; although, even with such built-in enhancements, the power savings can be a function of the input workload. In addition, however, on-chip *dynamic* power management has now become a necessity. That is, the ability to allocate, deallocate and reconfigure core and non-core resources in tune with the changing workload demand, is the modern server chip requirement. And, the goal is both average and maximum power management, as well as adherence to

peak temperature limits. Since the trade-off analysis requirement is multi-variate, and spans several cores and other resources on the chip, the on-chip power-performance manager needs to be architected as a hierarchical controller, with both local (per-resource) and global (chip-level) monitoring-and-response functions. The control function can be entirely "reactive and open-loop", with no feedback-directed attempt to correct, extend or curtail the response. It can also be implemented as a partly or fully "closed-loop" function, with feedback-directed control to correct overshoots and undershoots. In general, a hybrid approach with both open- and closed-loop control elements may be used to provide robust management functionality at affordable cost and complexity.

In an effort to optimize the resource usage efficiency through such dynamic adaptation, the aim of course is to minimize any performance impact. Only in rare cases of very significant overload, where all or most of the cores are subject to worst-case workload demand, should the global power manager function invoke a corrective response that significantly degrades the performance of one or more threads.

Prior research in the field of power-aware microarchitecture design has yielded a "bag of tricks" to reduce power while maintaining the original performance target within a specified range. Ideally, one would like to invoke a subset of these tricks, depending on the workload (or a particular phase of the workload) and the machine state. This is exactly where the global power manager function comes in. This "monitor-and-control" facility is architected in such a way that it would sense the workload demand and machine state and dynamically invoke particular mechanisms from within the full repertoire of architected techniques for power reduction and control.

As workload phase behavior can occur at granularities varying from very coarse (e.g. at the scale of hundreds or thousands of milliseconds) to very fine grain (nano and microseconds), conceptually, one would formulate the dynamic power management problem as a hierarchical feedback control algorithm involving hardware (fast) and software (slow) elements. This work focuses on the design of

the micro- and milli-second level fast control mechanisms affected by a global on-chip hardware controller.

2.1 Power Manager Design Alternatives

There are many possibilities for the design of a power management solution for CMPs. These solutions can be classified as either *local* management techniques or *global* management techniques.

Local solutions naively off-load the power management responsibilities to the individual cores. In this case, each core is responsible to locally manage its own individual power consumption without any global communication. This type of control requires the fewest changes to current designs and can be employed by simply using the existing techniques for single-core microprocessors. The drawback to such a design, however, is that opportunities for global optimization can be missed.

On the other hand, global solutions employ a power manager that can communicate with all of the individual cores. This globally aware power manager can then make power adjustment decisions in a way that is optimal for the entire chip. Such "global" power managers benefit from their comprehensive view of power and performance of the entire chip, as well as the individual contributions of each core. These types of power managers may be implemented in either hardware or software, as long as a communication link exists connecting the manager and all the cores.

Global power managers can be further classified as those that treat the cores *homogeneously* and those that treat the cores *heterogeneously*. With homogeneous solutions, all cores receive equal portions of the total power budget and every power adjustment affects all cores. Heterogeneous solutions go further and provide a dynamic redistribution of power among the cores. While heterogeneous solutions can better adapt to workloads with large inter-thread variations, homogeneous solutions may be less costly to implement and verify. This is because of the requirement of on-chip asynchrony and separately controllable voltage/frequency islands implied by the heterogeneous solutions. Therefore, in order for the heterogeneous solutions to be attractive, we surmise that the performance delta between heterogeneous and homogeneous techniques has to be quite large.

Little work has been done to explore the design trade-offs associated with each of these classes of power managers. This work makes several contributions to this end. First, we show that global power managers outperform the local managers in general because of the opportunity to take advantage of global optimizations afforded by such designs. Next, we show that there is little performance difference between the global-homogeneous and global-heterogeneous designs. This, coupled with the additional complexities associated with the global-heterogeneous designs, makes the global-homogeneous solutions the most attractive design point. We show that these conclusions hold true for both simple and complex power control knobs. Finally, we show that the global-homogeneous solutions are most effective a fine time granularities, and therefore necessitate an on-chip hardware implementation.

3. Methodology and Simulation Details

In this section, we describe the simulation methodology developed in order to evaluate the design-tradeoffs associated with various power-management solutions for CMPs. We begin by describing the steps used to model the on-chip asynchrony resulting from per-core DVFS. Then, we describe the power modeling methodology used in the paper.

3.1 Simulating On-Chip Asynchrony

In order to evaluate the design trade-offs associated with such power management solutions, we needed a simulation environment that permits the individual cores of the CMP to operate at different frequencies and allows those frequencies to be adjusted dynamically. Modeling such on-chip asynchrony is non trivial in today's simulation infrastructures. While the detailed implementation of an asynchronous processor is beyond the scope of this paper, our interests lie in the availability of a basic model with which to begin to explore the design space of CMP power management from a research perspective in order to evaluate whether a more detailed, full blown investigation is warranted.

To address this deficiency, we chose to build on top of the IBM Mambo full system simulator [2]. The simulator is execution-driven and has the ability to boot and run a full operating system. We

modeled an IBM POWER5-like processor with 4 cores and run Linux as the operating system within the simulations.

Table 1: Range of frequencies examined

Ratio	Percentage of Nominal Frequency
55/55	100.00%
54/55	98.18%
53/55	96.36%
52/55	94.55%
51/55	92.73%
50/55	90.91%
49/55	89.09%
48/55	87.27%
47/55	85.45%
46/55	83.64%
45/55	81.82%

We first defined the range of target operating frequencies that we would like to examine. For these studies, we choose to examine 10 frequency steps in the range of about 80%-100% of the nominal frequency, which yields 11 discrete operating points. This implies that each step should adjust the frequency by 20% / 11 = ~1.81%, or more precisely, $1/55^{th}$. Therefore, we consider the frequencies in the range of 55/55 * *Fn* to 45/55 * *Fn*, where *Fn* is the nominal frequency for the processor, as shown in Table 1.

After determining the frequency ratios to examine, we modified the simulator by increasing the frequency of the "simulator clock" and adjusting the number of "simulator clock ticks" that are mapped to a single "processor clock cycle". For example, in a traditional single processor event-driven simulator, events are scheduled based on the number of cycles between the current cycle and the cycle in which the event will occur. Instructions then schedule an event to occur in the cycle in which they will complete a given pipeline stage. When that event occurs, the instruction removes itself from the current pipeline stage and advances to the next stage, scheduling another event to occur in the cycle in which it will complete that stage. Typically, these delays are expressed in the number of processor clock cycles until the even occurs (i.e. 1 cycle, 2 cycles, 10 cycles, etc). This assumes a one-to-one mapping between "simulator ticks" and "processor cycles". However, this one-to-one mapping is not

necessary. It is possible to map one "processor cycle" to 2 "simulation ticks", in which case a request for an instruction to delay "one processor cycle" will be interpreted as a request to delay by "two simulation ticks". Notice that if this mapping applies to the scheduling of *every single processor event*, then the net simulation is exactly the same as with the one-to-one mapping, except that twice as many "simulation ticks" will have transpired in order to simulate the same number of "processor clock cycles". In general, one "processor cycle" can be mapped to *n* "simulation ticks" by mapping each event request to delay by *d* "processor cycles" to *d*n* "simulation ticks".

3.2 Power Estimation Methodology

For estimating the power of each processor core, we employ a utilization based power model using the following equation:

$$\begin{aligned}\mathbf{Power_{CORE}} = \quad & \mathbf{Activity_{IFU} * Weight_{IFU} + Activity_{IDU} * Weight_{IDU} + Activity_{ISU} * Weight_{ISU} +} \\ & \mathbf{Activity_{FXU} * Weight_{FXU} + Activity_{FPU} * Weight_{FPU} + Activity_{LSU} * Weight_{LSU}} \\ & \mathbf{+ Hold\ (switching)\ Power + Leakage\_Power}\end{aligned}$$

This model tracks the activity of the key processor components and each activity factor is multiplied by a "weight" that represents the corresponding component's contribution to the unconstrained chip power as extracted from real power measurements. Essentially, this abstract model expresses the power of a given unit j, as:

$$P_j = L_j + H_j + (S_j max - H_j) * A_j$$

where, L_j is the leakage power (independent of activity or utilization); H_j is the active (switching) hold power that is consumed when the unit is idle; A_j (1 μ A_j μ 0) is the activity factor, that captures the degree to which the unit is utilized over the period of measurement; $S_j max$ is the maximum (or unconstrained) active (switching) power that is consumed when the unit is fully utilized (i.e. for A_j = 1). The weight of the j^{th} unit then is $W_j = (S_j max - H_j)$.

We derive our unit-level "weights" from the detailed circuit-level power measurement data that were used to project the active power pie chart of [3]. The total chip active power is known, and the individual unit fractional powers are as reported in [3]. The active hold power for each unit is available by adding up the macro-level hold powers for that unit (available from circuit-level simulation). The leakage power for each unit (which, like the active hold power, is a constant adder in our linear power model) is also available from circuit-level, macro-wise gate-width analysis. However, for the purposes of our study, a simpler abstraction that models leakage as a parameterized multiplier (fraction) of the active hold power (H_j) is used. Note that the unit-level component active powers, built up from macro-level characterization at the circuit level [3] assumed a constant (average) data switching factor for a given unit. These data switching factors were derived by running representative test cases on the RTL (VHDL) model of the processor, and averaging the measured data switching factors over all macros constituting that unit. A pictorial illustration of the unit-level power model formulation is shown in Figure 1.

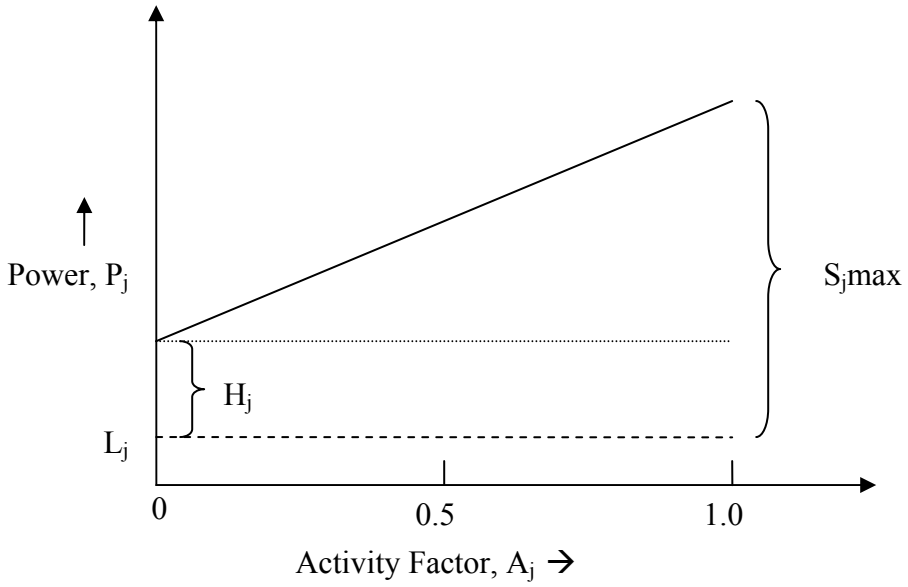

Figure 1. Power model for j^{th} modeled unit within a processor core. L_j is the constant leakage power; H_j is the active hold power; $S_j$max is the maximum value of the active (switching power component) when the activity factor, $A_j = 1$. A constant (average) data switching factor is implicitly assumed for the unit, based on measured values for representative test cases run on the RTL (VHDL) model.

In our experience, this type of abstract power model, constructed (in a bottom-up manner) from detailed macro-level power measurements of a POWER4/5-class processor [3], has been found to be quite accurate for multi-core power-performance tradeoff analysis research. The emphasis in such research is on understanding how multi-core (active) power budgets can be managed and redistributed across the cores, as the input workload changes, without giving up much throughput performance. Detailed characterization of intra-core power and thermal redistribution is not of direct interest in this research. Furthermore, in a physical implementation of an on-chip power management unit, the measurement of real, calibrated power consumption in hardware is a difficult task and it is therefore likely that such a utilization based power model will be used as a practical approximation. The goal of the power manager would be to always keep the total power within some safe limit such that calibrated on-chip thermal limit sensors do not keep triggering to invoke brute-force shutdown or slowdown mechanisms. To drive such a power management algorithm, a very precise measurement of per-core power is not necessary. Note that this power model is used to estimate the power of only the processor cores, and not the non-core elements. Non-core power can again be approximated as a constant adder, as needed.

We simulated several benchmarks from the SPLASH-2 application suite [24]. These are multi-threaded applications and are configured to run with four threads. For all simulations, the applications are run to completion and the results are reported for the entire run.

In choosing the power budgets, we first ran all benchmarks with no power manager and measure the power consumptions. Then, we choose power budgets as a percentage of the peak power consumption for each benchmark. For DVFS, we choose these power budgets to be 65%, 75%, 85%, and 95% of the peak power consumption. Since fetch throttling does not adjust the voltage, however, it is limited in the range of power reductions possible. Therefore, for fetch throttling, we choose power budgets of 85%, 89%, 93% and 96% of the peak power consumption.

4. Power Management Variations Examined

This section describes the power management variations examined in this work. We consider both a simple power control knob (fetch throttling) and a complex control knob (Dynamic Voltage/Frequency Scaling).

4.1 Power Management via Fetch Throttling

Fetch throttling is the ability to control the number of instructions a processor fetches each cycle. Typically, a processor core is permitted to fetch instructions every cycle. When fetch throttling is used, the core is only allowed to fetch instructions one out of every *t* cycles, where *t* is the level of fetch throttling. For example, when *t* is set to 2, the core can only fetch instructions every other cycle. By reducing the speed at which instructions are fetched, the activity throughout the core can be reduced. Assuming that there is prevalent clock gating throughout the core, the power consumption will decrease with the decrease in activity.

We examine three variations for power budget distribution via fetch throttling. First, we examine a local fetch throttling approach where each core individually controls its own throttling level. Here, the total chip power budget is divided evenly among the cores and each core then manages its own fetch throttling level in order to meet its individual power budget, without any global knowledge about the power consumptions of the other cores. With this scheme, all cores need not be throttled evenly. For example, one core may naturally meet its power budget without any fetch throttling while at the same time another core may need to invoke a very high level of throttling to meet its budget.

Next, we examine a global-homogeneous fetch throttling approach that monitors the total chip power consumption and throttles all cores evenly to meet the chip-wide power budget. This variation does not have any individual information about the power and performance of each core and simply relies on total chip power measurements, as well as the set chip-wide power budget, in making throttling decisions. With this scheme, all of the cores on the chip are throttled evenly at any given time. Because of this, one power-hungry core can push the entire chip over budget and therefore force

throttling to occur for all of the cores on the chip. This may be undesirable, especially if these cores are running threads from different applications.

The last variation is a global-heterogeneous fetch throttling approach that monitors the power consumption on the global level and has knowledge of the total chip power consumption as well as the contributions for each individual core. The throttling level for each core is then set so as to maximize the performance of the total chip while meeting the given power budget. The power redistribution algorithm works by evaluating all of the possible combinations of throttling levels for each core, and then chooses the one that maximizes the performance while meeting the power budget. To accomplish this, it assumes a linear scaling of power and performance with the throttling level. Furthermore, the algorithm is limited to not exceed the throttling level 8 for any core (beyond which point excessive performance degradations are incurred).

4.2 Power Management via Dynamic Voltage/Frequency Scaling

While fetch throttling has a linear relationship between power reduction and performance degradation, DVFS has a cubic relationship. Specifically, with DVFS it is possible to achieve a 1x reduction in performance for a 3x reduction in power [4]. This is because the (active) power is proportional to V^2F, while the performance is (roughly) proportional to F, which is (roughly) proportional to V. Thus, DVFS is an attractive approach for providing large power reductions at relatively small performance impact. However, the additional design and verification complexity required to support DVFS (especially on a per-core basis) is known to be quite significant. Despite these known roadblocks, we examine its potential from a research perspective in order to gauge whether or not it is an avenue worth pursuing. Previous work has examined the use of a CMP power manager using DVFS [8] using three *power modes* (i.e. voltage/frequency pairs). We extend this to incorporate 10 power modes and examine two variations of power management that use this 10-mode DVFS.

The first is a heterogeneous-global solution that implements the maxBIPS algorithm of [8]. This algorithm tries to optimize the overall system throughput by examining all voltage/frequency pairs for each core and selecting the one that meets the power budget while achieving the highest level of performance. The cores are permitted to operate at different voltages and frequencies. For a detailed analysis of the algorithm, we refer the reader to [8].

We also examine a global-homogeneous power manager using DVFS, which we refer to as chip-wide DVFS. Here, the voltage and frequency can only be adjusted at the chip level, so all cores operate at the same voltage/frequency at any given time. While the performance of this variation is limited by the chip-wide power modes, it may be more feasible to implement than per-core DVFS.

5. Results

This section examines the results of the various power management solutions.

5.1 Fetch Throttling Control Knob

We begin by examining the power management solutions using fetch throttling as a power control knob. Figure 2 presents the average $BIPS^3$/Watt across the simulated benchmarks for various power budgets, presented relative to the baseline machine $BIPS^3$/Watt. We choose to present the results in terms of this metric because it takes both power and performance into account. Considering power savings or performance degradation alone is not sufficient because there is a subtle tradeoff between power and performance. Therefore, considering the combined effect on both metrics is important. Since the goal of the examined power managers is to maximize performance given a power budget, we choose the $BIPS^3$/Watt metric[1] because it weighs the performance impact more than the power savings.

The first two bars show the results for the global homogeneous technique at 1μs and 1ms granularities, respectively. The next two bars show the results for the global heterogeneous solution

[1] BIPS stands for the standard performance metric of billions of instructions per second. The $(performance)^3$/watt is an [energy * $(delay)^2$] metric that is commonly used to measure power-performance efficiency, especially for evaluating the efficiency of active power management techniques in server-class processors: e.g. see D. Brooks et al., "Power-aware microarchitectures: design and modeling challenges for next generation microprocessors", *IEEE Micro*, vol. 20, no. 6, pp. 26-44, Nov. 2000; and V. Zyuban et al., "Integrated analysis of power and performance for pipelined microprocessors," *IEEE Trans. on Computers*, vol. 53, no. 8, Aug. 2004.

at 1μs and 1ms granularities, and finally the last two bars present the results for the local solution at the 1μs and 1ms granularities.

As seen from the graph, all of the global solutions outperform the local solutions quite significantly at both time granularities. Specifically, for the techniques invoked every 1μs with a power budget of 96%, the reduction in the average $BIPS^3$/Watt is 3.4% for the global homogeneous technique, 3.2% for the global heterogeneous technique, and 29.1% for the local technique. The local technique is inefficient because it lacks a chip-wide view of power and performance and therefore misses many opportunities for global optimization. Specifically, although one particular core (say $Core_N$) may consume more than its share of the power budget at a given time, the total chip power at this time may still be within the total chip power budget if the other cores consume less than their share of the power budget. In other words, the power budget can be met by having one core with high power consumption and three cores with relatively low power consumption. However, this situation can not be detected by the local technique and thus the local power manager for $Core_N$ will invoke fetch throttling unnecessarily, therefore directly reducing the performance/power efficiency as reflected in the $BIPS^3$/Watt results presented in the figure.

The next trend to observe is that both the global homogeneous and global heterogeneous techniques achieve similar results in terms of the $BIPS^3$/Watt metric for all the examined power budgets. Even examining the per-benchmark results (not presented due to space constraints), we see that although global homogeneous is better in some cases and global heterogeneous is better in others, there is generally very little difference between the two. Furthermore, we find only small differences between the techniques invoked every 1μs and those invoked every 1ms.

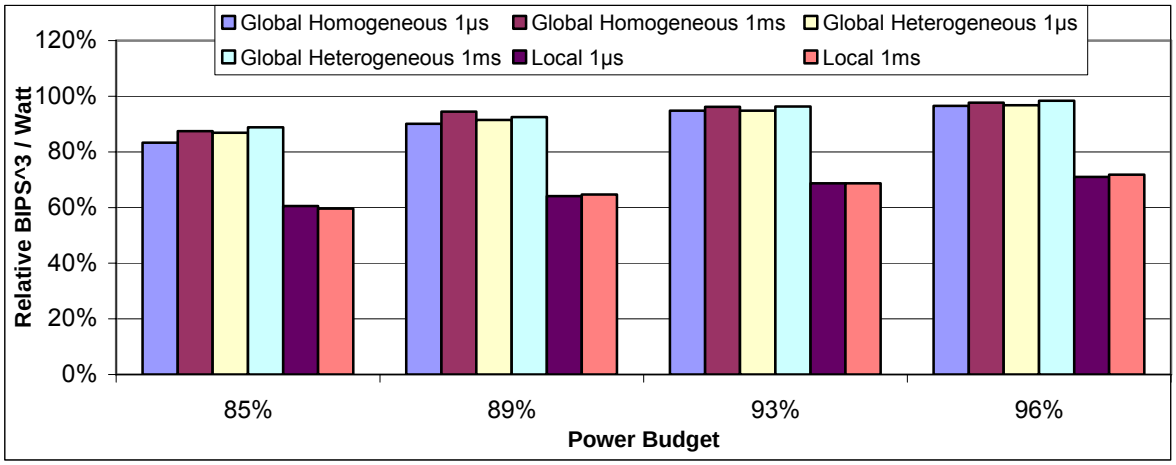

Figure 2: BIPS^3/Watt relative to the baseline machine for various fetch-throttling power management solutions and various power budgets. These results are presented for the average across the simulated benchmarks.

To further distinguish between the global homogeneous and global heterogeneous approaches, we now analyze the average budget overshoots. We define the overshoot as the percentage of time that the measured power is greater than the given power budget multiplied by the average amount (in Watts) that the measured power exceeds the power budget. Therefore, lower values of the overshoot are better whereas high values of the overshoot indicate an inefficiency at meeting the given power budget. As seen from Figure 3, the average overshoot is very small for all schemes at the 1µs granularity, but the overshoot becomes large when the granularity is increased to 1ms. This is because, at the 1ms granularity, the power manager can not react quickly enough to the changing workload behavior in order to prevent the large budget overshoots.

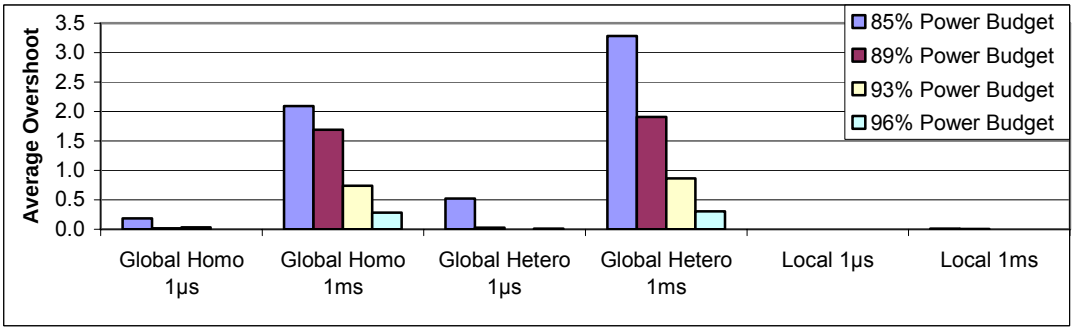

Figure 3: Average overshoot across all benchmarks for various fetch-throttling power management solutions. (average overshoot is defined as the percentage of time that the measured power is greater than the given power budget multiplied by the average amount (in Watts) of these overshoots.)

5.2 DVFS Control Knob

We now examine the results for the power management techniques using DVFS as the power control knob. Figure 4 presents the $BIPS^3$/Watt, relative to the baseline machine, for various power budgets. The first two bars show the global homogeneous solution at 500μs and 10ms time granularities, respectively, while the last two bars show the results for the global heterogeneous solution at 500μs and 10ms time granularities, respectively. As seen from the graph, there is very little difference between these techniques in terms of energy-efficiency. Notice that this result is different than that of [8], which found a large difference between these two alternatives. This is a result of the fact that we use 10 levels of DVFS, while the work of [8] uses only 3. When only 3 levels of DVFS are used, the global homogeneous technique is quite limited in its choice of power modes and therefore is forced to select a mode that results in large performance degradations.

An interesting trend to note on the graph is that as the power budget decreases, the energy efficiency increases. This is an artifact of DVFS in that lower power budgets force the voltage/frequency to be lowered, resulting in a more efficient power/performance tradeoff. Specifically, we know that the power consumption is roughly related to V^3. The BIPS, however, scales down better than linearly with the frequency because the effective memory latency is reduced as the frequency goes down. Assuming that the voltage and frequency scale linearly, we can then expect the $BIPS^3$ to scale better than V^3, while the power (Watts) scales with V^3**.** The net result is an increase in the $BIPS^3$/Watt with lower power budgets, albeit at the cost of a significant performance degradation.

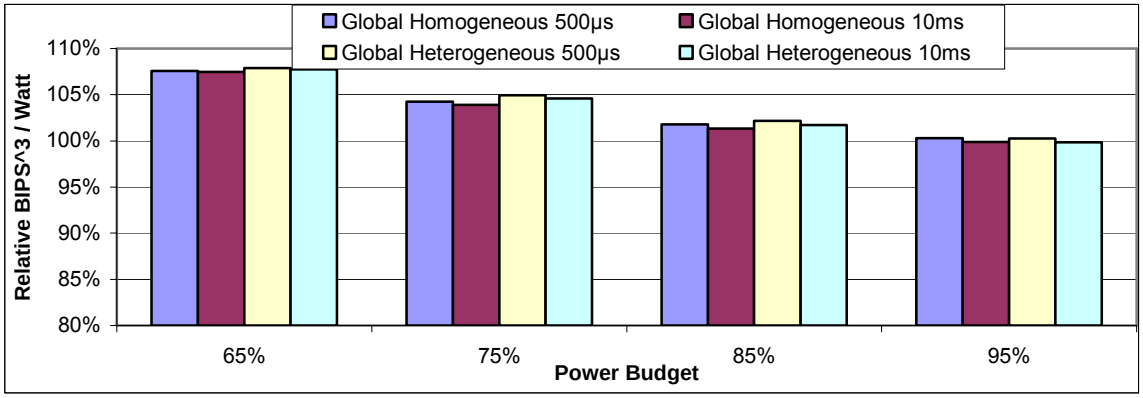

Figure 4: BIPS^3/Watt relative to the baseline machine for various DVFS power management solutions and various power budgets. These results are presented for the average across the simulated benchmarks.

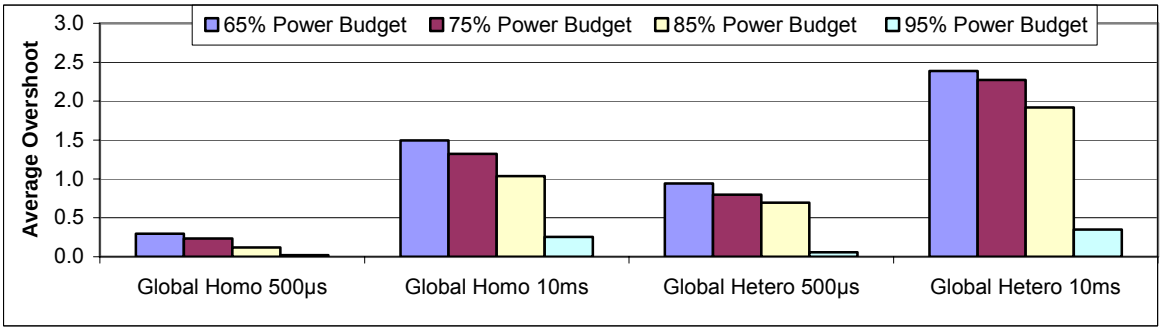

Figure 5: Average overshoot across all benchmarks for various power management DVFS solutions.

Now we examine the average overshoots for the power management systems harnessing the DVFS control knob, as presented in Figure 5. Lower values of the overshoot are better because an algorithm with less overshoot is more easily controlled by the setting of the power budget. If the overshoot is excessively large, then the given power budget is essentially meaningless.

As seen from the graph, the overshoot increases with the time granularity of control. For example, for the power budget of 85%, the overshoot of the global homogeneous solution increases from 0.3 to 1.5 as the time granularity goes from 500µs to 10ms. This is because when the time granularity is too coarse, all power management solutions have a difficult time enforcing the power budget because changes in the program phase that occur within that time period can not be adapted to. This is the main limitation of the techniques with a time granularity in the order of several milliseconds.

Also notice that for all power management solutions, as the power budget is decreased, the over shoot increases. This is because the smaller power budgets are more difficult to meet, and therefore often result in a larger overshoot.

Finally, we find that the global heterogeneous solution suffers a larger overshoot than the global homogeneous solution. The reason behind this is that, while the homogeneous solution has only one knob to control (i.e. global DVFS), the heterogeneous solution has 4 knobs (i.e. DVFS for each core). Therefore, it is more difficult for the heterogeneous solution to estimate what the exact power/performance implications will be of changing all 4 knobs at once, resulting in many cases where the prediction is incorrect. On the other hand, the homogeneous solution only has one knob to adjust and can therefore better estimate the impact of its power management decisions.

6. Related Work

The work most closely related to ours is that of [8], which examined the design of a power manager for CMPs using DVFS. Our work extends that of [8] in several key ways. First, we evaluate power management solutions in a full-system simulation environment, as opposed to the trace-driven environment of [8]. We provide a taxonomy for classifying CMP power management solutions as local or global and homogeneous or heterogeneous. We also examine the use of both simple and complex control knobs and evaluate both at different control sampling time granularities. Furthermore, we evaluate the design alternatives using multi-threaded (as opposed to multi-programmed) workloads.

Juang et al. [9] also look at CMPs with dynamically configured voltage and frequencies. However, their work adjusts individual core executions to improve power-performance trade-offs by balancing inter-thread producer-consumer rates. Kotla et al. [12] consider a CMP system with multiple effective frequency domains achieved by throttling each core at a different rate. They demonstrate that memory intensive applications can be run on a slower core without significant performance loss. In comparison to these, our work develops policies that can adapt each core's behavior under fixed power budgets.

Li and Martinez [14, 15] investigate methods to identify the optimal operating point on a CMP in terms of number of active cores and DVFS settings for parallel applications. They consider dynamic configurations under limited performance and power constraints and develop analytical models for attainable speedups. They consider application of chip-wide DVFS to manage parallel regions of applications and perform few explorations guided by heuristics to reach an optimal operating point. In contrast, our work focuses on comparing the per-core to chip-wide power management actions guided by policies that predict different power mode behaviors.

Another line of prior work also considers dynamic management under power budget constraints. Merkel et al. [19, 20] present system level methods that adjust processor execution to fit target budgets. They develop an energy-aware scheduling policy to meet uniform budgets for each core to eliminate thermal emergencies. Grochowski et al. [7] discuss latency and throughput trade-offs under chip power constraints. They survey different adaptation techniques and suggest asymmetric cores with DVFS as the most promising alternative. In a related study, Annavaram et al. [1] consider a real implementation example with both static and dynamic asymmetric multiprocessors (AMP) and improve performance of multithreaded applications under fixed power budgets. These works discuss adaptations in response to the available thread parallelism in applications, while our work investigates core power adaptation policies for fully-utilized CMPs based on predictive power and performance models at different power modes.

There is a wide range of different prior studies that focus on the design and control of CMP systems under power, thermal or area constraints [5, 6, 16, 22]. Most of these management techniques guide independent local actions or OS level scheduling decisions. In contrast, our work focuses on policies that perform power/performance management by adjusting the behavior of individual cores under global chip-wide power budgets with the supervision of global monitoring and control.

7. Concluding Remarks and Future Work

The power consumption of modern microprocessors is of utmost concern due to the thermal and reliability concerns that arise with high power dissipation. As the trend towards multi-core CMPs continues, it is important to consider the power management alternatives for such a system so as to reduce the impact on the overall system performance. However, modeling and analyzing the effects of techniques such as Dynamic Voltage/Frequency Scaling (DVFS) that result in on-chip asynchrony are not trivial in today's simulation environments.

To overcome this limitation, we developed a full-system simulation environment in which to model such asynchrony at the processor core level. We show the usefulness of such a tool to examine the design trade-offs associated with various CMP power management alternatives. We showed that chip-wide solutions are necessary in order to take advantage of the global optimization opportunities. Furthermore, we show that the choice of time granularity for the power manager can significantly impact its ability to maintain a power budget. When the time granularity is too coarse (i.e. several milliseconds), it becomes increasingly difficult to enforce the power budget because quick changes in program phase can not be detected. Therefore, we find small time granularities (in the order of several milliseconds) to be most effective for CMP power management and conclude that on-chip, hardware, global power manager designs may be necessary to meet these timing constraints.

Based on these findings, we feel that a combined solution that uses both a complex control knob and a simple control knob may provide the best trade-off. For example, implementing coarse-grained chip-wide DVFS with fine-grained per-core fetch throttling could provide the ability to both react quickly to changing workload behavior and at the same time achieve the attractive power/performance tradeoffs associated with DVFS. We leave the exploration and design of such a system to future work.

8. References

[1] M. Annavaram, E. Grochowski, and J. Shen. Mitigating Amdahl's Law Through EPI Throttling. In Proceedings of the 32nd International Symposium on Computer Architecture (ISCA-32), 2005.

[2] P. Bohrer, J. Peterson, H. Shafi, "Mambo: Advances in PowerPC System Simulation", Invited Tutorial, 2003 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), March 9, 2003, Austin, Texas.

[3] P. Bose, D. Brooks, A. Buyuktosunoglu, P.Cook, K. Das, P. Emma, M. Gschwind, H. Jacobson, T. Karkhanis, S. Schuster, J. Smith, V. Srinivasan, V. Zyuban, D. Albonesi, S. Dwarkadas. Early-Stage Definition of LPX: A Low Power Issue-Execute Processor Prototype. Power-Aware Computer Systems (PACS) workshop in conjunction with 8th International Symposium on High Performance Computer Architecture (HPCA-8), 2002.

[4] K. Choi, R. Soma, and M. Pedram. Dynamic Voltage and Frequency Scaling based on Workload Decomposition. In Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), Aug. 2004.

[5] J. D. Davis, J. Laudon, and K. Olukotun. Maximizing CMP Throughput with Mediocre Cores. In 14th International Conference on Parallel Architecture and Compilation Techniques (PACT'05), 2005.

[6] J. Donald and M. Martonosi. Techniques for Multicore Thermal Management: Classification and New Exploration. In Proceedings of the 33th International Symposium on Computer Architecture (ISCA-33), 2006.

[7] E. Grochowski, R. Ronen, J. Shen, and H. Wang. Best of Both Latency and Throughput. In Proceedings of the International Conference on Computer Desigd (ICCD), 2004.

[8] C. Isci, A. Buyuktosunoglu, C. Y. Cher, P. Bose, and M. Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In Proceedings of the International Symposium on Microrchitecture (MICRO) 2006.

[9] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. Clark. Coordinated, Distributed, Formal Energy Management of Chip Multiprocessors. In Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'05), Aug. 2005.

[10] R. Kalla, B. Sinharoy, and J. Tendler. IBM POWER5 Chip: A Dual-Core Multithreaded Processor. IEEE Micro, 24(2):40–47, Mar/Apr 2004.

[11] P. Kongetira. A 32-way Multithreaded SPARC(R) Processor. Hot Chips 16, Aug 2004.

[12] R. Kotla, A. Devgan, S. Ghiasi, T. Keller, and F. Rawson. Characterizing the Impact of Different Memory-Intensity Levels. In IEEE 7th Annual Workshop on Workload Characterization (WWC-7), Oct. 2004.

[13] K. Krewell. UltraSPARC IV Mirrors Predecessor: Sun Builds Dual-Core Chip in 130nm. Microprocessor Report, Nov 2003.

[14] J. Li and J. Martinez. Power-Performance Implications of Thread-Level Parallelism on Chip Multiprocessors. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'05), 2005.

[15] J. Li and J. Martinez. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA-12), 2006.

[16] Y. Li, D. Brooks, Z. Hu, and K. Skadron. Performance, Energy and Temperature Considerations for SMT and CMP Architectures. In 11th International Symposium on High Performance Computer Architecture (HPCA-11), 2005.

[17] S. Manne, A. Klauser, and D. Grunwald. Pipeline S. Manne, A. Klauser, and D. Grunwald. Pipeline Gating: Speculation Control for Energy Reduction. In Proceedings of the 25th International Symposium on Computer Architecture, pages 132–141, June/July 1998.

[18] C. McNairy and R. Bhatia. Montecito - The Next Product in the Itanium(R) Processor Family. Hot Chips 16, Aug 2004.

[19] A. Merkel. Balancing Power Consumption in Multiprocessor Systems. PhD thesis, Sept. 2005. System Architecture Group, University of Karlsruhe, Diploma Thesis.

[20] A. Merkel, F. Bellosa, and A.Weissel. Event-Driven Thermal Management in SMP Systems. In Second Workshop on Temperature-Aware Computer Systems (TACS'05), June 2005.

[21] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K.-Y. Chang. The Case for a Single-Chip Multiprocessor. In Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII), Oct. 1996.

[22] M. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. In Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) XI, 2004.

[23] L. Spracklen and S. G. Abraham. Chip Multithreading: Opportunities and Challenges. In 11th International Symposium on High Performance Computer Architecture (HPCA-11), 2005.

[24] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In Proceedings of the 22nd International Symposium on Computer Architecture, pages 24-36, Santa Margherita Ligure, Italy, June 1995.

[25] M. T. Zhang. Powering Intel(r) Pentium(r) 4 Generation Processors. In IEEE Electrical Performance of Electronic Packaging Conference, pages 215–218, 2001.