

CPM in CMPs: Coordinated Power Management in Chip-Multiprocessors

Asit K. Mishra, Shekhar Srikantaiah, Mahmut Kandemir, and Chita R. Das

Dept. of Computer Science and Engg., The Pennsylvania State University, University Park, PA - 16802, USA
{amishra, srikanta, kandemir, das}@cse.psu.edu

Abstract—Multiple clock domain architectures have recently been proposed to alleviate the power problem in CMPs by having different frequency/voltage values assigned to each domain based on workload requirements. However, accurate allocation of power to these voltage/frequency islands based on time varying workload characteristics as well as controlling the power consumption at the provisioned power level is quite non-trivial. Toward this end, we propose a two-tier feedback-based control theoretic solution. Our first-tier consists of a global power manager that allocates power targets to individual islands based on the workload dynamics. The power consumptions of these islands are in turn controlled by a second-tier, consisting of local controllers that regulate island power using dynamic voltage and frequency scaling in response to workload requirements.

I. Introduction

Driven by the need to alleviate the problem of power consumption in complex uniprocessor designs, coupled with the emphasis on maintaining the growing performance trend, we have seen an industry-wide adoption of chip multiprocessors (CMPs). However, in spite of the short term relief in power consumption provided by CMPs, the ever increasing emphasis on technology scaling for increased circuit densities/performance has only ensured the growing significance of the need to control chip power consumption.

Wire delays of global interconnects play an increasingly significant role in power-aware processor designs. However, shrinking technology nodes combined with the emphasis on higher clock frequencies has made it increasingly difficult to route a single global clock across a chip, while staying within the budget of the chip power. Globally Asynchronous Locally Synchronous (GALS) designs [18], [41], [30], [40] can significantly reduce power consumption by reducing the number of global interconnects. The design principle behind GALS architectures is that, wire delays can be controlled for short distances by means of design partitioning and integration of more metal layers which can limit the growth of wire dimensions to be slower than that of transistor dimensions. These recently proposed multiple clock domain architectures can benefit from having frequency/voltage values assigned to each domain based on workload requirements [30], [22]. Specifically, we envision a CMP design as depicted in Figure 1, where the cores on a chip are divided into several voltage/frequency islands, with each island having a separate clock. In general, each island may consist of more than one processor core.

Dynamic voltage frequency scaling (DVFS) for multiple clock domain micro-architectures has been studied by prior works [39], [34], [33] for different micro-architectural units of uniprocessors. However, multiple clock domain architectures have been relatively less explored in the context of chip multiprocessors. A recent study [14] has shown that DVFS can be highly effective in improving the energy-efficiency of CMPs running multithreaded commercial and scientific workloads, but the increasing flexibility offered by moving to fine-grained voltage/frequency islands does not necessarily translate into better energy-efficiency. DVFS techniques for voltage/frequency islands at coarser granularity, like multiple cores of a CMP forming different voltage/frequency islands, have been relatively less explored [6].

One of the important aspects to be considered when performing DVFS on such voltage/frequency islands of multiple cores is the impact of voltage/frequency scaling on co-scheduled application

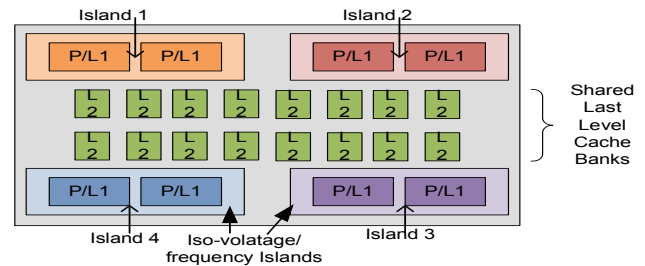


Fig. 1. CMP layout showing voltage/frequency islands with each island having 2 cores with private L1 cache.

threads (application threads scheduled on the same voltage/frequency island). Another important issue is accurate *provisioning of power* to these voltage/frequency islands based on time varying workload characteristics. Power provisioning to different islands should be aware of various aspects of system requirements like performance, reliability, process-variations and thermal impacts. Further, *capping* or *controlling* the power consumption to the provisioned power of an island is also important. Most prior research on provisioning power and controlling processor power consumption has focused on large-scale clusters or data centers [37], [31], [7], [32] by studying the efficacy of providing dynamic management of power and thermal hot spots through localized responses provided at the core-level or through global scheduling heuristics [17]. However, the monitoring logic used to enforce the power budgets are based on open loop control or ad-hoc heuristics and do not provide the same robustness compared to formal feedback control based solutions as proposed in our work. Furthermore, the constraints on the achievable performance were less stringent in the previous works as they perform per-core DVFS as opposed to per-island DVFS in our work where co-scheduled threads are influenced by the voltage scaling of an island.

Toward this end, we propose a feedback control solution for accurate power management in CMPs. Our approach proposes a two-tier solution consisting of using a *Global Power Manager* (GPM) to provision power to individual voltage/frequency islands at the first-level and *Local Per-Island Controllers* (LPIC) at the second-level, which regulate island power consumption using dynamic voltage/frequency scaling in response to modulations in workload requirements. Specifically, the merits of this paper are the following:

- We propose a decoupled and coordinated power control architecture for chip multiprocessors based on voltage/frequency scaling of islands consisting of *multiple cores*.
- We show that we can obtain *accurate* results in power provisioning as well as power capping in individual islands by deriving a *system model* for power consumption in a CMP. We evaluate the behavior of a Proportional, Integral and Derivative (PID) based controller derived from this model in terms of the maximum overshoot of power, maximum settling time and maximum steady-state error.
- We also discuss the flexibility obtained by decoupling the global (chip-wide) power manager and the local (per-island) controllers and illustrate this flexibility using three example policies: *performance-aware* power provisioning, *thermal-aware* power provisioning and *variation-aware* power provisioning. In the performance-aware power

provisioning policy, the goal is maximizing the performance while limiting the chip-wide power to the specified budget. In the thermal-aware power provisioning policy, the system tries to eliminate hot-spots and thereby provision power so that local hot-spots do not arise. In the variation-aware policy, the system tries to minimize the ratio of power and throughput when the CMP substrate has intra-die variations in leakage current. Note, that these are three representative policies we discuss to show the flexibility of our tiered control proposal and other policies are also feasible.

- Through extensive simulations using a full-system simulator, we show that our power control architecture is able to provision power and control per-island power in an optimal manner given the constraints enforced by the co-scheduled applications. Our experimental evaluations show that the proposed power management technique is able to track the chip-wide budget with a very high accuracy and the maximum overshoot in power consumption of each island in 8, 16 and 32 core CMPs is bounded within 4% of the target, and the controller is able to achieve an almost zero steady-state error within a small period of five controller invocations.

We structure the rest of this paper as follows. First, we describe the architecture of the proposed decoupled power control technique, our power/performance models the proposed GPM and LPIC schemes in detail in Section II. Our experimental setup is then described in Section III, followed by the experimental results in Section IV. The related work is described in Section V, and we finally conclude in Section VI.

II. Power Control Architecture

A. Background

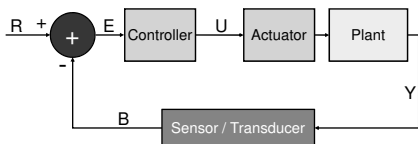


Fig. 2. A generic feedback control loop showing the various components in controlling a system (plant).

Before delving into the details of our proposed mechanisms, we briefly discuss a generic feedback controller and how it correlates to our problem statement. Figure 2 shows a generic controller schematic. To associate this generic feedback loop with our CMP power control problem, let us consider a particular scenario based on our target CMP depicted earlier in Figure 1. In this scenario, our reference input, R , is the maximum allowable power consumption of an island (“plant” in Figure 2) and we would like to modulate the frequency/voltage level of the cores of the island dynamically for achieving this. The output signal, Y , can be measured in terms of CPU utilization, which can be obtained using hardware performance counters available in many modern systems. Since R and Y are of different types of metrics, the job of the sensor/transducer component is to convert Y to a metric (B), which can be directly compared to R . The difference between R and B , denoted using E , is fed to the controller, which decides the new voltage and frequency levels. This voltage/frequency scaling is then implemented by an actuator exercising the necessary hardware knobs or by invoking certain OS services. Designing these components is quite non-trivial; for instance, the transducer in this scenario also involves modeling the power in terms of CPU utilization. Note that in this approach, the controller makes the “policy” and actuator enforces that policy using a “mechanism” such as voltage/frequency scaling.

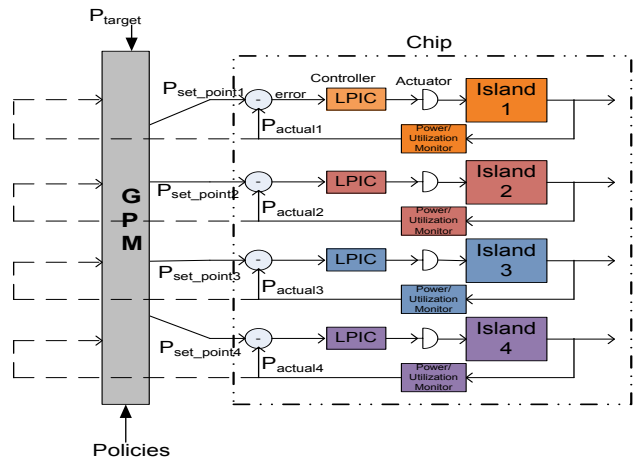


Fig. 3. Overview of our power control architecture.

Thus, a control-theoretic approach to system management in general requires:

- Identifying the optimization criteria (e.g., minimizing the tracking error between reference power and actual power), which is referred to as the “cost function”.
- Identifying the parameters that are responsible for control (e.g., the frequency or voltage of the core), called “control inputs”.
- Observing certain metrics to decipher the effect of the control (e.g., CPU utilization), called the “output variables”.
- A system model that characterizes how much effect would a change in the control parameters have on the output variables (e.g., a differential (or difference) equation that characterizes the dynamic relation between the reference frequency and CPU power), which is called the “dynamic system model”.

In designing and implementing such a feedback control system, it is critical to consider three important metrics that measure the robustness of the controller. The first metric, *maximum overshoot*, is the difference between the maximum observed output variable and the reference. The second metric is the *steady-state error*, which represents the difference between the observed output variable and the reference in the steady-state of the controller. The third metric is the *settling time*, which is defined as the number of control invocations necessary to reach the steady state. In a nutshell, we can use these principles and design methodologies to develop a control theoretic solution for coordinated power provisioning and capping to voltage/frequency islands in chip multiprocessors. We elaborate on one such control theoretic solution and evaluate it experimentally to prove the efficacy of our proposal.

B. Architecture Overview

Figure 3 depicts a high-level overview of our proposed two-tier power control architecture. The two major components of this architecture are a *Global Power Manager* (GPM) and *Local Per-Island Controllers* (LPICs). These two components are organized hierarchically with the GPM being responsible for provisioning power among the islands and the LPICs being responsible for capping the power consumption of each island to be restricted to the provisioned power using voltage/frequency scaling. The LPIC controllers rely on the processor utilization as feedback in order to control the power, while the GPM relies on feedback provided in terms of system-level performance of applications scheduled on the islands to provision power to the islands. Note that, a significant difference between the architecture in our work and a previous work by Isci et al. [17] is that, the local controllers in [17] are open loop controllers (without

feedback), whereas we rely on closed loop control techniques for highly accurate power-capping. Also, with the projected scaling of CMPs to hundreds of cores, it will be prohibitively expensive to provide a per-core DVFS controller on chip. With this in mind, we design our schemes to work at the granularity of islands where multiple CPUs share a common DVFS controller. This is in contrast to all previous works that assume a per-core DVFS controller. Note that this puts extra pressure on the power management policies since all cores in an island are now restricted to operate under identical voltage frequency settings.

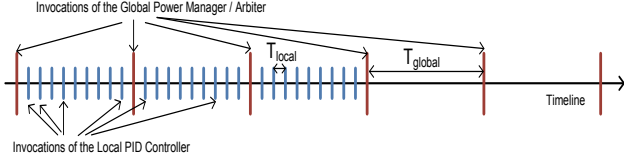


Fig. 4. Timeline of invocation of GPM and LPICs.

Figure 4 depicts a typical timeline of invocation of GPM and LPICs. In our two-tier scheme, the GPM is invoked at a coarser granularity of time to provision the power among the LPICs. Once provisioned, the LPICs are invoked more frequently to control the power at the provisioned level. The GPM is run by a supervisor code. The LPIC can be run by one of the cores in the island or can be implemented as part of a Power Management Unit (PMU), supported in architectures such as Itanium II [27] or can be implemented in firmwares. In the following sub-sections, we explain the GPM and LPIC components in further detail.

C. Global Power Manager (GPM)

The GPM is responsible for provisioning power to the different islands in the chip. It does so by having a global knowledge of the differences in workload dynamics across different islands. Such a mechanism provides *insulation* to the Local Per-Island Controllers and limits their functionality to only control of power (regulate power at a fixed value within a given interval of time), so that they can function independently. As long as the LPICs are effective in capping the power at the provisioned value, they are assured by the GPM that the global power consumption of the chip is maintained at the target power budget. Such hard guarantees can be provided by LPICs since in our approach they are designed based on theoretical foundations.

Another important flexibility in *decoupling* the GPM and LPICs is that we can adopt different high-level policies in provisioning power to different islands. The policies can consider various ramifications of power provisioning on different system metrics like performance, energy, thermal effects, variations, and reliability. As a proof of concept, we implemented three policies in our architecture. First, we describe a power provisioning policy that is *performance aware* and strives to obtain maximal instruction throughput for the given power budget. An alternate policy (discussed in Section IV) deals with the *thermal impact* of provisioning high power to neighboring cores. Additionally, we also evaluate a power provisioning policy (also discussed in Section IV) that is *process-variation aware* and seeks to minimize the ratio of power and throughput. Note that, many other policies like power provisioning for reducing energy consumption by providing a minimum guarantee on the performance or policies to increase reliability and QoS provisioning are also feasible using our approach, but are not evaluated here.

Performance-Aware Power Provisioning Policy: We now describe a GPM policy that provisions power, while optimizing performance of the system as a whole. In other words, our goal is to maximize overall

instruction throughput of the entire CMP, while ensuring that the overall power consumption is limited by the specified power budget P_{target} .

Consider, as an example, a chip multiprocessor system with N islands, $\{I_1, I_2, I_3 \dots I_N\}$. Suppose we denote the power budget of island I_i at time t as $P_i(t)$. In our approach, power is initially provisioned equally to each island, i.e., $P_i(0) = \frac{P_{target}}{N}$, where P_{target} is the total power budget for the system. The dynamic power consumption, $dP_i(t)$, of an island, I_i , at time t can be expressed as a function of frequency, $f(t)$ as:

$$dP_i(t) = k_0 \times f(t)^3, \quad (1)$$

where k_0 is a constant. Assuming that static power, $s_i(t)$, of the island I_i is a function of only time, we have the total power consumption of island I_i as:

$$P_i(t) = dP_i(t) + s_i(t), \quad (2)$$

Now, in order to improve the overall instruction throughput of the N islands measured in billions of instructions per second (BIPS), the performance aware provisioning policy scales the frequency of each island. In order to achieve optimal benefits, it has to do so in the proportion of expected performance variation for the scaling in frequency over the next interval of time. We know that:

$$Performance = \frac{1}{ExecutionTime} = \frac{f}{I_c \times CPI}, \quad (3)$$

where f is the frequency (or clock rate) of the system, I_c is the number of instructions in the program and CPI is the average clock cycles per instruction in the program. Assuming the performance scaling achieved during interval $t+1$ is similar to that achieved during interval t (since the time intervals are constant, we can ignore the time invariant impact of static power), we can estimate the performance at time $t+1$, from Equations (1), (2), and (3) as:

$$BIPS_i^e(t) = BIPS_i^a(t-1) \left(\frac{P_i(t-1)}{P_i(t-2)} \right)^{\frac{1}{3}}, \quad (4)$$

where the actual observed performance of island I_i at time t is denoted as $BIPS_i^a(t)$. We can now compute the ratio of actual performance to expected performance, $\phi_i(t)$ as:

$$\phi_i(t) = \frac{BIPS_i^a(t)}{BIPS_i^e(t)}. \quad (5)$$

For the best performance in terms of total BIPS in the next interval of time, the performance aware provisioning policy computes the provisions in proportions of the $\phi_i(t)$'s (of each island):

$$P_i(t+1) = P_{target} \times \frac{\phi_i(t)}{\sum_{i=1}^N \phi_i(t)}. \quad (6)$$

Note that, from Equation (6), $\sum_{i=1}^N P_i$ is always equal to P_{target} , i.e., the sum of the provisioned power to the islands is equal to the budgeted power at every instant of time when we adopt the performance aware power provisioning policy in the GPM. It may seem that GPM is merely executing a heuristic that may be biased toward particular application or lead to application starvation. For instance, say an island is provisioned 20% power and its utilization is 10%; the LPIC (described in next sub-section) would strive to increase the utilization by increasing the frequency so that the budgeted power is utilized. However, due to the limited DVFS knobs available on-chip, it may so happen that maximum power consumption in that island reaches only 18%. As a result, while computing the next power budget for this island, the GPM would realize this fact and provision less power budget. Essentially, the above formulation of budget allocation takes into account the performance vs. the allocated power consumed

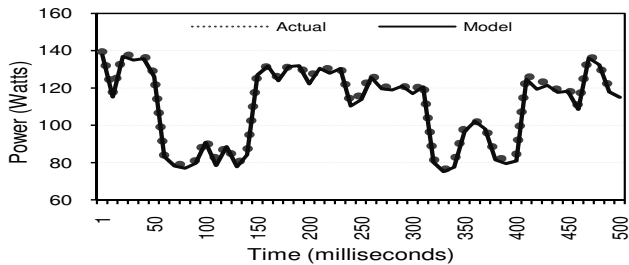


Fig. 5. Actual power consumption vs. model prediction.

in a particular interval. If the BIPS metric for an application was low with a high power budget in a particular interval, then the GPM will infer that the application does not require this high budget and hence will allocate the extra budget from this application to some other application.

Extending our performance-aware policy to some other policy can be done by simply augmenting the policy with additional constraints. For example, we could limit the maximum power allocation to a single island. In more concrete terms, the GPM could enforce a constraint that no single island can get more than $x\%$ of the total power budget and the local per-island controllers would continue to track such a power allocation. Thus, our approach provides enormous flexibility in terms of the GPM policy that is implemented. In addition to this performance-aware policy, in Section IV, we also discuss two other power provisioning policies, which are thermal and variation aware.

D. Local Per-Island Controller (LPIC)

Heuristic techniques and open-loop techniques work well with specific workloads or when an application's behavior is known beforehand, but may not work well for unanticipated workloads. Feedback control helps in such scenarios by monitoring the error, and guiding adaptation to minimize this error. Feedback control can, therefore, allow the adaptive response to adjust to a wide range of behaviors and can respond to unanticipated workloads or behavior in a predictable manner.

Motivated by this, we model the LPIC controllers as Proportional-Integral-Derivative (PID) controllers [10], [13]. The PID controller is probably the most widely used and well established class of feedback controllers. 'P', 'I' and 'D' refer to the three terms of the controller. These three terms are functions of the error signal and combine to produce a control signal. The intuitive design and practical relevance to design specifications is the primary reason for the wide spread adoption of PID control. A PID feedback control formulation takes the following form:

$$P_i(t) = P_i(t-1) + K_P \cdot e_i(t) + K_I \cdot \sum_{u=0}^{t-1} e_i(u) + K_D \cdot (e_i(t) - e_i(t-1)). \quad (7)$$

where K_P , K_I and K_D are design parameters subject to specified constraints, $e_i(t) = P_i(\text{alloc}) - P_i(\text{actual})$ denotes the error in the previous step, which is the difference between the actual power consumption and the power budget allocated by the GPM.

A proportional feedback control (P term) can reduce error responses to disturbances. This also provides robustness as the controller can adapt to minor mis-predictions by the models used. In our case, P term in the controller helps to bring the *controlled* power to the specified island budget. Proportional control still allows nonzero steady-state error to constant inputs (a standard difference between the

actual and reference power after the controller stabilizes). When the controller includes a term I , proportional to the integral of the error, this leads to the PI controller. In this case, the steady-state error can be eliminated, though typically at the cost of some deterioration in the dynamic response which might lead to undesired effects like abrupt changes in the input variables. In our case this leads to abrupt changes in frequency/voltage. By further adding a term D , proportional to the derivative of the error signal, which can often provide damping towards overshoot, we obtain the PID controller.

System Modeling: The design parameters K_P , K_I , are K_D can be computed accurately given a system model and design specifications like the maximum tolerable steady-state error, maximum tolerable overshoot in the power, and the maximum settling time in terms of controller invocations using formal methodologies like Bode plots, root locus analysis or through the application of stability criterion. We use pole-placement analysis [10], where stability is guaranteed if poles of the closed-loop controller lie within a unit-circle in the z -domain.

We model the power consumption of a CMP with a difference relation that is proportional to frequency. Previous studies [37], [31] have shown that, for limited DVFS range provided on chip, the difference relationship between power consumptions in successive intervals can be approximated linearly as a function of frequency. Based on this, we model the power consumption of a core in island i with a difference relation as:

$$P(t+1) = P(t) + a_i \cdot d(t), \quad (8)$$

where the term $d(t) = f(t+1) - f(t)$ and represents the difference in frequencies between two consecutive intervals. The term a_i , called the system gain, may vary across different islands for different workloads. In our case, we chose a_i using Equation (8) by running all PARSEC benchmarks except *bodytrack* (randomly chosen) and then taking the average of all a_i 's. In this way we determined a_i to be 0.79. To verify the accuracy of this model, we compared the actual power consumption of the CMP with the modeled power consumption by running *bodytrack* in all islands with added random white-noise [37] to change the DVFS levels of the cores in a random manner. Figure 5 shows that our system model is quite accurate with an average error well within 1%. Equation (8) thus represents the open-loop model of our system, and based on a PID controller, we derive the closed-loop transfer function of our system and evaluate its stability.

When using control-theory and transfer functions it is often easier to analyze the system dynamics after transforming the system parameters into the z -domain which transforms the discrete time-domain parameters into a frequency-domain representation. Transforming the open loop model of our system into the z -domain using Equation (8) gives us:

$$P(z) = \frac{a_i}{z-1}, \quad (9)$$

where z is the z -domain representation of t . Similarly, transforming the PID controller model from time-domain to frequency-domain gives us:

$$C(z) = K_P + \frac{K_I(z)}{z-1} + \frac{K_D(z-1)}{z}. \quad (10)$$

Using control-theory techniques, the *transfer function* of our closed-loop system in z -domain is:

$$Y(z) = \frac{P(z) \cdot C(z)}{1 + P(z) \cdot C(z)}. \quad (11)$$

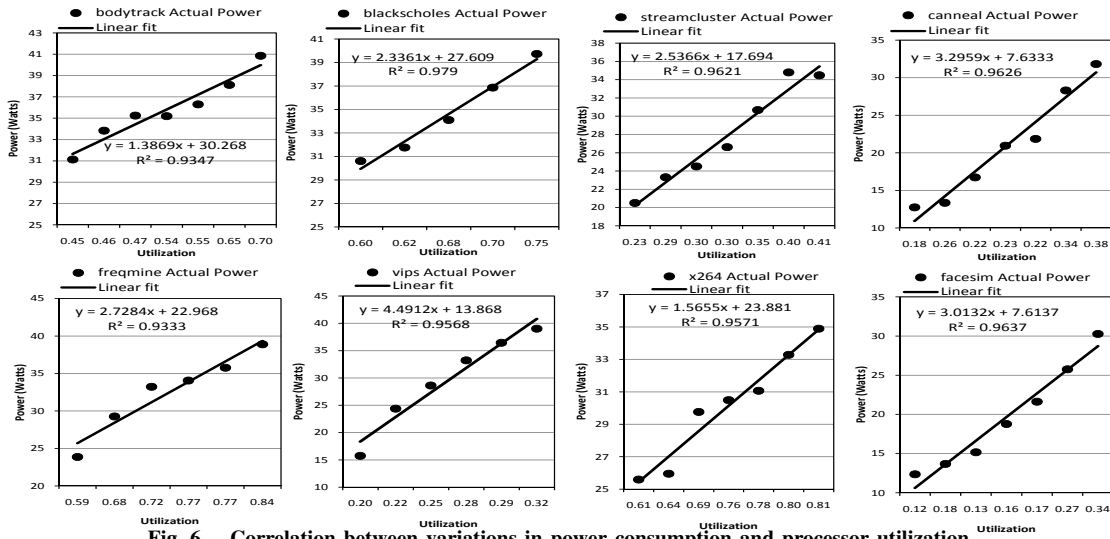


Fig. 6. Correlation between variations in power consumption and processor utilization.

From a system stability point of view, we require that the roots of the denominator in Equation (11) should lie within a unit circle in z -domain. These roots are called *poles* in control theory terminology and we use Matlab to determine the values of K_P , K_I and K_D such that the resulting system is always stable. We chose the values of K_P , K_I and K_D to be 0.4, 0.4 and 0.3, respectively, in all our evaluations such that the closed-loop poles of the transfer function lie within an unit-circle in the z -domain (in our case the closed-loop poles were -0.3366 , $0.7338 + 0.4069i$ and $0.7338 - 0.4069i$ with $a_i = 0.79$). The resulting transfer function then becomes:

$$Y(z) = \frac{0.869(z^2 - 0.9091z + 0.2727)}{(z + 0.3366)(z^2 - 1.468z + 0.704)}. \quad (12)$$

Stability Guarantees: As mentioned above, the term a_i may vary at runtime for different systems and different workloads. However, if the net-gain of the system lies within a certain range, then our controller will always guarantee stability. To show this, let us assume that the new system gain changes from a_i to $g' \cdot a_i$. With our chosen a_i to be 0.79 and values of K_P , K_I and K_D as mentioned above, we analyzed the stability of this new system and found that for $0 < g' < 2.10$ the system will always be stable and guarantee desired performance. Specifically, we analyzed the closed-loop poles of the new transfer function and chose the values of g' such that the closed-loop poles always lie within a unit circle. The transfer function of the system when g' is 2.10 is given in Equation (13). Any further increase in the value of g' increases the root of the first factor in the denominator of Equation (13) beyond unity, thus, making the system unstable.

$$Y(z) = \frac{1.825(z^2 - 0.9091z + 0.2727)}{(z + 0.9931)(z^2 - 1.168z + 0.5012)}. \quad (13)$$

Thus, an important advantage of formal feedback control is that we can keep the values of the parameters such as steady-state error, maximum overshoot and settling time under control as long as g' is within the specified range. What this means in the context of our CMP power management problem is that our local controllers will come very close to their GPM-allocated power budgets very quickly and, unlike other open-loop and ad-hoc schemes, our scheme will always have predictable behavior.

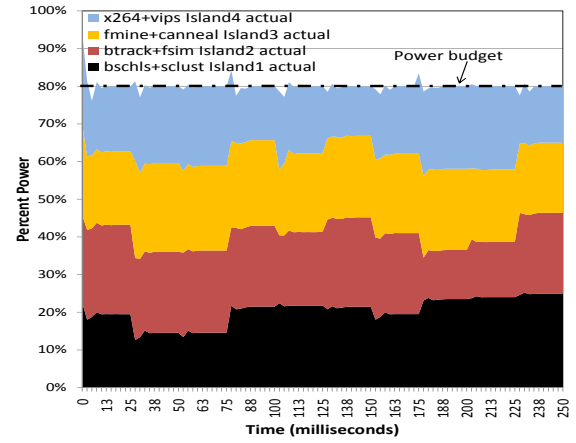


Fig. 7. Illustration of power provisioning where the power is provisioned among four islands with $P_{target} = 80\%$ of total power.

Sensor/Transducer: In our design of LPIC controllers, power is not a “measurable” output from an island. In fact, in real CMP systems it would be hard to measure power of individual islands directly. Hence, we need to look for other observable parameters like processor utilization that can be measured in order to provide the feedback. The system would then be observable since, the measured output variable (processor utilization) can reflect the system’s inherent states i.e. power consumption. Further, it is also important to have a transducer as the observable output parameter (processor utilization) is different from the reference input (power). Therefore, we need a model establishing the relationship between processor utilization and power.

In order to derive such a model, we measured the power versus processor utilization for different applications in our experimental suite (details of the experimental setup are provided in Section III). The plots in Figure 6 give the total power consumption (we used Watcht [5] for measuring dynamic power and HotLeakage [42] for measuring static power) versus CPU utilization. Each graph also gives the equation for the fitted regression line and the value for coefficient of determination, R^2 , of the fit. It can be seen from the figures that the power consumption depends almost linearly on the processor utilization. We also plotted the linear regression-line that is closest to the curve. We get an average R^2 value of 0.96, indicating

Technology	90 nm, 2 GHz (nominal)							
Core fetch/issue/commit width	4/2/2							
Register file size	80 entry							
Scheduler size	20 fp, 20 int							
L1 data cache	2-way, 16KB, 64B blocks, 1-cycle access delay							
L1 instruction cache	2-way, 16KB, 64B blocks, 1-cycle access delay							
L2 unified cache	shared, 512 MB per core, 16-way LRU, 64B blocks, 10-cycle access delay							
Memory	200 cycles access delay							
CMP configuration	8 x86 based out-of-order cores running Linux (4 islands, 2 cores per island)							
V	1.340	1.292	1.244	1.196	1.148	1.100	1.052	0.988
MHz	2000	1800	1600	1400	1200	1000	800	600

TABLE I Core, Memory, CMP configuration and Voltage(V)-Frequency(MHz) settings.

<p>PARSEC suite includes emerging RMS applications as well as large scale multi-threaded programs for CMPs. From this suite we choose six application and two kernel benchmarks. We used sim-large and native input sets and collected statistics from ROI point in the benchmarks.</p> <p>Application benchmarks details: blackscholes (bschls) uses PDE to solve an option pricing problem, bodytrack (btrack) tracks the body of a person, facesim (fsim) simulates motion of a human face, freqmine (fmime) does frequent item set mining, x264 is a video encoding app., vips is an image processing app.</p> <p>Kernel benchmark details: streamcluster (slust) does online clustering in an input stream and cannal simulates cache aware annealing to optimize routing cost in a chip design.</p>

TABLE II PARSEC benchmark details.

a very good linear fit of utilization with power. Therefore, we use this model in order to implement the transducer that converts the measured utilization into power in terms of a simple function of the form $P_i = k_0 \times U_i + k_1$, where P_i is the power of island I_i , U_i is the utilization of island I_i , and k_0 and k_1 are constants derived from the regression analysis of the models. With this model playing the role of the sensor/transducer in the LPIC controller, the converted value of power is fed back to the controller that computes the error between the actual power for this step and the reference value. Thus, the feedback loop is completed and the controller continues to adapt to changing workload characteristics by modulating the frequency/voltage levels according to the magnitude of this error.

III. Experimental Setup

We simulate a CMP comprising of voltage-frequency islands as depicted earlier in Figure 1. We used Simics [25] for performance simulations, Wattch [5] and HotLeakage [42] based models for power analysis. Each application is assigned to one core and the underlying operating system used is Linux. In our evaluations, we model each core and on-chip caches according to Table I. When using Wattch for power analysis, we used the linear clock-gating scheme with 20% power utilization for unused components. In our set-up, Wattch is integrated with Simics and we use *g-cache* modules in to model caches. We use the OPAL module in GEMS [26] to model cores. In our baseline evaluations, the GPM is invoked every 25 milliseconds and the LPIC is invoked every 2.5 milliseconds. The overhead of each DVFS interval is set to 0.5% of the CPU time based on [35], during which we assume no instructions are executed. Note that our assumptions on timing overheads are conservative since, with recently proposed on-chip controllers [21], these overheads can be brought down to nano-second levels. For DVFS coordinates, we assume each island supports 8 voltage-frequency pairs as listed in Table I, from 600 MHz to 2.0 GHz based on Pentium-M datasheet [1].

We chose 8 benchmarks from the PARSEC suite [2] and grouped them together so that there is variability among groups. The benchmark details are mentioned in Table II. Table III(a) shows our default

grouping of benchmarks that are scheduled across different islands and their characteristic. We found that when we use the *native* input set, the benchmarks become memory intensive. In particular, we used the *sim-large* input set for the CPU intensive benchmarks and the *native* input sets with memory intensive benchmarks and collected all the statistics after the *region-of-interest* as annotated in the benchmarks. We grouped the benchmarks so that each island has a cpu-bound and a memory-bound application. Later, we also report results from our experiments that use a different grouping of applications as shown in Table III(b). Finally, our experiments with 16,32 core CMPs and 4 cores per island uses the benchmark mix shown in Table III(c).

IV. Experimental Results

In this section, we present the results of our experimental evaluations. We measured the efficiency of the GPM in tracking the goal of a fixed chip wide budget as well as that of the LPICs in tracking the GPM-allocated power budget within an island. We report the performance of the controller in terms of three parameters: the maximum power overshoot, settling time, and the steady-state error. We also measure the performance degradation of the applications under different overall power budgets. Unless otherwise specified, the default power provisioning policy used in our experiments is the performance-aware policy explained in Section II.

Figure 7 illustrates how the GPM distributes the total power budget (by default, 80% of the maximum chip power) across four islands dynamically, under the default values of our simulation parameters. The plot shows that the power required for each island varies in each interval and this variation is captured by the GPM, which in turn provisions power for each island so as to maximize performance subject to the power budget in a particular interval. In this evaluation, we used our default application mixes (Table III(a)). If the current provisioned power in an island is less than that in the previous interval, then the controller tries to cap the power by reducing the voltage/frequency setting for all cores in the island. On the other hand, if more power is provisioned by the GPM for an island compared to the power provisioned in the previous interval, then the local controller for that island increases the voltage/frequency level in the island. Decreasing the voltage/frequency levels in an island slows down the execution speed for all applications scheduled in that island and also reduces power consumption. Recall that the goal of our performance-aware policy is to maximize overall instruction throughput while ensuring that the overall power consumption is limited by the specified power budget.

The dynamic power provisioning and capping or raising of power levels is shown in detail for our default configuration (8 core CMP with 2 cores per island) in Figure 8. In this evaluation, the total power budget is 80% of the required power by the whole chip. Initially 20% power is allocated to each of the four islands and then the GPM provisions power in subsequent intervals based on performance of islands in the past intervals. The results clearly show that the LPIC is able to effectively track the power provisions made by the GPM dynamically. These figures also depict the dynamic power needs of islands in detail; while island 1's power demand increases from 20% to 25%, island 3's power demand steadies around 19% and island 4's demand varies between 13% and 22%. Our proposed two-tier solution tracks this dynamic requirement of power budget at first tier and controls the provisioned budget at the second tier.

In order to measure the robustness of the LPIC, we next focus on the invocations of LPIC between two successive invocations of the GPM. Figure 9 plots the tracking of target power by the LPIC between two successive invocations of the GPM. We find that our LPIC controller

Island	Benchmarks	Characteristics
1	bschls, sclust	C, M
2	btrack, fsm	C, M
3	fmime, canneal	C, M
4	x264, vips	C, M

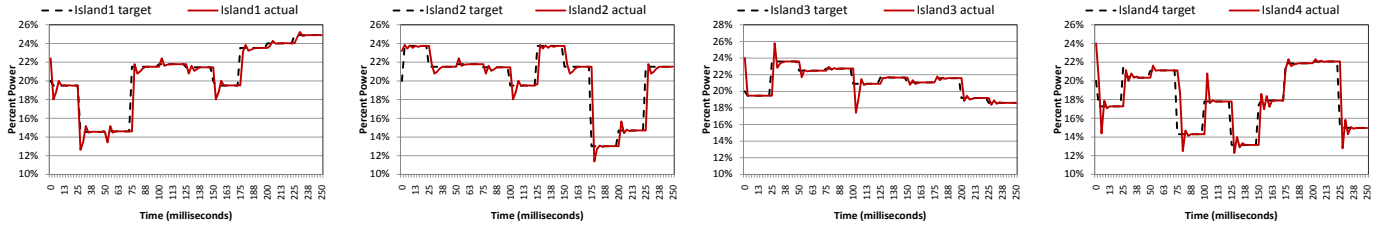
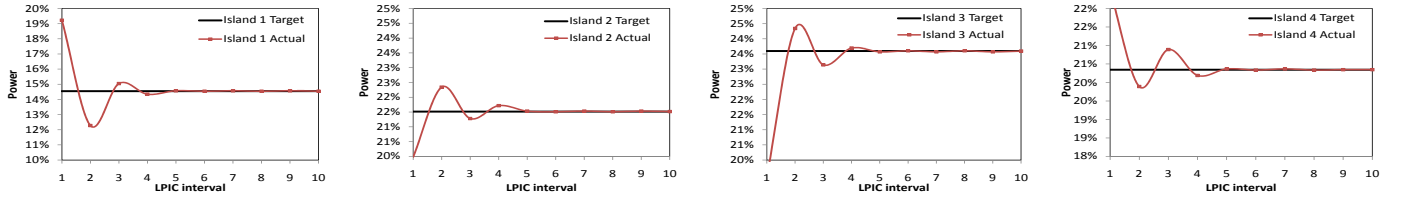
(a) Mix-1 for 8 core CMP.

Island	Benchmarks	Characteristics
1	bschls, btrack	C,C
2	sclust, fsm	M, M
3	fmime, x264	C, C
4	canneal, vips	M, M

(b) Mix-2 for 8 core CMP.

Island	Benchmarks	Characteristics
1	bschls, btrack, fmime, x264	C, C, C, C
2	sclust, fsm, canneal, vips	M, M, M, M
3	bschls, btrack, fmime, x264	C, C, C, C
4	sclust, fsm, canneal, vips	M, M, M, M

(c) Mix-3 for 16/32 core CMP.

TABLE III Application mix and island assignment for 8,16 and 32 core CMP configuration; C= CPU bound and M= Memory bound.**Fig. 8.** Tracking the target power in each island with time on our default configuration with 8 cores and 2 cores per island. The x-axis represents 10 invocations of the GPM at an interval of 25 ms and 100 invocations of the LPICs at an interval of 2.5 ms.**Fig. 9.** Tracking the target power in each island with time on our default CMP configuration with 8 cores and 2 cores per island. The x-axis represents 10 invocations of the LPIC at intervals of 2.5 ms between two successive invocations of the GPM.

does a neat job of controlling the island power to the desired target. There are overshoots in power at the start of each interval, especially if the current power target in an interval is more than the past interval target. However, these overshoots are mostly within 2% of the target. We also see that the actual power consumption in an interval settles down to the target power for that interval within 5-6 invocations of the LPIC, i.e., the steady state error is reduced to almost 0% within 5-6 controller invocations. Such guarantees in tracking power, in the face of changing workload dynamics, is assured by our controller since it is derived based on well established control theoretical foundations.

Figure 10 shows the goodness of the LPICs and their contribution to tracking the overall power budget with time. In this plot, the sum of the actual power consumption in each island is compared against the chip-wide power budget. The y-axis in the figure represents the actual chip-wide power consumption as a percentage of the maximum chip power, with a power budget of 80% used by the GPM. We observe that the overshoot and undershoot is mostly within 4% of the allocated power budget. Figure 11 shows the actual chip power consumption as compared against different power budgets. This curve shows that our power management policy closely tracks the budgeted power and never overshoots it.

For comparison purpose, we also implemented MaxBIPS [17], a recent work that manages chip wide power. When using MaxBIPS we used voltage/frequency settings as given in Table I. With MaxBIPS, given a power budget, the scheme selects DVFS co-ordinates from a *static prediction* table. Figure 11 also shows the budget curves with MaxBIPS. MaxBIPS's power consumption is always lower than the budget since, with this scheme, a DVFS knob is chosen to have a power consumption lower than the set-point, and with limited-DVFS knobs on-chip, a combination cannot always lead to power

consumption that is equal to budgeted power.

Figure 12 shows the percentage average performance degradation under different chip-wide power budgets against the case where no power management is done and all CPUs are allowed to operate at the maximum possible frequency. This scheme achieves better performance but may overshoot the power by a large degree (around 30-40% at 80% power budget). Our power management scheme incurs a 4% performance degradation with 80% power budget. The performance degradation incurred by our scheme depends on two factors. First, it depends on the applications co-scheduled on one island, where if cpu-bound applications are scheduled with memory-bound applications, then lowering down the voltage/frequency of the island is beneficial for the memory-bound application, but degrades performance of the cpu-intensive application. Second, performance degradation also depends on the number of cores per island. Specifically, if the number of cores per island is high, then the percentage degradation experienced by each application through lowering the voltage/frequency of an island would be comparable to the case when there are fewer cores per island. This aspect of performance degradation is further studied in Figure 13, where the degradation in performance for the same power budget increases as the number of cores per island increases. Note that, the core per island case corresponds to the architecture targeted in MaxBIPS. We found that, in this specific case, our approach and MaxBIPS generate similar results (our approach incurs 3.75% lesser performance degradation than MaxBIPS). However, as stated above, our target platform has multiple cores per island which is more scalable (from a power management perspective) as we move to larger CMPs and under such circumstances our scheme outperforms MaxBIPS as explained earlier.

We have so far studied the average performance degradation for

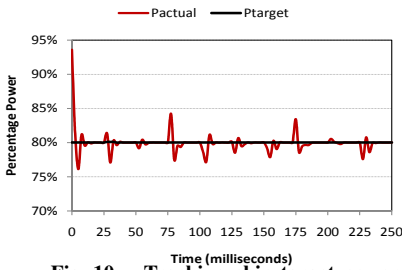


Fig. 10. Tracking chip target power.

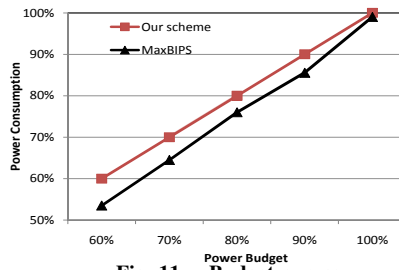


Fig. 11. Budget curves.

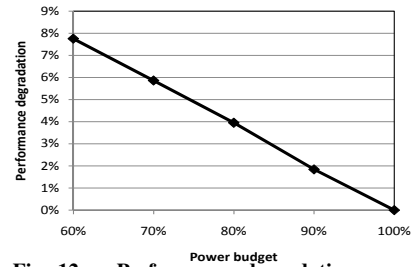


Fig. 12. Performance degradation vs. power targets.

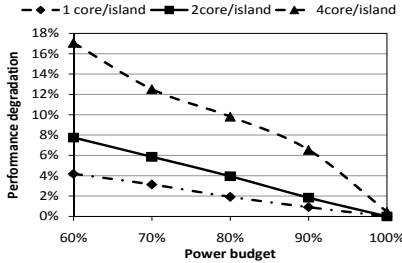


Fig. 13. Performance degradation vs. island size.

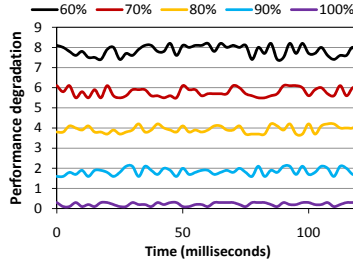


Fig. 14. Performance degradation with time.

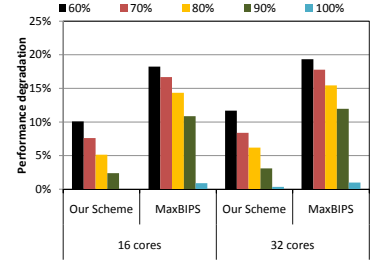


Fig. 15. Evaluation with 16 and 32 core CMPs.

different power budgets. However, in certain systems, it may also be important to track the performance degradation at any instant of time. Figure 14 shows the average performance degradation across all islands with time. We find that with 100% budgeted power, there is on an average 0.09% (maximum 0.19%) performance degradation. This is because of slight mis-predictions in our performance-aware power provisioning policy, where we predict the power to be provisioned on an island in forthcoming intervals. However, we find that our prediction scheme is quite accurate because of the feedback based prediction of performance in successive intervals. Overall, the results presented so far clearly show that we are able to track the specified power budgets with minimal performance penalty.

To test the scalability of our proposed approach, we also evaluated 16 and 32 core CMP configurations with 4 cores in each island. The application mix for 16 cores CMP is shown in Table III(c) and we replicate this mix twice for use with 32 cores CMP. With 16 and 32 cores, we found that our LPIC controller still tracked the power budget within 4% accuracy and had settling time of 4-5 controller invocations. The performance degradation with 16 and 32 core CMP is shown in Figure 15 for different power budgets. The degradation in performance is close to 4% with 80% power budget for 16 and 32 cores CMP. This figure also shows performance degradation with MaxBIPS (14% with 16 cores and 16.2% with 32 cores for 80% power budget). We find that our scheme is quite scalable in the sense that performance does not degrade too much with increase in cores.

Figure 16 shows the percentage of performance degradation in an 8 core CMP system, (2 cores per island) with two different benchmark mixes: Mix-1 and Mix-2 (Table III(b)). In Mix-2, two cpu-bound and two memory-bound applications are always scheduled in one island. The performance degradation with Mix-2 reduces compared to Mix-1 (see Table III(a)), since lowering the frequency of an island where two-memory bound applications are scheduled does not hurt performance as much when compared to the case of reducing the frequency of an island, where a memory-bound and a cpu-bound application are collocated.

We also performed experiments to analyze the sensitivity of the performance to the intervals of invocation of the GPM and LPIC. Recall

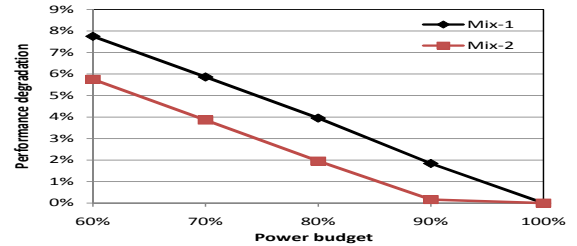


Fig. 16. Sensitivity to the mix of applications. Mix-1 and Mix-2 are given in Tables III(a) and III(b), respectively.

that, in our experiments so far, GPM and LPIC were invoked once at 20 ms and 2.5 ms, respectively. Figure 17 plots the performance degradation, when comparing our base values of 25 ms for GPM, 2.5 ms for LPIC against that with GPM interval of 50ms and LPIC interval of 5 ms. We observe that the performance degradation with GPM interval of 25 ms and LPIC interval of 2.5 ms is lesser due to more accurate predictions of power budget for the next cycle when predicted over smaller intervals of time. Making the interval too small also increases the overheads involved with invoking the controller.

A. Thermal-aware provisioning

As mentioned earlier, the GPM is flexible in terms of the policies that can be adopted for power provisioning. In addition to our performance-aware policy presented so far, we also evaluated a thermal-aware policy in a CMP environment using our GPM and LPIC based scheme. The policy in this evaluation was aimed at alleviating the problem of thermal hotspots in a CMP which is an important problem. To mitigate thermal hotspots, we employed a stringent policy in distributing power to nearby cores that always demanded a large-share of chip-wide power. For this evaluation, we considered 1 core per island in a 8 CMP organization, with core and cache configurations similar to Table I, and invoked GPM every 25 milliseconds and LPIC every 2.5 milliseconds. In this thermal-aware policy, we never provision more than 20% of total target power to two nearby islands for 2 successive intervals of 25 milliseconds each. Thus, in Figure 18(a), cores 1 and 2 cannot get more than 20% of chip-wide budget for more than 2 consecutive intervals. Similarly,

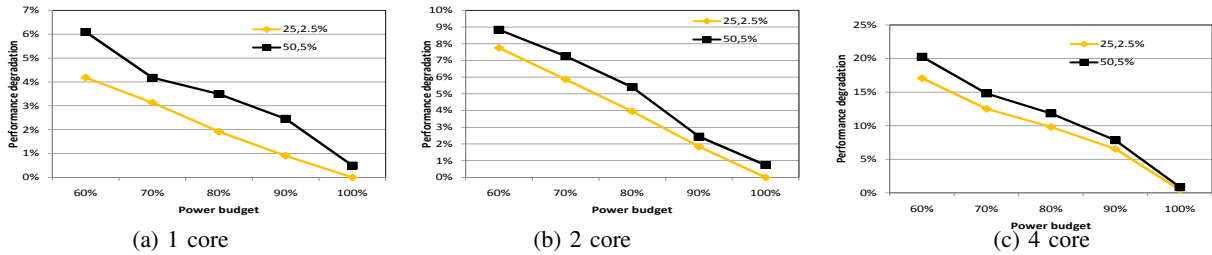


Fig. 17. Sensitivity to the intervals of invocation of the GPM and LPIC. (x, y) indicates (GPM invocation interval, LPIC invocation interval).

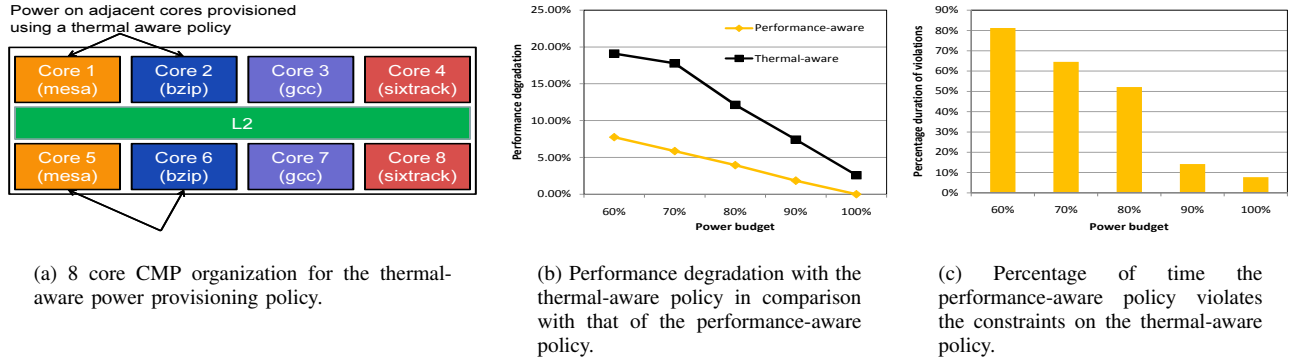


Fig. 18. Evaluation of a thermal-aware power provisioning policy.

cores 3 and 4 cannot get more than 20% of chip-wide power in two-consecutive GPM invocation cycles if they demand so. Additionally, a particular core cannot get more than 20% of the total power budget for 4 consecutive GPM invocation cycles. If these constraints are violated, we assume that a hotspot occurs.

We evaluated this policy using only cpu-bound applications i.e., *bschls*, *btrack*, *fmine* and *x264*, with each core running an application as shown in Figure 18(a). We find that, using such a policy along with our GPM and LPIC based CMP, chip-wide power budget is never exceeded and hot-spots never occur. We compare the performance degradation of using such a thermal aware power provisioning scheme with our performance-aware power provisioning scheme presented earlier. This is shown in Figure 18(b). As expected, the thermal aware policy incurs more performance penalty than the performance aware policy. Figure 18(c) shows the fraction of time during which the thermal policy was violated by the performance-aware policy between two successive GPM invocations. This analysis clearly shows that our feedback control based approach can accommodate different policies which have different performance and thermal trade-offs. Note that, we do not defend this particular, or any other power provisioning policy. This evaluation is meant to demonstrate the flexibility of our two-tiered feedback control based approach.

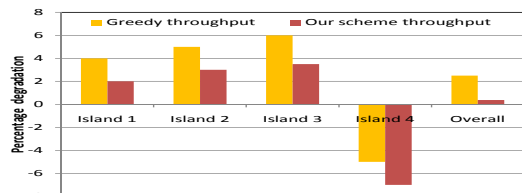
B. Variation-aware provisioning

Another important concern in power provisioning, given the recent technology trends, is the presence of process variations [3]. Technology scaling beyond 65nm is causing higher levels of device parameter variations, which are changing the design problem from deterministic to probabilistic. The demand for low power causes supply voltage scaling and hence making voltage variations a significant part of the overall challenge for CMPs [4], [16], [15]. Traditionally, variations in chip fabrication has been dominated by inter-die process variations and the problem was alleviated using speed-binning. However, intra-die variations are becoming increasingly common in recent technologies leading to power and performance variations among different cores

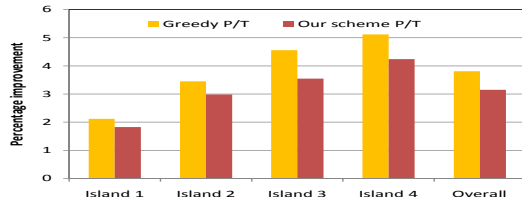
of a multicore processor [16]. Addressing intra-die variations requires that voltage frequency controllers be aware of such intra-die variations [15]. To illustrate the flexibility of our approach, we also made use of a variation-aware policy in the GPM to study the range of policies that can be adapted in the GPM.

We implemented a GPM power provisioning policy that is similar to the *greedy* approach presented in [15]. The policy described in [15] is based on a CMP extension by Herbert et al. [14] of the greedy-search scheme proposed by Magklis et al. in [24]. The particular policy used attempts to provision power to a voltage frequency island at a voltage-frequency level that minimizes power/throughput ratio, in presence of process variations, assuming that power/throughput ratio is a convex function of the voltage frequency level. In each interval, the GPM counts the number of non-spin instructions retired and our power estimation is used to approximate the energy consumed by the voltage frequency island over the interval, allowing the computation of energy per instruction. In every invocation, the GPM compares the value of energy per instruction in the current interval to that observed in the previous interval. If the energy per instruction value has improved during this time, the GPM makes another move in the same direction as the last one. If it has degraded in this interval, the GPM assumes that it has overshot the optimal level. It makes a transition in the opposite direction of the last one to the suspected optimal power provisioning and stays there for a fixed number of intervals (10 LPIC intervals). The GPM then continues exploration by making a move in the direction opposite the power provisioning which preceded the hold. We essentially attempt to operate the more leaky islands at lower V/F levels and less leaky islands at higher V/F levels.

Our target CMP configuration is same as shown in Figure 1 and we assume that the leakage current in Island 1, Island 2 and Island 3 is 1.2x, 1.5x and 2x, respectively, of Island 4. These leakage numbers roughly correspond to those assumed in [15]. We schedule Mix-1 applications (Table III(a)) in this setup and Figures IV-B and IV-B show the percentage degradation in throughput and percentage improvement in power/throughput respectively across the four VFI



(a) Percentage degradation in throughput.



(b) Percentage improvement in power/throughput (P/T).

Fig. 19. Evaluation of a variation-aware power provisioning policy.

islands. Our scheme slightly out-performs the greedy scheme solely because the LPIC in our case is better at regulating the power at the levels allocated by the GPM. As can be seen in these figures, Island 1, 2 and 3's throughput degrades since the GPM being variation aware, allocates less power to these islands and as a result LPIC can operate these islands at a lower V/F level. However, Island 4 which is the least leaky island is always allocated more power because of which it operates at higher V/F levels. This results in overall system throughput improvement and also better power/ throughput ratio. This analysis again shows the flexibility in our proposed tiered-control scheme which can be extended to handle variation aware CMPs in improving system throughput and reducing power/throughput ratio as well.

V. Related Work

A. Power and Thermal Management

Many previous works have focused on the design of power and thermal management schemes [8], [29], [23], [12], [28], [17]. These works also highlight the importance of adapting to workload behavior under power and thermal constraints. However, most of these works propose to adopt a combination of independent local schemes in the architecture or scheduling decisions in the operating system.

Meng et al. [28] proposed an adaptive, multi-optimization strategy for multi-core power management. They address the problem of meeting a global chip-wide power budget through run-time adaptation of configurable processor cores. Isci et al. [17] also used a global power manager, similar to ours, that provides a mechanism to sense the per-core power and performance state of the chip at periodic intervals; it then sets the operating power level or mode of each core to enforce adherence to known chip-level power budgets. However, the local monitoring used to enforce these power budgets were based on open loop control and does not provide the same robustness as the formal feedback control based LPICs in our work. Furthermore, the constraints on the achievable performance were less stringent as they perform per-core DVFS as opposed to per-island DVFS in our work where co-scheduled threads are influenced by the voltage scaling of an island.

B. DVFS for Multi-Clock Domain Architectures

Previous research has identified dynamic voltage frequency scaling as highly beneficial to effective power management of multi-clock domain architectures [34], [33], [11], [19]. In [19], Iyer et al. were able to assess the power/performance tradeoffs available for Multiple

Clock, Single Voltage (MCSV) as well as Multiple Clock, Dynamic Voltage (MCDV) cores. They showed that MCDV designs may be more power efficient than their fully-synchronous counterpart. Overall, by using fine grain clock speed and voltage adaptation, significant power savings can be achieved with a very small loss in performance. Semeraro et al. [33] proposed an online algorithm that reacts to the dynamic characteristics of the application to control domain frequencies. The algorithm uses processor queue utilization information and achieves improvements in energy delay product while degrading performance by small amounts.

C. Formal Feedback Control for Power Management

Donald and Martonosi [9] explored various thermal management techniques that exploit the distributed nature of multicore processors. They classify these techniques based on whether a policy is applied locally to a core or to the processor as a whole. The proposed design in [9] involves a PI-based core thermal controller and an outer control loop to decide process migrations on thermal emergencies. Juang et al. [20] extended [39] and proposed a distributed version of the scheme, applying the basic mechanics of the formal approach to CMPs.

In [38], Wang et al. propose a multi-input multi-output solution to CMP power management. One major difference between ours and the work by Wang et al. is that, while our approach is simple and flexible (in fact flexibility in the GPM policy was one of the prime design time consideration behind our proposed approach). Wang et al. use sophisticated controllers whose complexity is proportional to the number of cores in the chip. Consequently, it is hard to apply the same to really large CMPs. Also, we target a more constrained problem involving voltage/frequency islands in CMPs, where applications scheduled in one island are coupled together, whereas Wang et al. target single cores in CMPs. Our proposal is much more flexible in the sense that not only it can be used as a framework to manage power and thermal hot-spots, but also to optimize other system metrics like throughput in the presence of variation, etc.

VI. Conclusions

Power control in CMPs without significant performance degradation is a critical, but challenging problem. In this paper, we address this problem for a multiple clock domain CMP architecture, where each domain or island is controlled by a separate clock. We propose a two-tier power management architecture where, the first-tier *allocates* a target budget to each local island in the CMP and the second-tier *controls* the power at the target budget. We discuss the design details of these two tiers and demonstrate the advantages of such a feedback control scheme through experimental evaluations on 8, 16 and 32 core CMPs. The experimental evaluations show that the proposed control theoretic power management scheme tracks the chip power budget with high accuracy by limiting the maximum overshoot in power consumption of each island close to 4% of the targeted budget. In addition, the controller reaches the steady-state quickly in only four to five iterations. Additionally, to demonstrate the flexibility provided by our architecture we also show the applicability of our scheme to thermal management in CMPs and provisioning power in a CMP substrate that has intra-die leakage current variations.

Acknowledgement

The authors would like to thank the anonymous reviewers for their comments. This research is supported in part by NSF grants CNS #0720645, CCF #0811687, CCF #0702519, CNS #0202007 and CNS #0509251, and a grant from Microsoft Corporation.

References

- [1] Intel pentium m processor on 90 nm process with 2-mb l2 cache datasheet, january 2006. <http://download.intel.com/design/mobile/datashts/30218908.pdf>.
- [2] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, 2008.
- [3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *DAC '03: Proceedings of the 40th annual Design Automation Conference*, 2003.
- [4] K. A. Bowman, A. R. Alameldeen, S. T. Srinivasan, and C. B. Wilkerson. Impact of die-to-die and within-die parameter variations on the throughput distribution of multi-core processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 2007.
- [5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, 28(2), 2000.
- [6] A. Das, S. Ozdemir, G. Memik, and A. Choudhary. Evaluating voltage islands in cmps under process variations. In *ICCD '07: Proceedings of the International Conference on Computer Design*, 2007.
- [7] R. Das, J. O. Kephart, C. Lefurgy, G. Tesauro, D. W. Levine, and H. Chan. Autonomic multi-agent management of power and performance in data centers. In *Proceedings of the International Conference on Autonomous agents and multiagent systems*, 2008.
- [8] J. D. Davis, J. Laudon, and K. Olukotun. Maximizing cmp throughput with mediocre cores. In *PACT '05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*, 2005.
- [9] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, 2006.
- [10] G. F. Franklin, M. L. Workman, and D. Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., 1997.
- [11] S. Garg and D. Marculescu. System-level throughput analysis for process variation aware multiple voltage-frequency island designs. *ACM Trans. Des. Autom. Electron. Syst.*, 13(4), 2008.
- [12] M. Goma, M. D. Powell, and T. N. Vijaykumar. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. *SIGOPS Oper. Syst. Rev.*, 38(5), 2004.
- [13] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.
- [14] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED '07: Proceedings of the International Symposium on Low power Electronics and Design*, 2007.
- [15] S. Herbert and D. Marculescu. Variation-aware dynamic voltage/frequency scaling. In *Proceedings of the International Symposium on High-Performance Computer Architecture*, 2009.
- [16] E. Humenay, D. Tarjan, and K. Skadron. Impact of process variations on multicore performance symmetry. In *Proceedings of the conference on Design, Automation and Test in Europe*, 2007.
- [17] 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 Microarchitecture*, 2006.
- [18] A. Iyer and D. Marculescu. Power and performance evaluation of globally asynchronous locally synchronous processors. In *ISCA '02: Proceedings of the 29th annual international symposium on Computer architecture*, 2002.
- [19] A. Iyer and D. Marculescu. Power efficiency of voltage scaling in multiple clock, multiple voltage cores. In *Proceedings of the International Conference on Computer-Aided Design*, 2002.
- [20] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark. Coordinated, distributed, formal energy management of chip multiprocessors. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, 2005.
- [21] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*, 2008.
- [22] D. E. Lackey et al. Managing power and performance for system-on-chip designs using voltage islands. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM Intn. Conf. on Comp.-aided design*, 2002.
- [23] J. Li and J. F. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *HPCA '06: Proceedings of the Twelfth International Symposium on High-Performance Computer Architecture*, 2006.
- [24] G. Magklis, P. Chaparro, J. González, and A. González. Independent front-end and back-end dynamic voltage scaling for gals microarchitecture. In *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*, 2006.
- [25] P. S. Magnusson et al. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.
- [26] M. Martin et al. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 2005.
- [27] C. McNairy and D. Soltis. Itanium 2 processor microarchitecture. *IEEE Micro*, 23(2), 2003.
- [28] K. Meng, R. Joseph, R. P. Dick, and L. Shang. Multi-optimization power management for chip multiprocessors. In *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, 2008.
- [29] M. Monchiero, R. Canal, and A. Gonzalez. Power/performance/thermal design-space exploration for multicore architectures. *IEEE Trans. Parallel Distrib. Syst.*, 19(5), 2008.
- [30] K. Niyogi and D. Marculescu. Speed and voltage selection for gals systems based on voltage/frequency islands. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, 2005.
- [31] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News*, 36(1), 2008.
- [32] L. Ramos and R. Bianchini. C-oracle: Predictive thermal management for data centers. In *HPCA '08: Proceedings of the 14th IEEE Intern. Symp. on High-Performance Computer Archit.*, 2008.
- [33] G. Semeraro et al. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *MICRO 35: Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, 2002.
- [34] G. Semeraro et al. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *HPCA '02: Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, 2002.
- [35] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. H. S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim. (TACO)*, 2004.
- [36] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ISCA*

- '08: *Proceedings of the 35th International Symposium on Computer Architecture*, 2008.
- [37] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *HPCA '08: Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture*, 2008.
- [38] Y. Wang, K. Ma, and X. Wang. Temperature-constrained power control for chip multiprocessors with online model estimation. In *ISCA '09: Proceedings of the 36th International Symposium on Computer Architecture*, 2009.
- [39] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark. Formal online methods for voltage/frequency control in multiple clock domain microprocessors. *SIGARCH Comput. Archit. News*, 32(5):248–259, 2004.
- [40] Z. Yu and B. Baas. Implementing tile-based chip multiprocessors with gals clocking styles. *ICCD '06: International Conference on Computer Design*, 2006.
- [41] Z. Yu and B. M. Baas. Performance and power analysis of globally asynchronous locally synchronous multi-processor systems. In *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, 2006.
- [42] Y. Zhang et al. Hotleakage: A temperature-aware model of sub-threshold and gate leakage for architects, Tech. Report CS-2003-05, Univ. of Virginia, 2003.