Temperature-Constrained Power Control for Chip Multiprocessors with Online Model Estimation

Yefu Wang, Kai Ma, and Xiaorui Wang
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville, TN 37996
{ywang38, kma3, xwang}@eecs.utk.edu

ABSTRACT

As chip multiprocessors (CMPs) become the main trend in processor development, various power and thermal management strategies have recently been proposed to optimize system performance while controlling the power or temperature of a CMP chip to stay below a constraint. The availability of per-core DVFS (dynamic voltage and frequency scaling) also makes it possible to develop advanced management strategies. However, most existing solutions rely on open-loop search or optimization with the assumption that power can be estimated accurately, while others adopt oversimplified feedback control strategies to control power and temperature separately, without any theoretical guarantees. In this paper, we propose a chip-level power control algorithm that is systematically designed based on optimal control theory. Our algorithm can precisely control the power of a CMP chip to the desired set point while maintaining the temperature of each core below a specified threshold. Furthermore, an online model estimator is designed to achieve analytical assurance of control accuracy and system stability, even in the face of significant workload variations or unpredictable chip or core variations. Empirical results on a physical testbed show that our controller outperforms two state-of-the-art control algorithms by having better SPEC benchmark performance and more precise power control. In addition, extensive simulation results demonstrate the efficacy of our algorithm for various CMP configurations.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Design studies

General Terms

Design, Management, Performance, Experimentation

Keywords

Chip multiprocessor, power management, feedback control

1. INTRODUCTION

Chip-level multiprocessors (CMPs), *i.e.*, multi-core processors, have arguably become the main trend in the current processor development, due to some well-known technology barriers such as "Power Wall" and "ILP (instruction-level parallelism) Wall". However, power and thermal still remain the major constraints for the further throughput improvement of CMP processors. First, the peak power consumption of a CMP processor often needs to be controlled to enable higher computing densities. For example, Montecito, a dual-core product, has a reduced power envelope of 100 W while its single-core predecessor has a 130 W power envelope [21]. In addition, precisely controlling the power of processors is an important way to allow desired power shifting among different components in a server for optimized overall system performance. Second, the temperature of a CMP processor also must be kept lower than a threshold, because the increased heat dissipation with multiple cores integrated in a single die may result in a higher possibility of thermal failures. The power and thermal problems are also being exacerbated by the rapidly increasing level of core integration in the CMP design. Therefore, effective control algorithms need to be developed to maximize the performance delivered per watt while controlling both the power consumption and temperature of a CMP chip to stay below respective allowed thresholds.

There are several major challenges in developing power and thermal control algorithms for CMP processors. First, multiple cores may need to be manipulated simultaneously to control both power and temperature for a CMP chip. For example, the small overhead (in microseconds or even nanoseconds in the near future [15]) of per-core DVFS (dynamic voltage and frequency scaling) makes it possible to simultaneously change the DVFS levels of multiple cores. Hence, we need Multi-Input-Multi-Output (MIMO) control strategies. Second, optimal control algorithms need to be designed for power shifting among different cores based on their performance needs. This is particularly important because most today's application software is designed for single-core processors, and so cannot use multiple concurrent threads intensively. As a result, different cores usually have significantly different workload intensities. In addition, there could be within-die variations for different cores or even heterogeneous cores. An effective control strategy should be able to utilize the core variations to optimize the overall processor performance. Third, different cores may be coupled together due to application requirements or hardware constraints. For example, some processors have two cores coupled together and so their DVFS levels can only be changed to the same level. Those constraints need to be accounted for in order for the control strategies to be valuable in practice. Fourth and more importantly, since the workload of a CMP processor is unpredictable at design time and may vary significantly at runtime, control algorithms cannot rely on static power models and must be self-adaptive to workload variations for optimal control performance. Finally, as power and temperature both are critical, control accuracy and system stability must be analytically assured.

ISCA'09, June 20–24, 2009, Austin, Texas, USA.

In recent years, various power and thermal management strategies have been proposed for CMP processors. Most existing work focuses on minimizing the power consumption of CMPs. As a result, they cannot provide any explicit guarantees for the power and temperature of a CMP to stay below the thresholds. Some prior research (*e.g.*, Intel's Foxton technology [21]) has successfully controlled the power and temperature of a processor using chip-wide DVFS. However, those solutions rely on Single-Input-Single-Output (SISO) control methods and thus cannot be directly applied to CMP chips where per-core DVFS is available for better management. Several recently proposed algorithms [11, 32] present open-loop search or optimization strategies, with the assumption that power consumption of a CMP at each DVFS level can be estimated accurately. While those solutions can work effectively when the system is running workloads that are same or similar with the one used to do power estimation, they may result in severe performance degradation or even power constraint violation when workloads vary significantly. While some closed-loop solutions have also been proposed for CMPs based on heuristics [22, 11], basic SISO control theory has been adopted for processor thermal management [29]. Feedback control theory is an effective tool for power and temperature control due to its theoretically guaranteed control accuracy and system stability. Control theory also provides standard controller design approaches, *e.g.*, standard ways to choose the right control parameters, such that exhaustive iterations of tuning and testing can be avoided.

In this paper, we propose a chip-level temperature-constrained power control algorithm that is systematically designed based on well-established Model Predictive Control (MPC) theory, which is an advanced optimal MIMO control theory. Our algorithm can precisely control the power of a CMP chip to the desired set point while maintaining the temperature of each core below a specified threshold. To our best knowledge, our paper presents the first study of applying advanced MIMO control theory to power and thermal control at the chip level in a rigorous way with theoretical guarantees. Specifically, compared to existing work on CMP power and thermal management, this paper makes the following four new contributions:

- While most existing work addresses power and temperature separately, our solution can optimize the processor performance while controlling the total power of a CMP chip to stay below a budget and maintaining the temperature to be lower than a constraint.

- While most existing work heavily relies on open-loop optimization or heuristics, we design and analyze a temperature-constrained power control algorithm based on MPC MIMO control theory for theoretically guaranteed system stability and control accuracy. We also present a detailed analysis to establish a stability condition in the event of power model variations. This rigorous design methodology is in sharp contrast to heuristic-based adaptive solutions that heavily rely on extensive manual tuning.

- While most existing work depends on static models for power and temperature control, we design an online model estimator to update the system model at runtime. As a result, our solution can achieve analytical assurance of desired control performance, even in the face of significant workload variations or unpredictable chip and within-die core variations.

- While most existing work has only simulation results, we present empirical results to demonstrate that our controller outperforms two state-of-the-art power controllers: a prediction-based algorithm and a heuristic-based controller. We also present the system architecture of our control loop on a physical testbed and the implementation details of each component. In addition, extensive simulation results also demonstrate the efficacy of our algorithm for various CMP configurations.

The rest of the paper is organized as follows. Section 2 introduces the overall architecture of our power control loop. Section 3 describes the system modeling, design, and analysis of our controller. Section 4 presents the design of the online model estimator. Section 5 provides the implementation details of each component in the control loop. Section 6 presents the results of our empirical experiments conducted on a physical testbed and extensive simulation results for various CMP configurations. Section 7 highlights the distinction of our work by discussing the related work. Section 8 concludes the paper.

2. TEMPERATURE-CONSTRAINED POWER CONTROL LOOP

In this section, we give a high-level description of our power control loop that adaptively manages the power consumption and temperature of a CMP by conducting per-core DVFS.

There are two major reasons for us to use per-core DVFS as our actuation method. First, recent studies [15] have shown that the overhead of per-core DVFS can be in nanoseconds and is thus small enough to be used in real systems. Second, most today's application software is developed with a single thread (called worker thread) doing most computation-intensive work. As a result, it is currently less feasible for other actuation methods, such as thread mapping, to be used in practice for power and thermal control. We plan to extend our control architecture to include other actuation methods in our future work.

As shown in Figure 1, the key components in the power control loop include a *temperature sensor* on each core, a *power monitor* connected to the power supply circuit of the CMP, an MPC *power controller*, and an *online model estimator*. The power controller and the estimator can be implemented in the service processor firmware. The control loop is invoked periodically and its period is chosen based on a trade-off between actuation overhead and system settling time. The following feedback control loop is invoked at the end of every control period. 1) The power monitor (*e.g.*, an on board power measurement circuit) measures the average power consumption of the whole CMP chip in the last control period and then sends the value to the controller. The total power consumption is the *controlled variable* of the control loop. 2) The temperature sensor on each core sends the temperature of this core in the last control period to the controller. The temperature values are used by the power controller as constraints to manage core temperatures. 3) System-level performance of each core, such as its CPU utilization or MIPS (Million Instructions Per Second) value, is measured by the monitor on each core and sent to the controller. Those measured values can be used by the controller to optimize power allocation among different cores based on their performance needs. 4) Based on the collected power value, temperature vector, and performance value vector, the controller computes the new DVFS level for each core in the CMP chip and then sends the levels to the DVFS modulator on the service processor. The *DVFS levels* of the cores are the manipulated variables of the control loop. 5) The DVFS modulator changes the DVFS level of each core accordingly. 6) The online model estimator updates the power system

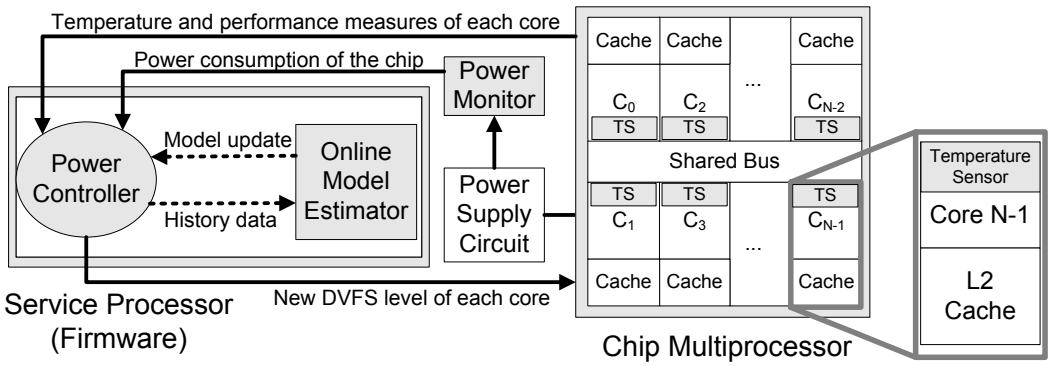


Figure 1: Temperature-constrained power control loop for a CMP with N cores.

model used by the controller based the measured power consumption and the DVFS levels.

Since the core of the control loop is the MPC power controller, we focus on the system modeling and controller design in the next section. The implementation details of other components are provided in Section 5.

3. CONTROLLER DESIGN AND ANALYSIS

In this section, we introduce the modeling, design, analysis, and constraints of the MPC power controller.

3.1 System Modeling

In this section, we model the power consumption of a CMP processor. We first introduce some notation. T is the control period. $p_i(k)$ is the power consumption of Core i in the k^{th} control period. $f_i(k)$ is the DVFS level of Core i in the k^{th} control period. $\Delta f_i(k)$ is the difference between $f_i(k+1)$ and $f_i(k)$, *i.e.*, $\Delta f_i(k) = f_i(k+1) - f_i(k)$. N is the total number of cores in the CMP. $cp(k)$ is the power consumption of the whole CMP processor, *i.e.*, $cp(k) = \sum_{i=1}^{N} p_i(k)$. P_s is the power set point, *i.e.*, the desired power constraint of the CMP processor. The control goal is to guarantee that $cp(k)$ converges to P_s within a given settling time.

In order to have an effective controller design, it is important to model the dynamics of the controlled system, namely the relation between the controlled variable (*i.e.*, $cp(k)$) and the manipulated variables (*i.e.*, $f_i(k), 1 \leq i \leq N$). It is well-known that DVFS can allow cubic reductions in power density relative to performance loss for each core in a CMP [31]. However, a cubic power model may lead to high complexity for controller design and large runtime overhead. On the other hand, real CMP processors usually only provide a limited DVFS range. Within the small range, previous studies [25, 34] have shown that the relationship between power and DVFS level can be approximated with a linear function. Therefore, the power consumption of a core is modeled as:

$$p_i(k) = a_i f_i(k) + c_i \tag{1}$$

where a_i is a generalized parameter that may vary for different cores. The dynamic model of the system as a difference equation is:

$$p_i(k+1) = p_i(k) + a_i \Delta f_i(k) \tag{2}$$

Based on (2), we now consider the total power consumption of all cores in a CMP processor. Their power consumptions can be modeled in the matrix form:

$$\mathbf{p(k+1)} = \mathbf{p(k)} + \mathbf{A\Delta f(k)} \tag{3}$$

where:

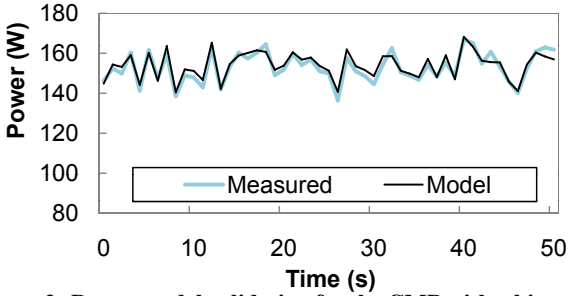


Figure 2: Power model validation for the CMP with white noise input.

$$\mathbf{p(k)} = \left[\begin{array}{cccc} p_1(k) & \dots & p_N(k) \end{array}\right]^T,$$
$$\mathbf{A} = diag[a_1 \dots a_N],$$
$$\mathbf{\Delta f(k)} = \left[\begin{array}{cccc} \Delta f_1(k) & \dots & \Delta f_N(k) \end{array}\right]^T.$$

The total power consumption of the CMP, $cp(k+1)$, is the summation of the power consumed by each core.

$$cp(k+1) = cp(k) + \left[\begin{array}{ccc} a_1 & \dots & a_N \end{array}\right] \left[\begin{array}{c} \Delta f_1(k) \\ \vdots \\ \Delta f_N(k) \end{array}\right]. \tag{4}$$

In a real system, the *estimated* system parameters $\mathbf{A}$ can be the results of system identification using a typical workload. In our experiments, we conduct system identification for all the benchmarks of the SPEC CPU2006 suite and receive a set of parameters for each benchmark. We then use the average values of all the benchmarks as our *estimated* system parameters on our dual-core CMP testbed. Specifically, $\mathbf{A} = diag[52.38, 49.32]$. To verify the accuracy of our system model, we run a randomly selected benchmark *gobmk* and stimulate the CMP chip with pseudo-random digital white-noise inputs [8] to change the DVFS levels of the cores in a random manner. We then compare the actual power consumption with the values predicted by our model. Figure 2 shows that the predicted power is adequately close to the actual power of the CMP processor. Note that the *actual* values of $\mathbf{A}$ in a real system may change for different workloads and are *unknown* at design time. However, in Section 3.4, we prove that a system controlled by the controller designed with the estimated parameters can remain stable as long as the variations of $\mathbf{A}$ are within an allowed range. In addition, as discussed in Section 4, our online model estimator can also dynamically correct the system parameters used by the MPC controller based on measured power data.

3.2 MPC Controller Design

We apply *Model Predictive Control* (MPC) theory [20] to design the controller based on the system model (4). MPC is an advanced

control technique that can deal with coupled MIMO control problems with constraints on the plant and the actuators. This characteristic makes MPC well suited for power control of a CMP.

A model predictive controller optimizes a *cost function* defined over a time interval in the future. The controller uses the system model to predict the control behavior over P sampling periods, called the *prediction horizon*. The control objective is to select an input trajectory that minimizes the cost function while satisfying the constraints. An input trajectory includes the control inputs in the following M sampling periods, $\mathbf{\Delta f(k)}$, $\mathbf{\Delta f(k+1|k)}$, ..., $\mathbf{\Delta f(k+M-1|k)}$, where M is called the *control horizon*. The notation $x(k+i|k)$ means that the value of variable x at time $(k+i)T$ depends on the conditions at time kT. Once the input trajectory is computed, only the first element $\mathbf{\Delta f(k)}$ is applied as the control input to the system. At the end of the next sampling period, the prediction horizon slides one sampling period and the input is computed again based on the feedback $cp(k)$ from the power monitor. Note that it is important to re-compute the control input because the original prediction may be incorrect due to uncertainties and inaccuracies in the system model used by the controller. MPC enables us to combine performance prediction, optimization, constraint satisfaction, and feedback control into a single algorithm.

The controller includes a least squares solver, a cost function, a reference trajectory, and a system model. At the end of every sampling period, the controller computes the control input $\mathbf{\Delta f(k)}$ that minimizes the following cost function under constraints.

$$V(k) = \sum_{i=1}^{P} \|cp(k+i|k) - ref(k+i|k)\|^2_{Q(i)} + \sum_{i=0}^{M-1} \|\mathbf{\Delta f(k+i|k)} + \mathbf{f(k+i|k)} - \mathbf{F_{max}}\|^2_{\mathbf{R(i)}} \quad (5)$$

where P is the prediction horizon, and M is the control horizon. $Q(i)$ is the *tracking error weight*, and $\mathbf{R(i)}$ is the *control penalty weight vector*. The first term in the cost function represents the *tracking error*, *i.e.*, the difference between the total power $cp(k+i|k)$ and a reference trajectory $ref(k+i|k)$. The reference trajectory defines an ideal trajectory along which the total power $cp(k+i|k)$ should change from the current value $cp(k)$ to the set point P_s (*i.e.*, power budget of the CMP). Our controller is designed to track the following exponential reference trajectory so that the closed-loop system behaves like a linear system.

$$ref(k+i|k) = P_s - e^{-\frac{T}{T_{ref}}i}(P_s - cp(k)) \quad (6)$$

where T_{ref} is the time constant that specifies the speed of system response. A smaller T_{ref} causes the system to converge faster to the set point but may lead to larger overshoot. By minimizing the tracking error, the closed-loop system will converge to the power set point P_s if the system is stable. The second term in the cost function (5) represents the *control penalty*. The control penalty term causes the controller to optimize system performance by minimizing the difference between the highest DVFS levels, $\mathbf{F_{max}}$, and the new DVFS levels, $\mathbf{f(k+i+1|k)} = \mathbf{\Delta f(k+i|k)} + \mathbf{f(k+i|k)}$ along the control horizon. The control weight vector, $\mathbf{R(i)}$, can be tuned to represent preference among cores. For example, a higher weight may be assigned to a core if it has heavier or more important workload so that the controller can give preference to increasing its DVFS level. As a result, the overall system performance can be optimized. In our experiments on the testbed, we use the CPU utilization of each core as an example weight. In our simulations, we use the MIPS measurement of each core as another example weight.

This control problem is subject to four sets of constraints. First, the DVFS level of each core should be within an allowed physical range (*e.g.*, 2GHz to 3GHz). Second, the total power consumption should not be higher than the desired power constraint. Third, as discussed in the next subsection, for the temperature of Core i to stay below the given threshold T_i, the DVFS levels of all the cores in the same CMP must be upper-bounded because the heat dissipation of any core in the same die may affect the temperature of Core i. Finally, two or more cores may need to have the same DVFS level due to application requirements or hardware limitations. Therefore, the constraints are modeled as:

$$\begin{aligned} & F_{min,j} \le f_j(k+1) \le F_{max,j} \quad (1 \le j \le N) \\ & cp(k) \le P_s \\ & \mathbf{B_i f(k+1)} < s_i \quad (1 \le i \le N) \\ & f_i(k+1) = f_j(k+1) \quad (1 \le i, j \le N) \end{aligned}$$

where $\mathbf{B_i}$ is the parameter vector for Core i's temperature constraint and s_i is a constant related to T_i. The derivations of $\mathbf{B_i}$ and s_i are introduced in Section 3.3.

Based on the above analysis, CMP power management has been modeled as a constrained MIMO optimal control problem. The controller must minimize the cost function (5) under the four sets of constraints. This constrained optimization problem can be easily transformed to a standard constrained least-squares problem [20]. The transformation is not shown due to space limitations. The controller uses a standard least-squares solver to solve the optimization problem on-line. In our prototype system, we implement the controller based on the `lsqlin` solver in Matlab. The computational complexity of `lsqlin` is polynomial in the number of cores and the control and prediction horizons. Please note that the computational complexity and runtime overhead of an MPC controller can be significantly reduced by using a multi-parametric approach proposed in a recent study [33]. This approach can divide the MPC control problem into an offline part and an online part. The offline part is complex but can be solved before the controller is implemented in the service processor firmware. At runtime, the controller only needs to solve the online part incrementally, which is a piecewise linear function. The multi-parametric approach makes it possible to use MPC for chip-level power control in practice.

3.3 Temperature Constraint

As introduced in Section 2, a temperature constraint is enforced on each core to prevent thermal emergency. In order for the constraint to be used by our MPC controller, we derive the temperature constraint by modeling the relationship between the temperature of a core and the DVFS level of the core. We first introduce some notation. In the k^{th} control period, The temperature sensor of Core i reports its reading as $t_i(k)$. The temperatures of all the cores in a CMP are represented as a vector: $\mathbf{t(k)} = [t_1(k) \dots t_N(k)]^T$. Recent studies [9, 1] present a thermal model between the temperature of a core and the power consumption of the core as:

$$\mathbf{t(k+1)} = \mathbf{A_t t(k)} + \mathbf{B_t p(k)} \quad (7)$$

To model the relationship between the temperatures and the DVFS levels. We substitute $\mathbf{p(k)}$ in (1) to (7). Our model is as follows:

$$\mathbf{t(k+1)} = \mathbf{A_t t(k)} + \mathbf{B_t(Af(k) + C)} \quad (8)$$

where $\mathbf{C} = [c_1 \dots c_N]^T$. However, since we can only measure the power consumption of the whole CMP processor instead of each individual core in system identification, c_i is unknown. Therefore, we have a difference model as:

$$\mathbf{\Delta t(k)} = \mathbf{A_t \Delta t(k-1)} + \mathbf{B \Delta f(k-1)} \quad (9)$$

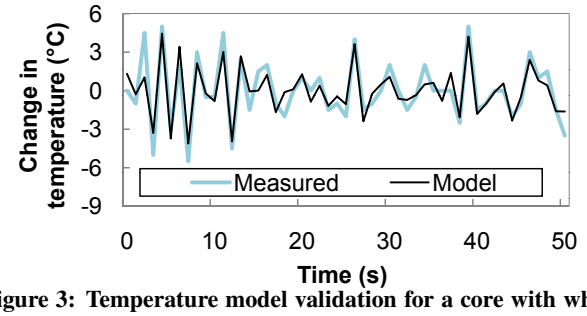


Figure 3: Temperature model validation for a core with white noise input.

where $\mathbf{\Delta t(k)} = \mathbf{t(k+1)} - \mathbf{t(k)}$, $\mathbf{\Delta f(k)} = \mathbf{f(k+1)} - \mathbf{f(k)}$, and $\mathbf{B} = \mathbf{B_t A}$.

Equation (9) models the relationship between the temperature change $\mathbf{\Delta t(k)}$ and the DVFS setting change $\mathbf{\Delta f(k)}$. $\mathbf{A_t}$ and $\mathbf{B}$ are two constant parameter matrices whose concrete values depend on the physical behavior of the chip, such as the heat sink and the time step (*i.e.*, the control period). In our experiments, we get the two matrices by conducting system identification. Specifically, we generate a pseudo random white noise signal as input $\mathbf{\Delta f(k)}$ to stimulate the open-loop system and measure the actual system output $\mathbf{\Delta t(k)}$. We then estimate $\mathbf{A_t}$ and $\mathbf{B}$ using the least square method. The concrete values of $\mathbf{A_t}$ and $\mathbf{B}$ used in our experiments are:

$$\mathbf{A_t} = \begin{bmatrix} -0.2259 & 0.1676 \\ 0.0896 & -0.2029 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 13.3368 & 6.8533 \\ 6.0877 & 12.7510 \end{bmatrix}.$$

We then validate the model (9) using a different random white noise signal. Figure 3 shows that the estimated outputs of the model are sufficiently close to the measured actual system outputs.

Our temperature constraint for Core i is $t_i(k+1) < T_i$. In every control period k, the controller needs to determine $\mathbf{f(k+1)}$ that can make the system converge to the power set point and meet the temperature constraint. Based on the temperature model (9), we can rewrite the temperature constraint as a function of $\mathbf{f(k+1)}$:

$$\mathbf{B_i f(k+1)} < s_i \quad (1 \le i \le N) \tag{10}$$

where $\mathbf{B_i}$ is the i^{th} row vector of the matrix $\mathbf{B}$, s_i is constant related to the temperature thresholds T_i $(1 \le i \le N)$.

Since our temperature constraint is built based on system identification, we use a safety margin $\delta = 2.822$ such that the predicted temperature is lower than the actual value most (*e.g.*, 98%) of the time in our system identification experiments. Therefore, the real constraint used in our MPC controller is $t_i(k+1) < T_i - \delta$.

3.4 Stability Analysis

A fundamental benefit of the control-theoretic approach is that it gives us theoretical confidence for system stability, even when the system power model (1) may change at runtime due to workload variations. We say that a system is *stable* if the total power $cp(k)$ converges to the desired set point P_s, that is, $\lim_{k\to\infty} cp\,(k) = P_s$. Our MPC controller solves a finite horizon optimal tracking problem. Based on optimal control theory [18], the control decision is a linear function of the current power value, the power set point of the CMP, and the previous decisions for per-core DVFS levels.

We now outline the general steps for analyzing the stability of a CMP when the actual system model is different from the *estimated* model used to design the MPC controller. First, given a specific system, we derive the control inputs $\mathbf{\Delta f(k)}$ that minimize the cost function based on the estimated system model (4) with estimated parameters $\mathbf{A}$. The control inputs represent the control decision based on the estimated system model. Second, we construct the *actual* system model by assuming the actual parameter $a_i' = g_i a_i$, where g_i represents the unknown system gain. The stability analysis of the actual system needs to consider a composite system consisting of the dynamics of the original system and the controller. Third, we derive the closed-loop system model by substituting the control inputs derived in the first step into the actual system model. Finally, we analyze the stability of the closed-loop system by computing all the poles of the closed-loop system. According to control theory, if all poles locate inside the unit circle in the complex space, the controlled system is stable. The allowed variation range of g_i can be established by computing the values of g_i that cause the poles to move across the unit circle.

We now apply the above steps to the CMP processor used in our testbed, which has two cores. In the first case, we assume that all the cores have a uniform workload variation, *i.e.*, $\mathbf{G} = g\mathbf{I}$. We derive the range of g as $0 < g \le 8.83$. That means a system controlled by the MPC controller designed based on the estimated model (4) can remain stable as long as the real system parameters are smaller than 8.83 times of the values used to design the controller. In the case that each core has a different workload variation, our second way of analyzing the system stability is to analyze one core at a time. Our results show that the ranges of g_i are $0 < g_1 \le 15.7$ and $0 < g_2 \le 17.6$. Other workload variation patterns can be analyzed in a similar way. A Matlab program is developed by us to perform the above stability analysis procedure automatically. In our stability analysis, we assume the constrained optimization problem is feasible, *i.e.*, there exists a set of DVFS levels within the acceptable ranges that can make the total power consumption converge to its set point. If the problem is infeasible, no control algorithm can guarantee the set point through core DVFS adaptation only. In that case, we need to integrate with other actuation mechanisms, which is our future work.

4. ONLINE MODEL ESTIMATOR

As discussed in Section 3, the MPC controller is designed based on an estimated power model that may be different from the actual system model, because the controller can be used on a different CMP and the workload may also change significantly at runtime. Although we have proven that the controller can remain stable when the variations are within a wide range, the MPC controller is designed to achieve the optimal control performance, such as short settling time and zero overshoot, when the model is accurate. With model variations, the MPC controller may have degraded control performance. Therefore, it is important to dynamically correct the system model based on measured power data. In this section, we describe the design of the online model estimator that can periodically learn from the measured data and then update the system model.

In this paper, we use a Recursive Least Square (RLS) estimator with directional forgetting [19] to estimate and update the parameter matrix $\mathbf{A}$ in the system model (3). To do that, we first transform the system model (1) to $cp(k) = \mathbf{A_e(k)f_e(k)}$, where $\mathbf{A_e(k)} = [\ a_1 \quad \dots \quad a_N \quad c\ ]$, $c = \sum_{i=1}^{N} c_i$, and $\mathbf{f_e(k)} = [\ \mathbf{f(k)} \quad 1\ ]^{\mathbf{T}}$. In each control period, the RLS estimator calculates the matrix $\mathbf{A_e}$ based on the following equation:

$$\mathbf{A_e(k)} = \mathbf{A_e(k-1)} + \frac{e(k)\mathbf{f_e^T(k)P(k-1)}}{\lambda + \mathbf{f_e^T(k)P(k-1)f_e(k)}} \tag{11}$$

where $e(k) = cp(k) - \mathbf{A_e(k)f_e(k)}$ is the estimation error, $\mathbf{P(k)}$ is the covariance matrix, and λ is the constant forgetting factor with $0 < \lambda \le 1$. A smaller λ allows the estimator to forget the history data faster. In our experiments, we use $\lambda = 0.8$. The following

Table 1: Core configuration parameters in simulations.

Processor		Memory hierarchy	
Processor core	Alpha 21264 like	L1 data-cache	64K, 2-way
Processor technology	65nm	L1 instruction-cache	64K, 2-way
Nominal frequency	5GHz	L2	Unified, 2M

iteration is invoked in every control period: 1) The RLS estimator records the frequency vector, $\mathbf{f}(\mathbf{k})$, and the total power consumption of the CMP, $cp(k)$. 2) The estimator calculates $\mathbf{f_e}(\mathbf{k})$, $\mathbf{A_e}(\mathbf{k})$ and then a_i $(1 \leq i \leq N)$. 3) The estimator updates $\mathbf{A}$ with a_i in the system model (3) of the MPC controller.

5. SYSTEM IMPLEMENTATION

In this section, we first introduce our physical testbed and benchmark, as well as the implementation details of each component in the control loop. We then introduce our simulation environment.

5.1 Testbed

Our testbed is an Intel Xeon X5365 Quad Core processor with 8MB on-die L2 cache and 1333 MHz FSB. The processor supports four DVFS levels: 3GHz 2.67GHz 2.33GHz, and 2GHz. Our experiments show that the processor has Core 0 and Core 1 in a group and Core 2 and Core 3 in the other group. We have to change the DVFS levels of the 2 cores in each group together in order to have a real impact on the processor power consumption. Therefore, we use this processor to emulate a dual-core processor that supports per-core DVFS. The operating system is Fedora Core 7 with Linux kernel 2.6.23.

We run SPEC CPU2006 suite (V1.0) as our workload both in our experiments on the testbed and in simulations. SPEC CPU2006 is configured with four user threads to run four copies of a benchmark on the four cores, respectively. Each performance measurement is the average of the four copies and recorded as performance ratio, *i.e.*, the relative speed of the processor to finish each benchmark (compared to a reference Sun UltraSparc II machine at 296 MHz). CPU2006 includes a collection of 29 benchmarks and is divided into CINT2006 and CFP2006, which consist of integer and floating-point benchmarks, respectively.

We now introduce the implementation details of each component in our power control loop.

Power Monitor. To measure the power consumption of the processor, we use the approach proposed in [12, 35]. An Agilent 34410A digital multimeter (DMM) is used together with a Fluke i410 current probe to measure the current running through the 12V power lines that powers the processor. The probe is clamped to the 12V lines and produces a voltage signal proportional to the current running through the lines with a coefficient of 1mv/A. The resultant voltage signal is then measured with the multimeter. The measured value is read by the server through a USB cable using a *USBTMC* device driver. The accuracy of the current probe is (3.5% of reading + 0.5A).

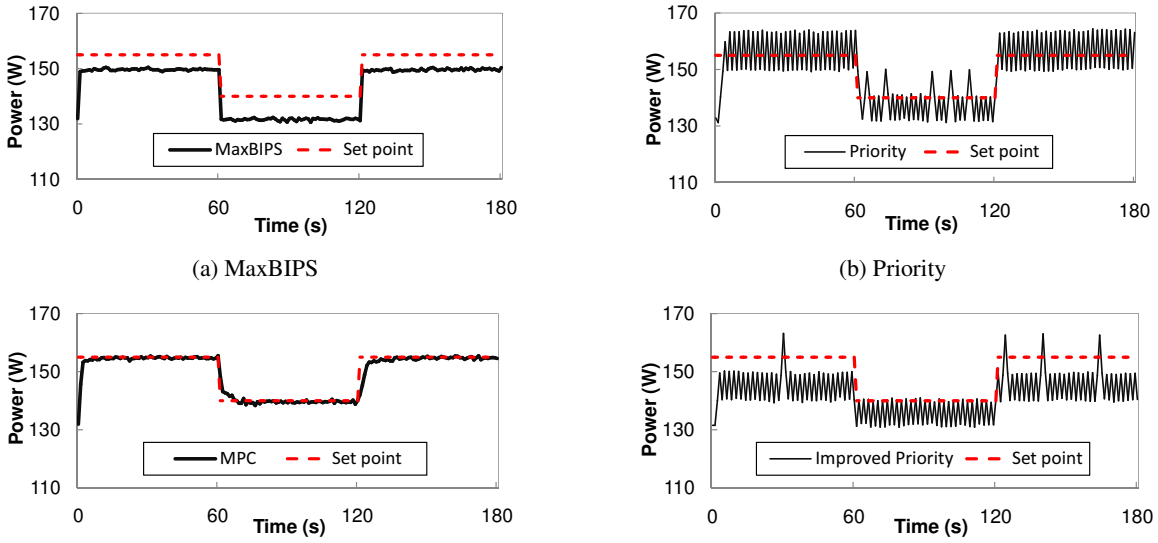
Temperature Sensor. We use the *coretemp* driver in Linux to read the temperature values reported by the thermal sensors in each core through the system file */sys/devices/platform/coretemp.[X]/temp1_input*, where [X] is the index of the core.

Controller. In our prototype system, we implement the controller as a C++ program running on a different processor to control the target processor, though the controller can be implemented in the service processor firmware in a real system. The controller periodically reads the power consumption from the power monitor and the temperatures from the sensors. It then executes the control algorithm presented in Section 3. As the outputs of the control algorithm, new DVFS levels are calculated and sent to the DVFS modulator to enforce the new levels in the next control period. The MPC controller parameters used in all experiments include the prediction horizon as 8 and the control horizon as 2. The time constant T_{ref}/T_s used in (6) is set to 2 to avoid overshoot while having a relatively short settling time. The control period T for the controller is set to 1 second because the timer resolution in Linux is $10ms$. Note that much shorter control periods could be used in a real firmware implementation.

CPU Frequency Modulator. We use Intel's SpeedStep technology to enforce the desired DVFS level. In Linux systems, to change the DVFS level, one needs to install the *cpufreq* package and then use root privilege to write the new DVFS level into the system file */sys/devices/system/cpu/cpu[X]/cpufreq/scaling_setspeed* where [X] is the index of the core. The processor used in our experiments supports only four discrete DVFS levels for each core. However, the new DVFS level periodically received from the MPC controller could be any value that is not exactly one of the four supported DVFS levels. Therefore, the modulator code must locally resolve the output value of the MPC controller to a series of supported DVFS levels to approximate the desired value. For example, to approximate 2.89GHz during a control period, the modulator would output the sequence, 2.67, 3, 3, 2.67, 3, 3, etc, on a smaller timescale. To do this, we implement the first-order delta-sigma modulator proposed in [17] to generate the sequence in each control period. Clearly, when the sequence has more numbers during a control period, the approximation will be better but the actuation overhead may become higher. In this paper, we choose to use 20 values to approximate the desired DVFS level, which leads to a subinterval of 50ms for a control period of 1s. As a result, the effect of actuation overhead on system performance is no more than 0.04% ($20\mu s$ of DVFS overhead [31] divided by $50ms$) even in the worst case when the DVFS level needs to be changed in every subinterval. This amount of overhead is acceptable to most computer systems.

5.2 Simulation Environment

To test our control algorithm with different CMP configurations (*e.g.*, number of cores in a CMP), we conduct simulations using the cycle-accurate SESC simulator [27] with modifications for per-core DVFS support. The cores are configured based on Alpha 21264 (EV6) [14], with the main parameters listed in Table 1. In our simulations, we configure the floor plan for 4, 8, and 16 cores with private L1 and L2 hierarchy with cores placed in the middle of the die. We integrate Wattch [3] with SESC to estimate the power change caused by DVFS scaling. We use per-core DVFS in our simulations and each core has 3 DVFS levels (3.88GHz, 4.5GHz, and 5GHz) configured based on the parameters of Alpha 21264 [14]. The control period of the MPC controller is set to 40ms and each control period has 20 subintervals for the first-order delta-sigma modulation, resulting in a subinterval of 2ms. During each DVFS transition time, we assume there is no instruction executed to keep synchronization [11]. The overhead of each DVFS scaling is set to $20\mu s$ based on a recent study [31] or 1% of the CPU time ($20\mu s$/2ms). Note that our MPC would have even higher performance improvement if the DVFS overhead can be reduced to nanoseconds in the near future [15]. We also use SPEC CPU2006 benchmark suite as our workload in simulations and run a copy of each benchmark on each core.



(a) MaxBIPS

(b) Priority

(c) MPC controller

(d) Improved Priority

Figure 4: A typical run of the MPC controller and the three baselines.

6. EXPERIMENTATION

We first introduce two state-of-the-art baselines. We then present our empirical results conducted on the physical testbed. Finally, we describe our simulation results for CMP processors configured with 4, 8 and 16 cores.

6.1 Baselines

Our first baseline, referred to as Priority, is a heuristic-based chip-level power controller proposed in a recent paper [11]. Priority represents a typical solution that is designed, without feedback control theory, to use per-core DVFS to control the power of a CMP chip. We compare our MPC controller against Priority to show that a well-designed ad hoc controller may still fail to have accurate power control and thus lead to degraded application performance. The control scheme of Priority is briefly summarized as follows. 1) Every core is assigned a priority. 2) In each control period, if the total power consumption of the CMP chip is lower than the set point, Priority chooses the core with the highest priority to increase its DVFS level by one. If the core is already running at the highest DVFS level, the core with the next highest priority will be tried. On the other hand, if the power of the chip is above the set point, Priority chooses a core (starting from the lowest priority one) to decrease its DVFS level by one. 3) Priority repeats step 2 until the system stops.

A fundamental difference between Priority and our MPC controller is that Priority simply raises or lowers the DVFS level of a selected core by one step, depending on whether the measured power is lower or higher than the power set point. In contrast, MPC computes a desired frequency level for each core based on well-established control theory and uses the DVFS modulator to approximate this output with a series of supported DVFS levels. In addition, MPC can prevent thermal emergency and dynamically adapt to workload variations.

The second baseline, referred to as MaxBIPS, is a recently proposed power management solution [11]. Given a power set point, MaxBIPS uses exhaustive search to find a combination of DVFS levels for all the cores, which is *predicted* to result in the best application performance while keeping the power of the chip below the set point. In order to do prediction online, MaxBIPS conducts offline experiments using a *typical workload* to build a static table, which includes all the possible DVFS combinations for all the cores, with the power consumption and performance (*i.e.*, BIPS values) of each combination measured from the offline experiments. While MaxBIPS can work effectively to find the best combination when the system is running the same or a similar workload with the one used to do offline experiments, MaxBIPS may fail to achieve good application performance and even may violate the power set point when the workload is totally different or vary significantly at runtime. In order to ensure that MaxBIPS does not violate the power set point at runtime, we use a power-intensive SPEC benchmark, *calculix*, to build the static table used by MaxBIPS in our experiments.

Although our MPC controller also has a prediction step, a fundamental advantage of MPC is that the prediction is continuously corrected based on system feedback, as discussed in Section 3.2. In addition, with the online estimator, the power model used by MPC is a dynamic one that can adapt to workload variations. In contrast, the prediction in MaxBIPS is only based on a static power model that is vulnerable to workload variations.

Please note that both Priority and MaxBIPS are demonstrated to have better application performance than power control algorithms that rely on chip-wide DVFS instead of per-core DVFS [11]. Therefore, by outperforming the two baselines, our MPC control algorithm also outperforms chip-wide DVFS solutions such as Intel's Foxton technology [21].

6.2 Empirical Results

In this section, we first compare MPC against the two baselines in terms of control accuracy and application performance. We then evaluate the temperature constraint. Finally, we examine the online model estimator.

6.2.1 Control Accuracy

In this experiment, we randomly select a SPEC benchmark, *gobmk*, to test MPC and the baselines in a scenario where the power budget of the CMP chip needs to be reduced from 155 W to 140 W at time 60s due to various reasons (*e.g.*, thermal emergency). The power budget is then raised back to 155 W at time 120s after the emergency is resolved.

Figure 4(a) shows the results of MaxBIPS. Given a power set point, MaxBIPS selects a DVFS combination from its static prediction table and sets the DVFS level of each core based on the

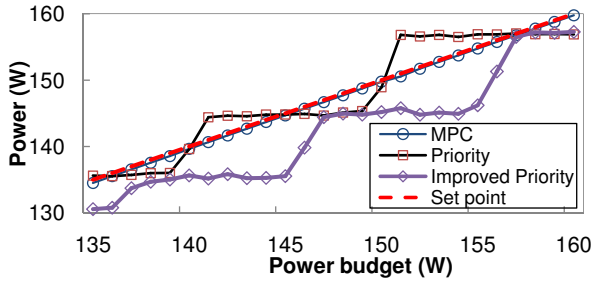


Figure 5: Comparison of steady state errors.

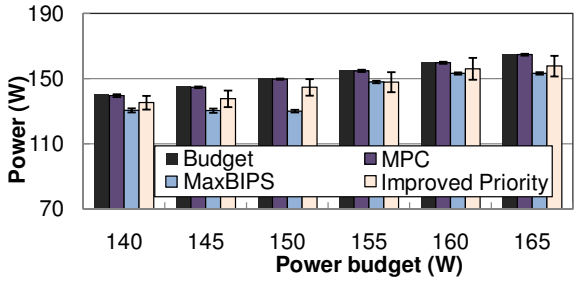


Figure 6: Control accuracy.

selected combination. Under MaxBIPS, the power of the CMP is much lower than the set point because of two reasons. First, MaxBIPS selects a DVFS combination to have a power consumption lower than the set point. Since each core of the CMP only has a few DVFS levels, a combination normally cannot lead to a power value that is exactly equal to the set point. With only two cores in the CMP, the gap can be large. Second, MaxBIPS makes decision based on the static table, which is generated offline based on a benchmark that is more power-intensive than the current workload. In the next subsection, we can see that the wasted power budget leads to degraded application performance.

Figure 4(b) shows the results of Priority. Priority starts with all cores throttled to the lowest frequency level. Since the power is lower than the set point at the beginning of the run, Priority responds by stepping up the DVFS level of one core at a time, until the power is higher than the set point at time 4s. Afterward, Priority oscillates between two DVFS levels on the core with a lower priority, because the set point is between the two power levels at two adjacent DVFS levels of a single core. As a result, the power never settles to the set point.

Figure 4(c) shows that MPC can precisely control the power of the CMP, with a standard deviation smaller than 1 W, by having a desired DVFS level from the MPC controller, and then using the DVFS modulator to generate a series of supported DVFS levels on a finer timescale to approximate the desired level. One may think that Priority could be improved by also using a series of DVFS levels for each core in a control period. However, Priority would still have the same steady-state error because, without a desired DVFS level based on control theory, Priority can only oscillate between two DVFS levels of a core.

Figure 5 shows the result of running MPC controller and Priority under a series of power set points from 135 W to 160 W. The steady-state power level of each run is the averaged power level in the steady state of the controller, which is calculated by eliminating the transient power values at the beginning of the run. The MPC controller is able to meet all the set points with a precision less than 1 W. However, since the CMP power under the control of Priority always oscillates between two levels, Priority shows steady-state errors that are often above the set point. For example, when the set point is 151 W, Priority has the maximum positive steady-state error as 5.8 W above the set point.

Since Priority has positive steady state errors, it may be undesirable to use Priority in a real system because a positive steady state error (*i.e.*, average power is above the set point) may violate the power budget. One may think that Priority could be easily modified to eliminate its positive steady-state errors by having a safety margin. To do this, we can get the steady-state error of each single run of Priority. To ensure that the safety margin is safe for all the benchmarks, we run every benchmark in SPEC CPU2006, and then get the maximum positive steady-state errors for each set point and then the maximum errors for all the set points from 140 W to 160 W. By doing that, we get a safety margin of 6.12 W. We then rerun the experiments for Priority with its power budget deducted this margin. This modified Priority policy is referred to as Improved Priority. Figure 4(d) shows that Improved Priority still exceeds the budget at times but the average power is always below the budget. Please note that Improved Priority is actually infeasible in practice because it is hard to have such a *priori* knowledge about the safety margin before spending a lot of time measuring this margin at runtime. However, we use Improved Priority as a baseline that can achieve the best possible performance in an ad hoc way and yet does not violate the power constraint.

6.2.2 *Application Performance*

In this experiment, we investigate the impact of chip-level power control on the performance of four randomly selected SPEC benchmarks. First, Figure 6 plots the average power with standard deviations for Improved Priority, MaxBIPS, and MPC. As discussed before, MPC can precisely achieve the desired power budgets while the other two waste the budgets. Improved Priority has a much bigger standard deviation compared with MPC and MaxBIPS because the power consumption under Improved Priority is always oscillating between two levels.

Figure 7 plots the SPEC benchmark performance (in terms of base rate, *i.e.*, the relative CPU speed compared to the reference machine used by SPEC) using different power budgets and benchmarks. MPC achieves better performance than Improved Priority and MaxBIPS in all the runs because MPC can precisely achieve the set-point power. The maximum performance improvement of MPC is 20.79% over MaxBIPS with *sjeng* at the set point of 150 W, and 18.77% over Improved Priority with *gobmk* at the set point of 155 W, respectively. The average improvement of MPC is 9.69% over MaxBIPS and 8.95% over Improved Priority.

Note again that it is actually infeasible in practice for Improved Priority to have such a tight safety margin resulted from extensive experiments in this paper. In a real system, Improved Priority is commonly configured with a large safety margin and thus would result in much worse performance. Although future CMPs may have significantly increased DVFS levels and cores, and thus can allow Improved Priority to achieve a control accuracy closer to that of MPC, Improved Priority will lead to a very long settling time with a large number of DVFS levels and cores since it only steps up/down one level of a core in each control period. The dependence of MaxBIPS on its static prediction table limits its applicability to CMPs with more than 8 cores, as discussed in Section 6.3.

6.2.3 *Temperature Constraint*

In this experiment, we evaluate the temperature constraint of the MPC power controller. In a real system, the maximum allowed temperature can be set based on the datasheet of the CMP chip. The temperature bound is initially $65^{\circ}C$ in our experiment. Between time 60s and 120s, we reduce the bound to $40^{\circ}C$ to emulate

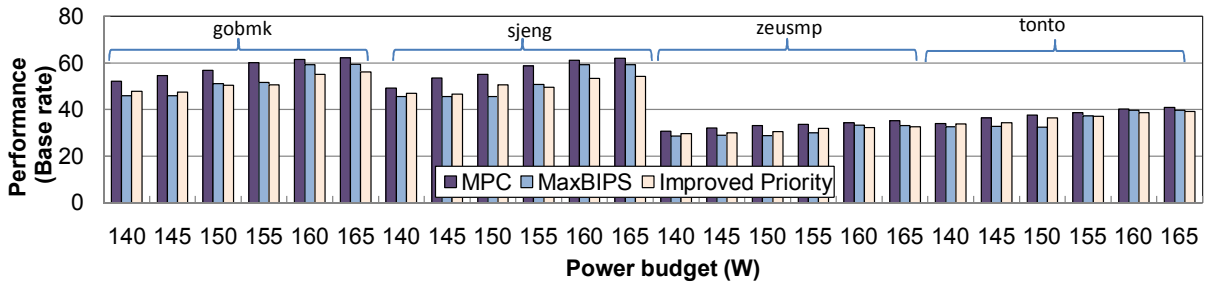
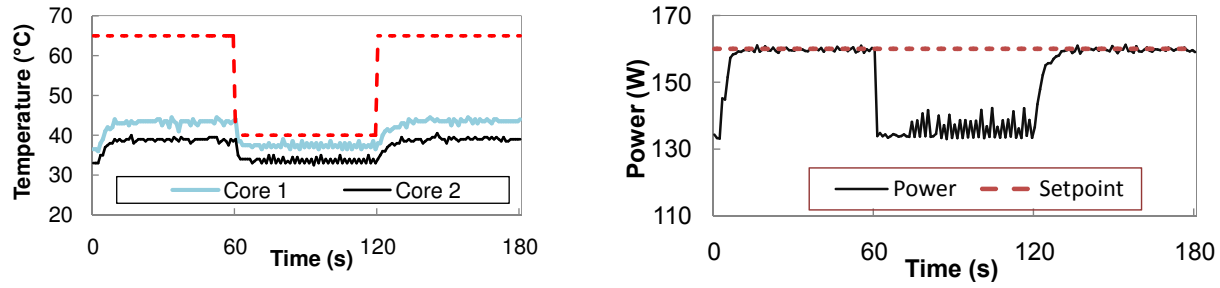


Figure 7: SPEC benchmark performance comparison between MPC and the baselines.



(a) Temperature (b) Power

Figure 8: Temperature constraints working effectively during an emulated thermal emergency.

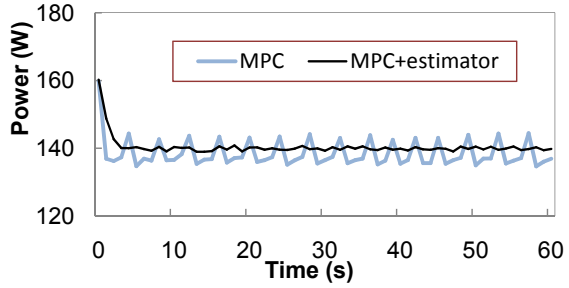


Figure 9: Effect of online model estimator when system model differs significantly from estimation.

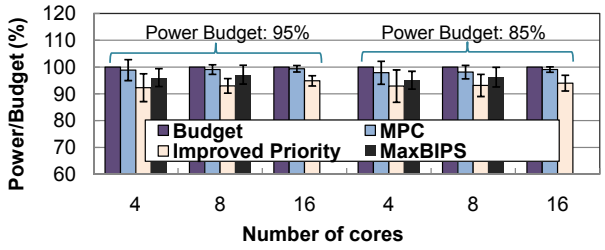


Figure 10: Control accuracy comparison under different number of cores in simulations.

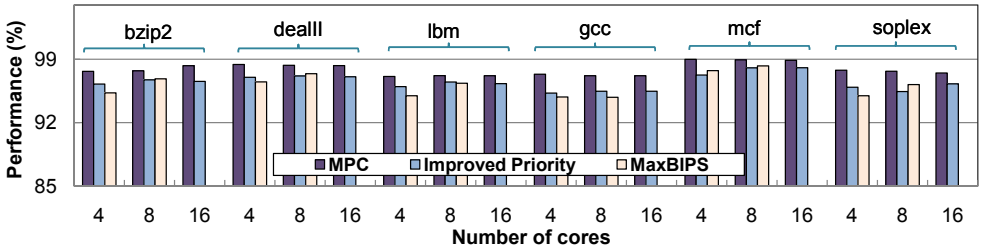
a thermal emergency event. Figure 8(a) shows that the measured temperatures of the cores are quickly constrained to stay below the lowered bound. Figure 8(b) shows that the temperature constraint works effectively to reduce the power of the CMP for the purpose of temperature reduction. There is a certain distance between the measured temperature and the bound because of the safety margin used in our temperature constraint, as discussed in Section 3.3.

6.2.4 Online Model Estimator

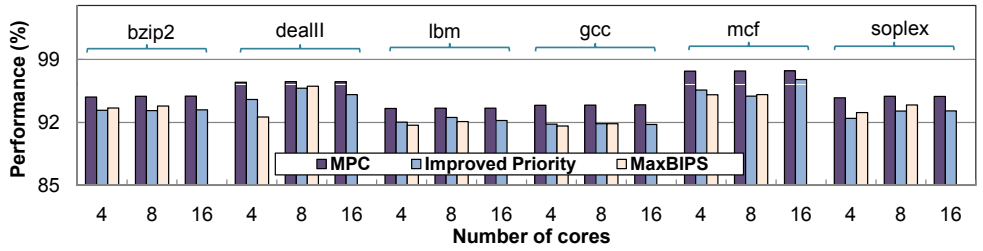
In this experiment, we examine the online model estimator. The estimator is important because MPC may have degraded control performance when the system model is inaccurate. To stress test the MPC controller, we intentionally choose the parameters, *i.e.*, $\mathbf{A}$, in the system model (2) to have values that are several times smaller than the values used in the previous experiments. Figure 9 shows that MPC (with the estimator disabled) now has much larger oscillations than before, though the average power consumption has been successfully controlled to the set point, *i.e.*, 140 W. Although MPC still outperforms Improved Priority in this case because it has a zero steady-state error, the instantaneous power spikes are undesirable. For example, at time 28s, the power consumption of the CMP is 4.16 W higher than the set point. The standard deviation of the measured power values is 3.24 W. We then enable the online model estimator and rerun the experiments with the same small parameter values in the MPC system model. Figure 9 shows that the estimator can quickly correct the system model at runtime in just several steps from the beginning. As a result, MPC performs as well as before, with a standard deviation of only 0.49 W. This experiment demonstrates that the online model estimator is important to the MPC power controller, especially when the workload is unknown or could vary significantly at runtime and when the controller is used to control different CMP processors with different processing capacities and within-die variations.

6.3 Simulation Results

In the previous subsection, we have tested our MPC controller on our testbed, where the CMP processor has only 2 scalable cores. In this subsection, we use simulations to demonstrate the efficacy of our MPC controller when the number of cores increases to 4, 8, and 16. We present the power reading from the simulator as a relative value. We first define a reference power value for each benchmark by measuring the maximum power consumptions of six randomly selected SPEC benchmarks: *bzip2*, *mcf*, *gcc*, *dealII*, *lbm*, and *soplex*. Every power value in this subsection is relative to the reference power value of that benchmark. In our simulations, we test MPC and the baselines using each benchmark under two power set points: 95% and 85%. Note that MaxBIPS is not implemented on the 16-core CMP because MaxBIPS needs a static prediction table, whose size increases exponentially as the number of cores



(a) Set point = 95%



(b) Set point = 85%

Figure 11: SPEC benchmark performance comparison under different number of cores in simulations.

increases. For the 16-core CMP, if each core has 3 DVFS levels, the prediction table for MaxBIPS will have 3^{16} entries, making it infeasible to be implemented.

Figure 10 plots the average power and standard deviation of the system controlled by MPC, Improved Priority, and MaxBIPS. With different number of cores, the power consumption of the CMP processor under MPC precisely converges to the desired power set points. Both Improved Priority and MaxBIPS waste the power budget because their average power consumptions always stay below the budget. The results confirm what we have observed on our testbed. Figure 11 plots the performance results of MPC, Improved Priority, and MaxBIPS under different number of cores, benchmarks, and power set points. MPC achieves the best benchmark performance in all runs. Our simulation results demonstrate that MPC can precisely control power to achieve better application performance for CMPs with different number of cores.

7. RELATED WORK

Power and temperature are two important design constraints for high-performance processors. Most of the prior work controls them separately. Isci et al. [11] propose both a heuristic-based closed-loop algorithm and a prediction based algorithm to control the power of a CMP to stay below a chip-level power budget based on per-core DVFS. Meng et al. [22] propose to use both per-core DVFS and cache resizing as actuators to control power. Teodorescu et al. [32] develop an optimization algorithm based on linear programming to provide power management for a CMP based on both DVFS and thread mapping. In contrast to their work that relies on heuristics, we present the first study to apply advanced optimal MIMO control theory to address this problem in a more rigorous way with theoretical guarantees. Some other prior work addresses the dynamic thermal management (DTM) problem [30, 2]. In contrast to those studies, our solution manages both power and temperature. Intel's Foxton technology [21]) and IBM's TPMD [7] have successfully controlled the power and temperature of a processor by using chip-wide DVFS. We use per-core DVFS to achieve better control accuracy and application performance.

Several research projects [23, 17, 29] have successfully applied control theory to explicitly control power or temperature of a single enterprise server. Kephart et al. have proposed a coordinated management strategy to achieve trade-offs between power and performance for a single server [13]. Kusic et al. present a power and performance management strategy based on lookahead control [16]. Some recent work has proposed heuristic-based control strategies at the server rack level [26, 6]. Control-theoretic solutions have also been designed to control rack-level power consumption for optimized system performance [34, 25]. Fan et al. [5] also investigate the aggregate power usage characteristics of a warehouse-sized data center. While those studies are at the server, server rack, and data center levels, we apply advanced MIMO control theory to control power for a CMP processor.

Some prior work has been proposed to use power as a tool for application-level performance requirements. For example, Horvath et al. [10] use dynamic voltage scaling (DVS) to control end-to-end delay in multi-tier web servers. Sharma et al. [28] effectively apply control theory to control application-level quality of service requirements. Ogras et al. [24] use DVFS to control the utilization of the inter-domain queues in future multiprocessor systems-on-chip. Chen et al. [4] also present a feedback controller to manage the response time in a server cluster. Although they all use control theory to manage power consumption, power is only used as a knob to control application-level performance. As a result, they do not provide any absolute guarantee to the power consumption of a processor. In this paper, we explicitly control the power consumption to adhere to a given constraint.

8. CONCLUSIONS

Existing work on power and thermal management heavily focuses on either open-loop search and optimization strategies based on static models, or heuristic-based closed-loop solutions that relies on oversimplified control algorithms without any theoretical guarantees. In this paper, we present the first study of applying advanced optimal MIMO control theory to chip-level power control based on per-core DVFS. Our algorithm can precisely control the power of a CMP chip to the desired set point, while maintaining the temperature of each core below a specified threshold. Furthermore, an online model estimator is designed to achieve analytical assurance of control accuracy and system stability, even in the

face of significant workload variations or unpredictable within-die core variations. Empirical results on a physical testbed show that our controller outperforms two state-of-the-art control algorithms by having better SPEC benchmark performance and more precise power control. In addition, extensive simulation results demonstrate the efficacy of our algorithm for various CMP configurations.

Acknowledgements

This work was supported in part by NSF under an NSF CAREER Award CNS-0845390 and a CSR grant CNS-0720663, and by Microsoft Research under a power-aware computing award in 2008. We thank the anonymous reviewers for their valuable comments.

9. REFERENCES

[1] D. Brooks, R. P. Dick, R. Joseph, and L. Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro*, 27(3), 2007.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA*, 2000.

[4] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. *ACM SIGMETRICS Performance Evaluation Review*, 33(1), 2005.

[5] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007.

[6] M. E. Femal and V. W. Freeh. Boosting data center performance through non-uniform power allocation. In *ICAC*, 2005.

[7] M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware. System power management support in the ibm power6 microprocessor. *IBM Journal of Research and Development*, 57(6), 2007.

[8] G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems, 3rd edition*. Addition-Wesley, 1997.

[9] Y. Han, I. Koren, and C. M. Krishna. TILTS: A fast architectural-level transient thermal simulation method. *Journal of Low Power Electronics*, 3(1), 2007.

[10] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multi-tier web servers with end-to-end delay control. *IEEE Transactions on Computers*, 56(4), 2007.

[11] 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 *MICRO*, 2006.

[12] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO*, 2003.

[13] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, F. Rawson, and C. Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *ICAC*, 2007.

[14] R. E. Kessler. The alpha 21264 microprocessor. *IEEE Micro*, 19(2), 1999.

[15] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA*, 2008.

[16] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *ICAC*, 2008.

[17] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *ICAC*, 2007.

[18] F. L. Lewis and V. L. Syrmos. *Optimal Control, Second Edition*. John Wiley & Sons, Inc., 1995.

[19] X. Liu, X. Zhu, P. Padala, Z. Wang, and S. Singhal. Optimal multivariate control for differentiated services on a shared hosting platform. In *CDC*, 2007.

[20] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[21] R. McGowen, C. A. Poirier, C. Bostak, J. Ignowski, M. Millican, W. H. Parks, and S. Naffziger. Power and temperature control on a 90-nm itanium family processor. *IEEE Journal of Solid-State Circuits*, 41(1), 2006.

[22] K. Meng, R. Joseph, R. P. Dick, and L. Shang. Multi-optimization power management for chip multiprocessors. In *PACT*, 2008.

[23] R. J. Minerick, V. W. Freeh, and P. M. Kogge. Dynamic power management using feedback. In *COLP*, 2002.

[24] U. Y. Ogras, R. Marculescu, and D. Marculescu. Variation-adaptive feedback control for networks-on-chip with multiple clock domains. In *DAC*, 2008.

[25] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No power struggles: Coordinated multi-level power management for the data center. In *ASPLOS*, 2008.

[26] P. Ranganathan, P. Leech, D. Irwin, and J. S. Chase. Ensemble-level power management for dense blade servers. In *ISCA*, 2006.

[27] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[28] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware QoS management in web servers. In *RTSS*, 2003.

[29] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *HPCA*, 2002.

[30] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. *SIGARCH Computer Architecture News*, 31(2), 2003.

[31] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1), 2004.

[32] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ISCA*, 2008.

[33] P. Tondel, T. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *CDC*, 2001.

[34] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *HPCA*, 2008.

[35] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *DAC*, 2006.