# Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management

—UNIV. OF VIRGINIA DEPT. OF COMPUTER SCIENCE TECH. REPORT CS-2001-27*—

Kevin Skadron†, Tarek Abdelzaher†, Mircea R. Stan‡
†Dept. of Computer Science, ‡Dept. of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA 22904
skadron@cs.virginia.edu, zaher@cs.virginia.edu, mircea@virginia.edu

## Abstract

*This paper proposes the use of formal feedback control theory as a way to implement adaptive techniques in the processor architecture. Dynamic thermal management (DTM) is used as a test vehicle, and variations of a PID controller (Proportional-Integral-Differential) are developed and tested for adaptive control of fetch "toggling." To accurately test the DTM mechanism being proposed, this paper also develops a thermal model based on lumped thermal resistances and thermal capacitances. This model is computationally efficient and tracks temperature at the granularity of individual functional blocks within the processor. Because localized heating occurs much faster than chip-wide heating, some parts of the processor are more likely to be "hot spots" than others.*

*Experiments using Wattch and the SPEC2000 benchmarks show that the thermal trigger threshold can be set within 0.2° of the maximum temperature and yet never enter thermal emergency. This cuts the performance loss of DTM by 65% compared to the previously described fetch toggling technique that uses a response of fixed magnitude.*

## 1. Introduction

Recent research in the computer architecture field has explored techniques for making the processor's response *adapt* to the current workload. Unfortunately, almost all this work has used engineering solutions whose effectiveness is only verified experimentally for a limited set of applications. This paper introduces the notion of using *formal feedback control theory* to precisely control the adaptivity and minimize performance losses. The use of control theory provides an established design methodology that yields several benefits: runtime adaptivity, very tight control over the value of interest, the ability to avoid large parameter-space searches, and adaptive controllers that are easy to design. In particular, we demonstrate its effectiveness for adaptive control of chip temperature—*dynamic thermal management* or DTM [2].

Researchers have recently expressed interest in managing heat dissipation in order to reduce the growing cost of the CPU's thermal package. Borkar [1] estimates that above 35–40 Watts (W), additional power dissipation increases the total cost per chip by more than $1/W. Without some sort of dynamic technique for managing temperature, thermal packages must typically be designed for peak power in order to ensure safe operation—even though peak power may rarely be observed. Unfortunately, peak power dissipation may soon be as high as 130 W [18], making thermal packages expensive. As the sophistication of high-performance processors grows, much of the added complexity and transistors are dedicated to extracting further ILP. Yet in some cases, these features are only intermittently active during the program. This only widens the gap between peak and average power dissipation.

The gap between peak and average power dissipation suggests that, instead of designing the thermal package for sustained peak power dissipation, designers should instead be able to target the thermal package for some lower temperature and hence lower cost. In those cases when power dissipation does approach peak levels and temperature approaches the limits of the thermal package, various techniques for DTM can be engaged to force a reduction in power dissipation and hence avoid thermal emergency. Because these techniques typically entail slowing down the processor in some way, there may be performance penalties whenever DTM is triggered. The decision to deploy DTM therefore requires a tradeoff between savings in ther-

*This technical report (Nov. 2001) is an extended version of a paper with the same title and authors, appearing in *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, Cambridge, Mass., USA, Feb. 2002. ©2001 by Skadron, Abdelzaher, and Stan.

mal packaging costs and consequent losses in performance. Yet speed remains critically important for high-performance computing, and many embedded-systems applications as well. The challenge of DTM is to adapt the thermal management to the application's needs in order to minimize the slowdown and extract the optimum combination of temperature and performance. For any chosen thermal package, better performance can be accomplished either by developing new mechanisms for implementing DTM, or by more precisely engaging and disengaging the DTM mechanisms to minimize the performance loss. Precision is what feedback control provides.

To better evaluate our techniques, this work also develops a more detailed model of thermal effects. Prior work by Brooks and Martonosi [2] on architecture-level DTM used a moving average of chip-wide power dissipation as a proxy for temperature. We instead apply a model from the thermal-packaging literature that uses thermal resistances (R) and thermal capacitances (C) to directly model temperature. This model is also used in the TEMPEST work by Dhodapkar *et al.* [6], but only for average, chip-wide temperatures to capture packaging effects. Our work uses the RC model to derive a computationally efficient thermal model for temperature effects in individual structures, and hence lets us model "hot spots" and appropriate DTM responses. This is important, because localized heating occurs much faster than does chip-wide heating, and so thermal emergencies may occur at individual hot spots well before the chip as a whole shows any signs of entering thermal emergency. DTM mechanisms must take this fact into account. Other work—the most recent by Yuan and Hong [24]—has also modeled non-uniform heating effects, but not in conjunction with architecture-level modeling. We are unaware of any other work that combines localized thermal modeling with a microarchitecture-level model.

Overall, this paper introduces the use of thermal-RC modeling for architecture-level thermal effects, and introduces the use of control theory to guide architecture-level thermal management. We find that the control-theoretic DTM techniques (CT-DTM) that are based on the commonly used PID industrial controller (Proportional-Integral-Differential) substantially outperform the non-control-theoretic (non-CT) techniques. Indeed, the PI and PID controllers are able to respond so quickly that thermal response can be delayed until the chip temperature comes within $0.2^\circ$ of the desired maximum temperature. This responsiveness provides savings of 65% in performance loss compared to the best DTM technique described by Brooks and Martonosi while never causing thermal emergencies. The results presented here suggest that feedback control would be beneficial in a range of other adaptive architecture techniques as well.

The rest of this paper is organized as follows. The next three sections provide some necessary background on DTM, controller design, and thermal modeling. Then Section 5 describes our specific simulation and thermal-modeling techniques, and Section 6 compares our thermal model to the prior technique of using only a boxcar average of power measurements. Section 7 evaluates the effectiveness of our control-theoretic DTM techniques, and Section 8 concludes the paper.

## 2. Dynamic Thermal Management

### 2.1. Non-Control-Theoretic DTM Techniques

The DTM work by Brooks and Martonosi [2] proposed three microarchitectural and two scaling techniques for controlling temperature. All are engaged at some trigger level close enough to the emergency threshold that a trigger indicates emergency is imminent, and far enough from the threshold to give the DTM mechanism enough time to successfully reduce temperature before an emergency occurs. For their work, Brooks and Martonosi used power dissipation as a proxy for temperature. They chose an emergency threshold of 25 W and a trigger threshold of 24 W, both based on the boxcar average of per-cycle power dissipation over the last 10 K cycles. Unfortunately, chip-wide power is a poor proxy for measuring temperature. Chip-wide measurements do not account for the fact that localized heating occurs much faster—typically orders of magnitude faster—than chip-wide heating. And while power and temperature are clearly related, heating is an exponential effect (like an electrical RC circuit) that a boxcar average cannot capture. This makes it difficult to choose an average power level that indicates thermal stress. Instead, Section 4 of this paper describes a way to model temperature directly and permit the modeling of hot spots at various granularities, and Section 6 compares the two types of modeling.

For the DTM mechanisms studied here, we set the goal that the techniques we study should *never* allow the chip to enter thermal emergency. The most effective microarchitectural mechanism that Brooks and Martonosi explored consists of *toggling*: every $N$ cycles, instruction fetch is disabled. This reduces the average number of instructions in the pipeline and reduces the rate of accesses to the major structures on the chip. They explore values for $N$ of 1 (instruction fetch stops completely until the DTM mechanism is disengaged) and 2 (instruction fetch occurs every other cycle until disengaged). "Toggle1" is able to eliminate emergencies, because it stops fetching entirely; "toggle2" is not. For the control-theoretic DTM mechanisms, we consider more fine-grained variations of the toggling rate. Compared to these responses of fixed strength, the control-theoretic approach lets us use mild reductions in fetch toggling when the thermal stress is mild, but more aggressive toggling as the thermal stress becomes more severe.

Brooks and Martonosi also explored two other microarchitectural techniques, *throttling* [17] and *speculation control* [14]. In throttling, instruction fetch is performed every cycle, but the number of instructions fetched is reduced from the normal operating bandwidth. This has the problem that the number of accesses to many structures, especially the branch predictor and I-cache, are not reduced, and this technique often cannot prevent certain hot spots. These problems point out the importance of monitoring temperature on a per-structure basis; the chip-wide average of power dissipation will not capture these effects. With speculation control, whenever more than $M$ unresolved branches are present in the pipeline, no further instructions are fetched until the number of unresolved branches falls below $M$. Unfortunately, this technique is ineffective for programs with excellent branch prediction or periods of excellent branch prediction.

The scaling techniques consist of either scaling just the clock frequency or scaling both the frequency and the operating voltage. The advantages of the microarchitectural techniques are that they can be engaged and disengaged quickly, and that in some cases, the program's ILP characteristics permit the DTM mechanism to work well without penalizing performance. The scaling mechanisms, on the other hand, necessarily entail some loss in performance because the entire processor operates more slowly. In addition, the processor must stall for as much as 10–20 millisec. while the clock re-synchronizes. The overhead of invoking a scaling policy also means that it must be left in place for a significant *policy delay* in order to ensure that the temperature has been sufficiently reduced so that scaling will not be re-initiated in the near future. But as mentioned, during this period of time the slower clock rate entails a loss in performance, even after temperature decreases. For these reasons, Brooks and Martonosi found that the microarchitectural techniques had less impact on performance. Of course, if a microarchitectural technique fails to stem rising temperatures, scaling can also be performed as a backup policy.

Based on the inferior performance of fetch throttling, speculation control, and the scaling techniques, we do not consider them further, even though a realistic implementation might employ a hierarchy of DTM techniques: for example, a low-cost mechanism like toggling might be used with a high trigger threshold. Only when temperature gets truly close to emergency would auxiliary mechanisms like voltage/frequency scaling be employed.

The prior DTM work also explored two ways in which DTM may be triggered. In the first case, a thermal trigger engages an interrupt, which invokes a handler that then sets the DTM policy. After a specified time, another interrupt checks the thermal condition and if there is no longer a triggering condition, the DTM policy is turned off. The use of interrupts, however, introduces some delay (*e.g.*, 250 cycles) for each event. This incurs some small but unavoidable loss in performance even for an ideal DTM policy. Brooks and Martonosi also postulate the existence of a microarchitectural mechanism by which a thermal trigger immediately engages the DTM policy, with no delay. The only performance loss is then from the interaction of the DTM policy and the program's natural instruction throughput. This can be implemented by having each temperature sensor assert a signal indicating that it has been triggered, a feature we assume in our model.

As mentioned, a policy delay is also necessary, even for microarchitectural mechanisms. Once a DTM policy is invoked, it must stay active for a long enough period of time to sufficiently reduce the temperature. The delay must be set empirically: too short a policy, and the system will stay at or near trigger; too long a policy, and the system will incur an unnecessary loss in performance. The trigger level is also set empirically. Too close to the emergency threshold, and DTM will not be engaged soon enough to prevent emergency. Too far, and DTM is engaged unnecessarily.

The use of better thermal modeling and the application of control theory to the design of DTM mechanisms solves these problems. Simulating temperature directly provides more realistic indicators of how fast temperature changes, how different structures move in and out of thermal stress, and a more faithful replica of what actual temperature sensors in a real chip would observe. Section 4 discusses issues related to thermal modeling.

## 2.2. Control-Theoretic DTM Techniques

Clearly, any one DTM policy with fixed response (*e.g.* toggle1) will be sub-optimal. To guarantee elimination of thermal emergencies, a policy must aggressively reduce activity whenever triggered. But in many cases, such aggressive response is not needed. For example, mild thermal stress need not employ the very aggressive toggle1 policy, but of the mechanisms described so far, only toggle1 can completely avoid emergencies. An adaptive policy, on the other hand, can adjust the degree of response to the severity of the thermal situation. Yet designing adaptive mechanisms by hand is difficult, because it typically requires large parameter-space searches.

In this paper, we propose to use formal feedback control theory to manage the DTM and avoid the difficulties associated with ad-hoc techniques. A control-theoretic approach provides a number of advantages. It permits the DTM policy to adapt its response in proportion to the thermal emergency and also permits it to take account of prior history and the rate of change in temperature. The degree of response to each of these factors is controlled by a formal methodology, avoiding the need for ad-hoc methods. The toggling mechanisms provide an excellent basis for using a controller. Instead of a fixed toggling policy, the rate at which fetching

is permitted is set by the controller and adjusted every $n$ cycles, where $n$ is the sampling rate—every 1000 cycles in our experiments. As the control output ranges from 0% to 100%, the fetch toggling ranges proportionally from all the way off to all the way on (toggle1). An output of 50%, for example, corresponds to toggle2.

The use of control theory also provides other benefits. Although beyond the scope of this work, controllers can be designed with guaranteed settling times (*i.e.*, how quickly the temperature settles back to a desired, safe level), and an analysis of the maximum overshoot can be used to choose a setpoint that, in conjunction with the appropriate controller, is as high as possible without risking an actual emergency.

For this paper, we focus on integrating DTM with more accurate thermal modeling and explore only variations of the PID controller, commonly used in industry. This simple controller is sufficient to demonstrate our techniques and to provide substantial gains in performance. The next section presents some basic precepts of controller design and uses them to derive the adaptive controllers we test in Section 7.

## 3. Basic Control-Theoretic Techniques

Feedback control theory has been very successful at developing analytic frameworks and algorithms for robust performance control in physical systems. Using a mathematical model of the controlled process (such as the thermal dynamics relating power dissipation to temperature output of various microprocessor structures), it produces controllers which help achieve the desired output. Control-theoretic approaches have been applied to a variety of aspects of computer systems design to achieve adaptive response, *e.g.*, CPU scheduling [13, 21] and Internet congestion control [9]. Closer to the topic of this proposal, Dragone *et al.* have described a feedback technique for voltage scaling [8], and Wong *et al.* [23] have described a feedback circuit for canceling leakage currents.

Temperature control of a physical device such as a microprocessor is a natural application of control theory. A commonly used industrial controller is the PID (Proportional-Integral-Differential) controller. It has been proven very successful and robust even in controlling highly non-linear and poorly modeled systems. This subsection explains our derivation of a PID controller for DTM; those familiar with such derivations may wish to skip to the next section.

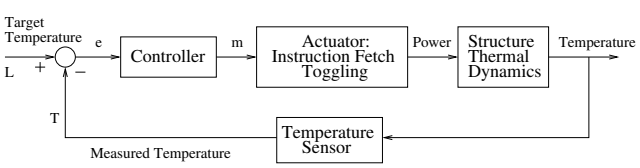


**Figure 1. Control Loop**

### 3.1. The Feedback Control Loop

Figure 1 presents the control loop being modeled. Let $E$ be the thermal emergency level, such that heating of some chip structure beyond $E$ poses a threat to that structure. We set a target temperature $L < E$ for the structure, that is very close to, but does not exceed, the emergency limit. A feedback control loop works by sampling the actual temperature, $T$, at discrete time instances (*e.g.*, once every 1000 cycles) and comparing it to the target $L$. The difference, $e = L - T$ is the current error in achieving the performance target. If the error is negative, the system is overheated, hence power dissipation should be reduced during the next sampling interval. If the error is positive, the system can relax any power-reduction techniques that are currently engaged.

### 3.2. Controller Derivation

The basic PID controller responds to error by a correction that is an algebraic superposition of three actions (or forces). The first is the proportional action. It changes power in proportion to the value of the error and in the direction that reduces the error. The second is the integral action. It adjusts power incrementally, in proportion to the time integral of previous errors. This action tends to accumulate a slowly-changing bias that becomes constant when the error becomes zero, hence maintaining the system at the zero-error state. The last component is the derivative action. It adjusts power in proportion to the rate of change of error in the direction that reduces the rate of change, damping the response to avoid overshoot. At any time $t$, controller output, $m(t)$, is the weighted sum of the above three terms:

$$m(t) = K_C \left( e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt} \right) \qquad (1)$$

where $K_C$, $K_I$, and $K_D$ are constants that need to be tuned according to stability analysis to ensure that the system will not oscillate. The selection of weights in the above equation gives a rich design space for controller functions. Finding the best weights requires a model of the controlled system. This model depends on whether the system is continuous or discrete. While in principle the system is discrete by virtue of sampling, our sampling period is much smaller than the time-scale of the thermal dynamics for the controlled chip structures. Hence, for all practical purposes, the system behaves in a continuous manner. Continuous controller design is typically performed in the Laplace transform domain. The Laplace transform of the controller transfer function is:

$$m(s) = K_C \left( 1 + \frac{K_I}{s} + K_D s \right) e(s) \qquad (2)$$

where $s$ is the Laplace transform operator. The thermal dynamics of the controlled structure are represented by its

| Thermal quantity | unit | Electrical quantity | unit |
|---|---|---|---|
| $P$, Heat flow, power | $W$ | $I$, Current flow | $A$ |
| $T$, Temperature difference | $K$ | $V$, Voltage | $V$ |
| $R_{th}$, Thermal resistance | $K/W$ | $R$, Electrical resistance | $\Omega$ |
| $C_{th}$, Thermal mass, capacitance | $J/K$ | $C$, Electrical capacitance | $F$ |
| $\tau_{th} = R_{th} \cdot C_{th}$, Thermal RC constant | $s$ | $\tau = R \cdot C$, Electrical RC constant | $s$ |

**Table 1. Equivalence between thermal and electrical quantities**

thermal time-constant, $\tau$, and a steady state gain, $K_p$, representing the steady state ratio of the change in output temperature to the change in input power that produces it—the thermal R in this application. In addition, sampling introduces an effective delay, $T_d$, in the loop, equal to half the sampling period. Hence, in the Laplace domain, the controlled system model of temperature is:

$$G(s) = \frac{K_p e^{-T_d s}}{1 + \tau s} \tag{3}$$

where $K_p = R_{block}$. Note from Figure 1 that any deviation, $e$, traveling through the feed-forward path of the loop will be transformed by the combined function $G(s)m(s)$, which is the product of all components on that path. If the actuator and the temperature sensor introduce some gain into the system, $K_a$ and $K_s$, then the total transformation is $G(s)K_s m(s)K_a$.

In the interests of space, we omit the rest of the derivation and stability analysis, and merely observe that this yields two equations in four unknowns: $w$, $K_C$, $K_I$, and $K_D$. To solve this set, the simplest approach is to set $K_I = K_D = 0$. This solution yields a proportional controller which has the benefit of computational efficiency. Alternatively, it is possible to set only one of $K_I$ and $K_D$ to zero, producing a PI or PD controller respectively. This makes it possible to introduce an additional design constraint via a phase constant, $\omega$. For the PI controller, we set $\omega$ to $\pi/6$; for PID, 0; and for PD, $-\pi/6$. Finally, in more complex systems, two additional design constraints may be accommodated which uniquely determine both $K_I$ and $K_D$. In our PID controller for fetch toggling, we set $K_I = 1/4K_D$. All the preceding values are common values that are known to work well in practice. They were successful with no tuning, making the derivation of effective controllers extremely easy. We did test other values for these parameters, but found that they were no better than the conventional values. This also illustrates the robustness of control-theoretic techniques: the performance of feedback-control systems remains largely unaffected even when the controlled system has not been accurately modeled in the analysis, or when external factors that have not been accounted for at design time interfere with its operation.

The final values for the PID controller were were $K_s = 1$ (idealized temperature sensor), $K_a = 3$ (actuator gain), $K_C = 4.42$, $K_I = 2.36 \times 10^6$ and $K_D = 1.06 \times 10^{-7}$ for a sampling frequency of 1000 clock cycles or 667 nanosec. The time constant of the system is the $RC$ time constant from the thermal model; we used the longest time constant of the various blocks under study.

### 3.3. Actuator Saturation Effects

In designing the controller, it is important to consider saturation effects. In general, actuator saturation may have a negative effect in the presence of integral action, which is commonly known as *integral windup* and is a common problem in the design of industrial controllers. In our example, if the application produces only a small power dissipation, it may be impossible to reach the target chip temperature even when the processor operates at full speed. In this case, a positive error persists causing an arbitrarily high increase in the output of the integral action. This increase is meaningless to the actuator, which in this case becomes saturated. Eventually, if power dissipation does increase and temperature overshoots the set point, it will take the integral output a long time to "unwind", *i.e.*, return to a reading that is within the actuator's input range. During that time the actuator continues to be saturated, and the processor continues to operate at full speed, possibly entering a thermal emergency. Integral windup can be easily avoided by freezing the integrator when controller output saturates the actuator. Hence, once the error changes sign due to an overshoot, this permits controller output to immediately decrease below saturation (*i.e.*, *within* the actuator range), causing it to slow down pipeline execution and reduce power dissipation immediately, as needed. As mentioned above, we implemented this mechanism in our PI and PID controllers by preventing the integral from taking on a negative value.

## 4. Thermal Modeling

### 4.1. Using an Equivalent RC Circuit to Model Temperature.

For an integrated circuit at the die level, heat *conduction* is the dominant mechanism that determines the temperature. There exists a well-known duality [11] between heat transfer and electrical phenomena as summarized in Table 1. Any heat flow can be described as a "current", and the passing of this heat flow through a thermal "resistance" leads to a temperature difference equivalent to a "voltage".

This is really Ohm's law for thermal phenomena. Thermal resistances are enough for describing steady-state behavior, but dynamic behavior is important for DTM, and this requires thermal "capacitances" as well. Thermal capacitances imply that even if the power flow changes instantaneously, there is a delay before the temperature changes and reaches steady state. The thermal resistances and capacitances together lead to exponential rise and fall times characterized by thermal RC time constants similar to the electrical RC constants.

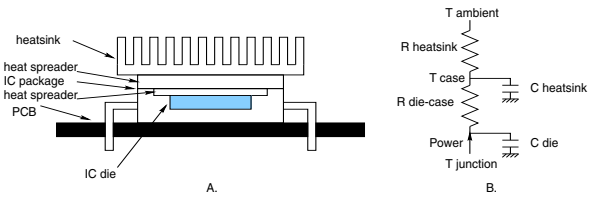


**Figure 2. IC package with heatsink.**
A. physical structure, B. simple lumped thermal model.

Figure 2A shows a typical IC package with a heatsink and an equivalent simplified thermal model. The IC die consumes power when active and this power needs to be dissipated as heat. In order to simplify the analysis it is general practice to ignore all large thermal resistances and only consider a simplified model as in Figure 2B with very little loss of accuracy. Large thermal resistors in parallel with smaller ones can safely be ignored because they cannot transfer enough heat to further reduce the temperature determined by the smaller values. Thermal capacitances are necessary for dynamic behavior and these have been also represented. To see the usefulness of the model, assume an IC that dissipates 25 W, die-to-case thermal resistance of 1 K/W, and heatsink resistance (conduction plus convection to ambient) of another 1 K/W. For an ambient temperature of 27 degrees Celsius, we can predict that the steady state average die temperature will be $25\text{W} \cdot 2\text{K/W} + 300\text{K} = 350\text{K}$ or $77^\circ$ Celsius. Furthermore, if dynamic behavior is desired, the thermal capacitances can be used to derive such information. For example, assuming a heatsink thermal capacitance of 60 J/K and a much smaller die thermal capacitance, we can determine that the time constants involved in how fast the circuit heats up when powered on, or cools down when powered off, are on the order of $60\text{J/K} \cdot 1\text{K/W} = 60\text{s} = 1$ minute.

## 4.2. Modeling Localized Heating

Large ICs have a heterogeneous structure with many different areas on the die working at different rates, which implies that power is not dissipated uniformly on the chip. This *spatial non-uniformity* is complemented by a *temporal non-uniformity* in power density as many structures on the die go from idle mode to full active mode and vice-versa at different times. Recent emphasis on low-power design techniques such as power modes, clock gating, etc. are exacerbating the spatial and temporal non-uniformity for on-chip power density. As a result of this non-uniformity, the chip will exhibit so-called "hot spots". These hot spots have a spatial distribution as a result of the non-zero thermal resistivity of silicon and also a temporal distribution due to changing program behavior and the time constants implied by the thermal mass (capacitance).

A model for localized heating allows simulation of the actual physical temperature at different locations on the chip. In order to derive a lumped circuit model we first need to decide on the level of granularity for the lumped elements. We decided to use a natural partitioning where functional blocks on the chip are the nodes in the lumped circuit model. This has the advantage that there is a one-to-one correspondence between the model in the architectural simulator and the thermal circuit, which leads to a good coupling between the two. We also currently make the simplifying assumption that it is feasible to have thermal sensors associated with each functional block. This is unrealistic, since the number of sensors is likely to be limited, and they may not be co-located with the most likely hot spots. Developing a model for temperature sensor behavior (as distinct from true physical temperature) is an important area for future work.

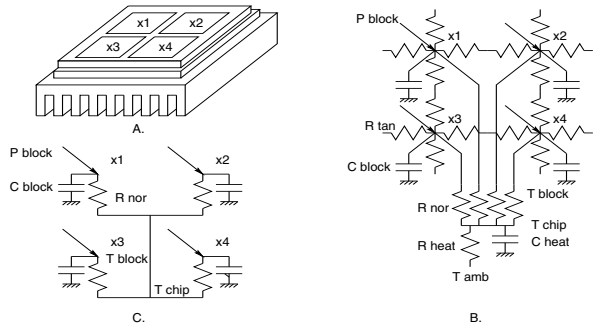


**Figure 3. IC die with 4 functional blocks.**
A. physical structure with die, heat spreader and heatsink, B. detailed lumped thermal model with tangential and normal block thermal resistances and block capacitances, heatsink resistance and capacitance, connected to ambient temperature, C. simplified lumped model with only block normal resistances and capacitances connected directly to the chip temperature.

We model the thermal circuit as in Figure 3B. Each block dissipates a (different) power $P_{block}$ and as a result will tend to have a (different) temperature $T_{block}$. Nearest neighbor blocks are connected together through "tangential" thermal resistances $R_{tan}$ and each block is also connected through "normal" thermal resistances $R_{norm}$ to the heatsink through

the heat spreader. Each block also has a (different) thermal capacitance $R_{block}$. While silicon is not a very good thermal conductor, hence the possibility of hot spots and the necessity for finer grain modeling, the heat spreader and heatsink are designed to be very good conductors, and can be lumped into a single thermal resistance $R_{heat}$ and capacitance $C_{heat}$.

### 4.3. Component Values for the Lumped Circuit

When determining the actual values for the lumped circuit, we observe two important aspects which allow us to further simplify the thermal model as in Figure 3C:

- The values of $R_{tan}$ are much larger than $R_{norm}$. This means that the spatial distribution of the hot spots is dominated by $R_{norm}$ and the tangential values can be ignored for a first-order analysis.
- The RC time constants for the heatsink are orders of magnitude larger than for the individual blocks. This means that the temporal distribution of hot spots for short time periods is dominated by the block values and the heatsink can be considered at a constant temperature over short periods of time.

Here is how we determine the component values for the lumped circuit. For the individual blocks we first consider the material properties of silicon. Both the thermal capacitance and thermal resistance for silicon are variable with temperature, but the variation is small. From published data [12] for silicon, at target temperatures, we derived an approximate thermal capacitance $c = 10^6 J/m^3 K$, and an approximate thermal resistivity $\rho = 10^{-2} mK/W$. With these values we can easily derive the block thermal capacitances $C_{block}$ as $c \cdot A \cdot t$, and the block normal thermal resistances $R_{norm}$ as $\rho \cdot t/A$ where $A$ is the block area and $t$ the wafer thickness. For a wafer thickness of 0.1 mm, $C_{block}$ becomes $100 \cdot A$ and $R_{norm}$ becomes $10^{-6}/A$, with the block area $A$ expressed in square meters.

Calculating $R_{tan}$ is slightly more complex. Assuming that heat is flowing uniformly and circularly from the center of the block, and writing thermal Ohm's law for an infinitesimally narrow circle of width $dx$ at distance $x$ from the outside (i.e. at radius $L - x$), we get:

$$T(x + dx) - T(x) = P \frac{\rho dx}{2\pi(L - x)t} \pi (L - x)^2 \qquad (4)$$

We omit the details, but this can be computed for a silicon wafer of thickness 0.1mm to derive $R_{tan} = 100K/W$. Since this is orders of magnitude larger than on chip $R_{norm}$ values we conclude that we can safely ignore $R_{tan}$.

The time constants associated with a block on the chip will be on the order of $10^{-4}$ seconds (see next section), which is much smaller than the time constants for the heatsink. We conclude that we can safely ignore the dynamic aspects of the heatsink for short time periods and consider the average heatsink temperature as a constant. When we compared observed temperatures and thermal-emergency events between our localized model and current chip-wide thermal modeling techniques, we found that almost all thermal-emergency events detected with the localized model failed to be observed by the chip-wide model. The reason for this is that localized heating is much faster than chip-wide heating.

We used the resulting model for computing actual temperatures at various locations on the chip, and use the same model as a proxy for what temperature sensors would observe as they drive our PID-DTM control system. The next section describes our simulation environment and how we derived the specific thermal R and C values used in the simulations. Section 6 briefly compares results with our thermal model to those obtained with just a boxcar average of recent power measurements.

The only other architecture-level modeling of which we are aware is the TEMPEST work by Dhodapkar *et al.* [6], which describes a multi-mode simulation package that models performance, power, and temperature. They use a similar RC model, but only for the microprocessor as a whole with a focus on the effects of thermal packaging. To the best of our knowledge, this is the first work to derive a thermal model for heating effects in individual architectural structures.

## 5. Simulation Technique and Metrics

### 5.1. Performance and Power Simulation

To model temperatures and controllers, we extend the DTM version of Wattch [3] used by Brooks and Martonosi. We use Wattch version 1.02, which is now widely used for research on power issues in architecture. Wattch in turn is based on the *sim-outorder* simulator of SimpleScalar [4] version 3.0. Wattch adds cycle-by-cycle tracking of power dissipation by estimating unit capacitances and activity factors. For Wattch, we chose a feature size of $0.18\mu$, a $V_{dd}$ of 2.0V, and a clock speed of 1.5 GHz, which is roughly representative of values in contemporary processors.

We extended the simulators in several ways. Because most processors today have pipelines longer than five stages, our simulations extend the pipeline by adding three additional stages between decode and issue. These stages model the extra renaming and enqueuing steps found in many pipelines today, like the Alpha 21264 [10], and are necessary to properly account for branch-resolution latencies and extra mis-speculated execution. The extra stages are also included in the power model. We also improve the performance simulator by updating the fetch model to count only one access (of fetch-width granularity) per cycle. We

improve the power simulation by using the improved access counts that result from these behavioral changes and by adding modeling of the column decoders on array structures like the branch predictor and caches [16]. These changes were straightforward, but were also validated by testing with microbenchmarks plus comparison with known results.

As a processor configuration, we approximately model the Alpha 21264, as shown in Table 2. The hybrid branch predictor [15] is SimpleScalar's slightly simplified version, using bimodal (plain 2-bit counter style) predictors as the chooser and as one of the components. The branch predictor is updated speculatively and repaired after a misprediction [19]. We do not model the register-cluster aspect of the 21264.

| Processor Core | |
|---|---|
| Instruction Window | 80-RUU, 40-LSQ |
| Issue width | 6 instructions per cycle (4 Int, 2 FP) |
| Functional Units | 4 IntALU,1 IntMult/Div, 2 FPALU,1 FPMult/Div, 2 mem ports |
| **Memory Hierarchy** | |
| L1 D-cache Size | 64 KB, 2-way LRU, 32 B blocks |
| L1 I-cache Size | 64 KB, 2-way LRU, 32 B blocks both 1-cycle latency |
| L2 | Unified, 2 MB, 4-way LRU, 32B blocks, 11-cycle latency, WB |
| Memory | 100 cycles |
| TLB Size | 128-entry, fully assoc., 30-cycle miss penalty |
| **Branch Predictor** | |
| Branch predictor | Hybrid: 4K bimod and 4K/12-bit/GAg 4K bimod-style chooser |
| Branch target buffer | 1 K-entry, 2-way |
| Return-address-stack | 32-entry |

**Table 2. Configuration of simulated processor microarchitecture.**

### 5.2. Temperature Modeling

Due to a lack of any published data on thermal properties for specific structures and a general lack of any information on per-structure areas (publicly available models for area are currently in development but not yet available), we were forced to derive thermal $R_{block}$ and $C_{block}$ values using estimates for area. We obtained these using one of the most recent die-photos that we could find with a floorplan, for the MIPS R10000 [7]. We scaled the resulting areas by two generations to $0.18\mu$ and by architectural size, and then combined these with the values for $c$ and $\rho$ from the previous section. This approach is clearly unsatisfactory in terms of the approximations it requires, and we are in the process of generating new data with areas provided by Wilcox and Manne for the Alpha 21264 [22]. We feel, however, that different ratios and areas of structure sizes would not materially affect the main conclusions of this paper. The use of lumped thermal R and C values is a general approach for modeling temperature. And regardless of structure sizes and temperature thresholds, if DTM is used for thermal management, feedback control confers the advantage of fine-tuning the thermal response to the operating conditions.

To derive thermal properties, we assumed a die thickness of 0.1 mm. This implies a "thinned" wafer necessary for removing heat in high-performance processors. The areas and corresponding $R_{block}$, $C_{block}$, and $RC_{block}$ values are presented in Table 3. For this paper, we explore the load-store queue, the instruction window (which includes physical registers for uncommitted instructions), the architectural register file, the branch predictor (including branch target buffer), the data cache, the integer execution unit, and the floating-point execution unit.

Temperature is computed on a cycle-by-cycle basis. First the SimpleScalar pipeline model determines the activity of each structure; then Wattch computes power dissipation for each of them ($P_i$); and finally our thermal model computes temperature based on the values of $R_{block}$, $C_{block}$, and the power dissipation in the past clock cycle. The specific difference equation for computing each $\Delta T$ for each structure $i$ is

$$\Delta T_i = \frac{P_i \cdot \Delta t}{C_i} + \frac{T_i \cdot \Delta t}{R_i \cdot C_i} \tag{5}$$

where $\Delta t$ is one clock cycle, 0.667 nanosec. The simulation cost of computing this for each block of interest is minor compared to the pipeline and power modeling.

We observed local dynamic temperature variations of up to $13^\circ$ (see Section 5.4). This number is within the range that peak power $\cdot R_{block}$ suggests. Of course, these temperature changes slowly raise the surrounding IC temperature. Because chip-wide temperature variations are on the order of seconds to minutes in the presence of a heat sink and fan, chip-wide temperatures cannot be modeled in Wattch in any reasonable amount of time. Instead we choose a baseline heatsink temperature of $100^\circ$ C based on the SIA roadmap [18]. We set the thermal emergency threshold at $108^\circ$ C based on the localized temperature ranges in [24]. At $108^\circ$, at least one of the benchmarks (if not controlled by DTM) puts each structure within $1^\circ$ of thermal emergency for some period of time, while there exist some benchmarks that never experience thermal emergencies, and some that never put any structure within $1^\circ$ of thermal emergency. This means that our benchmark set exhibits a range of thermal behavior.

Our metrics of success are the percentage of cycles spent in thermal emergency and percentage of the non-DTM IPC.

| structure | area ($m^2$) | peak power (W) | R (K/W) | C (J/K) | RC (J/W = sec) |
|---|---|---|---|---|---|
| LSQ | 5.0e-7 | 2.7 | 8 | 5.0e-5 | 400 us |
| inst. window | 9.0e-7 | 10.3 | 0.9 | 9.0e-5 | 81 us |
| regfile | 2.5e-7 | 5.5 | 4 | 2.5e-5 | 100 us |
| bpred | 3.5e-7 | 5.3 | 1.4 | 3.5e-5 | 49 us |
| D-cache | 1.0e-6 | 11.6 | 1 | 1.0e-4 | 100 us |
| int exec. unit | 5.0e-7 | 4.3 | 2 | 5.0e-5 | 100 us |
| FP exec. unit | 5.0e-7 | 6.7 | 2 | 5.0e-5 | 100 us |
| chip | | 112.9 | 0.34 | 340 | 115 sec |

**Table 3. Per-structure area and thermal-R and thermal-C estimates.**
For comparison, a value for chip-wide thermal R and C is also provided for the chip as a whole with heatsink [5].

Our rule is that the temperature for any structure must never exceed the emergency temperature. Because DTM slows down or turns off parts of the processor, IPC can never exceed the baseline value, and the benchmarks with extreme thermal behavior will necessarily engage DTM and experience some slowdown. We are unaware of any technique for modeling optimal DTM to obtain a lower bound on the induced performance loss, so we simply try to make the induced performance loss as small as possible.

### 5.3. DTM Mechanisms

As mentioned earlier, of the five DTM mechanisms Brooks and Martonosi propose, we selected fetch toggling, specifically toggle1, as the vehicle for applying adaptive control.

For the CT-DTM mechanisms, we assume the presence of dedicated hardware—an adder and a multiplier—to perform the controller computations. These need not be especially fast, reducing their already small cost, and they operate so infrequently that their contribution to power and temperature should be negligible. An alternative is to inject instructions directly into the pipeline to perform the controller computations. To avoid interrupting the pipeline, special instructions that access 2–3 special registers would be required. Because the controllers only execute at most every 1000 cycles, the impact of this too should be negligible. Indeed, we could likely have used a longer sampling interval without significantly affecting accuracy, since the thermal time constants are on the order of tens to hundreds of microsec., which is much greater than 667 nanosec. Determining the best sampling interval and modeling the associated overhead of computing the controller in terms of both performance and power are areas for future work.

In applying the different controllers to fetch toggling, we assume that the controller can vary the degree of toggling among eight discrete values distributed evenly across the range from 0% to 100%.

To more clearly demonstrate the benefit of applying control theory to design adaptive controllers, we also manually developed a controller ("M") whose response is proportional to the temperature. This mechanism simply sets the toggling rate equal to the percentage error in temperature, where $107^\circ$ and less represents 0, and $108^\circ$ and above represents 100%. For example, a temperature of $107.5^\circ$ would correspond to an error of 50% and toggle the pipeline every other cycle.

The DTM schemes proposed by Brooks and Martonosi use some *thermal trigger* temperature level, less than the emergency level, at which the DTM mechanism is engaged. The CT-DTM schemes we propose also require a similar trigger level, this time to use as the setpoint. For the non-CT schemes (toggle1 and M), we use a trigger level of $107^\circ$. We found that a level any closer to the emergency threshold caused toggle1 to fail to stop some emergencies. For the P controller, we used a setpoint of $107.5^\circ$, and a sensor range of $107.0^\circ - 108.0^\circ$; in other words, the "trigger" threshold is 107.0, as with toggle1 and M. For the PI and PID controllers, on the other hand, their more robust control permitted us to use a setpoint of $107.9^\circ$ for the PI and PID controllers, with a sensor range of $107.8^\circ - 108.0^\circ$; in other words, the "trigger" threshold above which toggling starts to engage for these controllers is 107.8. We did also test a lower setpoint of $107.5^\circ$ (sensor range of $107.0^\circ$ to $108.0^\circ$) to see how much the choice of setpoint affects performance—see Section 7.

For all the DTM techniques, we also assume a direct microarchitectural technique for signaling temperature effects, avoiding the overhead of OS interrupts as Brooks and Martonosi suggested.

### 5.4. Benchmarks

We evaluate our results using benchmarks from the SPEC CPU2000 suite [20]. The benchmarks are compiled and statically linked for the Alpha instruction set using the Compaq Alpha compiler with SPEC *peak* settings and include all linked libraries. For each program, we skip the first 2 billion instructions to avoid unrepresentative behavior at the beginning of the program's execution, and then simulate 200 million (committed) instructions using the reference input set. Simulation is conducted using SimpleScalar's EIO

traces to ensure reproducible results for each benchmark across multiple simulations.

| | Avg. IPC | Avg. pwr (W) | Avg. temp. °C | Above 108° | Above 107° |
|---|---|---|---|---|---|
| 164.gzip | 2.1 | 47.7 | 43.2 | 0.0 | 0.0 |
| 168.wupwise | 2.0 | 46.0 | 42.6 | 0.0 | 0.0 |
| 175.vpr | 1.7 | 45.0 | 42.3 | 0.0 | 0.0 |
| 176.gcc | 2.0 | 52.7 | 44.9 | 81.9 | 99.2 |
| 177.mesa | 2.4 | 50.7 | 44.2 | 0.5 | 46.4 |
| 179.art | 1.2 | 39.0 | 40.3 | 3.9 | 5.9 |
| 183.equake | 2.8 | 56.4 | 46.2 | 99.7 | 99.8 |
| 186.crafty | 2.3 | 50.4 | 44.1 | 0.0 | 6.8 |
| 187.facerec | 2.3 | 49.9 | 44.0 | 0.0 | 63.8 |
| 191.fma3d | 2.6 | 48.0 | 43.3 | 99.6 | 99.7 |
| 197.parser | 1.4 | 40.6 | 40.8 | 4.1 | 24.8 |
| 252.eon | 2.3 | 52.2 | 44.8 | 0.0 | 98.7 |
| 253.perlbmk | 2.8 | 47.3 | 43.1 | 48.4 | 48.5 |
| 254.gap | 2.1 | 45.2 | 42.4 | 0.0 | 0.0 |
| 255.vortex | 2.1 | 47.4 | 43.1 | 0.0 | 97.2 |
| 256.bzip2 | 2.0 | 46.6 | 42.8 | 13.3 | 61.7 |
| 300.twolf | 1.5 | 40.3 | 40.7 | 0.0 | 0.0 |
| 301.apsi | 0.9 | 33.6 | 38.4 | 0.0 | 0.2 |

**Table 4. Average IPC, power, and temperature characteristics for each benchmark**

"Avg. temp." assumes that the heatsink temperature is at an ambient of 27° C and uses the chip-wide thermal-R of 0.34 K/W. Percent of cycles spent in thermal emergency (above $108^\circ$) and in a level of thermal stress (above $107^\circ$) assume no thermal management and that the heatsink temperature has risen to $100^\circ$ and use the per-structure thermal R/C values.

| Extreme | High | Medium | Low |
|---|---|---|---|
| gcc | mesa | facerec | gzip |
| equake | art | eon | wupwise |
| fma3d | parser | vortex1 | vpr |
| perlbmk | bzip2 | | crafty |
| | | | twolf |
| | | | apsi |

**Table 5. Categories of thermal behavior.**

Due to the extensive number of simulations required for this study, we used only 18 of the total 26 SPEC2k benchmarks. A mixture of integer and floating-point programs with low, intermediate, and extreme thermal demands were chosen. Table 4 provides a list of the benchmarks we study along with their basic performance, power, and thermal characteristics. Table 5 breaks them into categories of thermal behavior, and Table 6 provides more detail by showing per-structure mean and maximum temperatures.

In addition to the eight benchmarks that experience actual emergencies, several others spend significant amounts of time within $1^\circ$ of the emergency threshold. Table 4 lists the percent of cycles spent above $108^\circ$ as well as the percent of cycles spent above $107^\circ$, and Tables 7 and 8 break this down further by structure. Of particular interest are programs like *mesa, facerec, eon*, and *vortex*, which spend as much as 98% of their time above $107^\circ$, yet spend almost no time in thermal emergency. Any successful DTM scheme should minimize the penalties for these programs, and indeed the CT-DTM techniques provide some of the best improvements for this category. The program *art*, on the other hand, has the opposite behavior. It spends only 6% of its time above $107^\circ$, but more than half of these cycles—4%—are spent in emergency, meaning that *art* has very bursty thermal behavior.

Using these two tables together, we break the set of benchmarks into four categories of thermal behavior: extreme, high, medium, and low, as shown in Table 5.

## 6. Comparison of Power and RC-based Temperature Modeling

Before exploring the effectiveness of the CT-DTM techniques, we briefly demonstrate the importance of modeling localized temperatures. Prior work has only used boxcar averages of power measurements over $N$ cycles, where $N = 1 - 100$ K cycles.

We performed two sets of experiments. In the first, we measure thermal emergencies using the boxcar average of per-structure power dissipation over 10 K and 500 K cycles. We are then able to count, on a per-structure-basis, how many true thermal emergencies the power-based modeling fails to observe and how many unnecessary thermal triggers it generates. In the second, we measure thermal emergencies using boxcar averages of power dissipation on a chip-wide basis for the same window sizes. We are then able to count, how many true thermal emergencies the chip-wide treatment fails to observe and how many unnecessary thermal triggers it generates.

In both cases, the same simulation also runs our temperature model as a reference using all the parameters specified in the previous section. For chip-wide power, a trigger occurs when the boxcar average exceeds a specified trigger-threshold value for power, 47 W in this case. For per-structure power values, we can more directly tie the average power readings to the thermal model. For each structure, a trigger occurs whenever $P_{avg}R > trigger$ where $R$ is based on the values in the previous section.

The results appear in Tables 9 and 10. Table 9 shows that both a very small and a very large window produce significant variations between the average-power and the direct-temperature model for some programs. Although the average frequency of missed emergency and false trigger cycles for the structure-specific average-power model is small on

| | Avg. IPC | Avg. pwr (W) | Avg. temp. °C | Max/Avg. Temperature (°C) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | LSQ | window | regfile | bpred | D-cache | ALU | FPALU |
| 164.gzip | 2.1 | 47.7 | 43.2 | 105.9/106.6 | 105.3/105.9 | 105.2/106.8 | 105.9/106.9 | 104.9/106.1 | 105.1/106.2 | 101.3/101.3 |
| 168.wupwise | 2.0 | 46.0 | 42.6 | 105.1/105.1 | 104.5/104.6 | 105.1/105.1 | 105.9/106.0 | 104.1/104.2 | 103.5/103.5 | 104.8/104.9 |
| 175.vpr | 1.7 | 45.0 | 42.3 | 105.8/106.0 | 104.7/104.9 | 104.7/104.9 | 105.8/106.1 | 104.6/104.8 | 104.3/104.5 | 101.9/102.0 |
| 176.gcc | 2.0 | 52.7 | 44.9 | 108.3/109.1 | 105.3/105.6 | 105.2/105.7 | 105.3/106.4 | 107.9/109.3 | 104.6/104.8 | 101.3/101.3 |
| 177.mesa | 2.4 | 50.7 | 44.2 | 106.8/107.6 | 105.3/105.7 | 105.5/105.9 | 106.1/106.6 | 105.7/110.6 | 104.8/105.3 | 103.2/104.5 |
| 179.art | 1.2 | 39.0 | 40.3 | 104.5/106.9 | 103.1/105.2 | 104.1/106.0 | 105.2/109.4 | 103.5/106.6 | 102.6/104.0 | 102.7/105.9 |
| 183.equake | 2.8 | 56.4 | 46.2 | 107.2/107.3 | 106.1/106.2 | 104.9/105.0 | 109.2/109.3 | 106.0/106.1 | 106.2/106.2 | 101.3/101.3 |
| 186.crafty | 2.3 | 50.4 | 44.1 | 106.8/107.2 | 105.5/105.8 | 105.2/105.4 | 106.5/107.0 | 105.7/106.1 | 105.4/105.9 | 101.3/101.3 |
| 187.facerec | 2.3 | 49.9 | 44.0 | 106.2/107.6 | 104.8/106.2 | 105.6/106.6 | 104.6/106.4 | 105.7/107.3 | 103.8/105.2 | 105.5/107.9 |
| 191.fma3d | 2.6 | 48.0 | 43.3 | 104.4/104.4 | 105.3/105.3 | 105.0/105.0 | 108.8/108.9 | 103.4/103.4 | 105.9/105.9 | 101.3/101.3 |
| 197.parser | 1.4 | 40.6 | 40.8 | 105.3/107.9 | 103.7/106.1 | 103.7/105.4 | 106.0/110.4 | 104.1/106.7 | 103.6/105.9 | 101.3/101.5 |
| 252.eon | 2.3 | 52.2 | 44.8 | 107.5/107.7 | 105.3/105.5 | 104.9/105.2 | 106.7/107.3 | 106.5/106.8 | 104.7/105.1 | 103.1/103.2 |
| 253.perlbmk | 2.8 | 47.3 | 43.1 | 104.1/106.0 | 107.1/109.7 | 105.7/107.4 | 104.9/106.0 | 103.1/105.0 | 106.4/108.7 | 101.4/101.5 |
| 254.gap | 2.1 | 45.2 | 42.4 | 105.8/106.5 | 104.6/104.8 | 104.7/104.9 | 105.2/105.9 | 105.1/105.5 | 104.8/105.0 | 101.4/101.4 |
| 255.vortex | 2.1 | 47.4 | 43.1 | 106.6/106.9 | 104.8/105.0 | 104.3/104.5 | 107.2/107.6 | 105.8/106.4 | 104.8/104.9 | 101.4/101.4 |
| 256.bzip2 | 2.0 | 46.6 | 42.8 | 106.0/108.1 | 104.8/105.9 | 104.6/105.3 | 106.6/112.7 | 105.0/107.0 | 104.9/106.0 | 101.3/101.3 |
| 300.twolf | 1.5 | 40.3 | 40.7 | 104.8/105.0 | 103.9/104.0 | 104.0/104.0 | 105.6/105.8 | 103.8/103.9 | 103.7/103.8 | 101.7/101.8 |
| 301.apsi | 0.9 | 33.6 | 38.4 | 103.6/107.3 | 102.4/106.0 | 104.0/106.5 | 104.0/104.9 | 102.7/106.8 | 101.7/105.0 | 103.6/104.9 |

**Table 6. Average IPC, power, and temperature for each benchmark (assuming an ambient of 27° C and a thermal R of 0.34 K/W), plus avg./max. temperatures for individual structures assuming a current operating temperature of 100° C and no thermal management.**

| | Any | LSQ | window | regfile | bpred | D-cache | ALU | FPALU |
|---|---|---|---|---|---|---|---|---|
| 164.gzip | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168.wupwise | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 175.vpr | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 176.gcc | 81.9 | 81.8 | 0.0 | 0.0 | 0.0 | 50.8 | 0.0 | 0.0 |
| 177.mesa | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 |
| 179.art | 3.9 | 0.0 | 0.0 | 0.0 | 3.9 | 0.0 | 0.0 | 0.0 |
| 183.equake | 99.7 | 0.0 | 0.0 | 0.0 | 99.7 | 0.0 | 0.0 | 0.0 |
| 186.crafty | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 187.facerec | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 191.fma3d | 99.6 | 0.0 | 0.0 | 0.0 | 99.6 | 0.0 | 0.0 | 0.0 |
| 197.parser | 4.1 | 0.0 | 0.0 | 0.0 | 4.1 | 0.0 | 0.0 | 0.0 |
| 252.eon | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 253.perlbmk | 48.4 | 0.0 | 48.4 | 0.0 | 0.0 | 0.0 | 48.2 | 0.0 |
| 254.gap | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 255.vortex1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 256.bzip2 | 13.3 | 12.0 | 0.0 | 0.0 | 1.3 | 0.0 | 0.0 | 0.0 |
| 300.twolf | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 301.apsi | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

**Table 7. Percent of cycles each structure spends in thermal emergency (above $108^\circ$) without thermal management.**

| | Any | LSQ | window | regfile | bpred | D-cache | ALU | FPALU |
|---|---|---|---|---|---|---|---|---|
| 164.gzip | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168.wupwise | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 175.vpr | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 176.gcc | 99.2 | 99.1 | 0.0 | 0.0 | 0.0 | 92.5 | 0.0 | 0.0 |
| 177.mesa | 46.4 | 46.0 | 0.0 | 0.0 | 0.0 | 0.7 | 0.0 | 0.0 |
| 179.art | 5.9 | 0.0 | 0.0 | 0.0 | 5.9 | 0.0 | 0.0 | 0.0 |
| 183.equake | 99.8 | 98.3 | 0.0 | 0.0 | 99.8 | 0.0 | 0.0 | 0.0 |
| 186.crafty | 6.8 | 6.8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 187.facerec | 63.8 | 49.1 | 0.0 | 0.0 | 0.0 | 49.0 | 0.0 | 52.9 |
| 191.fma3d | 99.7 | 0.0 | 0.0 | 0.0 | 99.7 | 0.0 | 0.0 | 0.0 |
| 197.parser | 24.8 | 15.6 | 0.0 | 0.0 | 19.2 | 0.0 | 0.0 | 0.0 |
| 252.eon | 98.7 | 98.7 | 0.0 | 0.0 | 10.6 | 0.0 | 0.0 | 0.0 |
| 253.perlbmk | 48.5 | 0.0 | 48.5 | 48.1 | 0.0 | 0.0 | 48.4 | 0.0 |
| 254.gap | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 255.vortex | 97.2 | 0.0 | 0.0 | 0.0 | 97.2 | 0.0 | 0.0 | 0.0 |
| 256.bzip2 | 61.7 | 18.2 | 0.0 | 0.0 | 43.5 | 0.0 | 0.0 | 0.0 |
| 300.twolf | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 301.apsi | 0.2 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

**Table 8. Percent of cycles spent above $107^\circ$ without thermal management.**

average, this is heavily influenced by the programs that exhibit little thermal stress.

| | 500K | | 10K | |
|---|---|---|---|---|
| | missed-e | false-t | missed-e | false-t |
| 164.gzip | 0.00 | 0.00 | 0.00 | 7.77 |
| 168.wupwise | 0.00 | 0.00 | 0.00 | 0.00 |
| 175.vpr | 0.00 | 0.00 | 0.00 | 0.09 |
| 176.gcc | 3.92 | 0.03 | 28.81 | 0.24 |
| 177.mesa | 0.20 | 7.08 | 0.07 | 22.81 |
| 179.art | 2.39 | 1.66 | 0.09 | 35.18 |
| 183.equake | 0.36 | 0.00 | 0.04 | 0.20 |
| 186.crafty | 0.00 | 2.10 | 0.00 | 29.36 |
| 187.facerec | 0.00 | 0.64 | 0.00 | 3.34 |
| 191.fma3d | 0.26 | 0.00 | 0.01 | 0.19 |
| 197.parser | 2.13 | 4.46 | 1.07 | 13.57 |
| 252.eon | 0.00 | 0.58 | 0.00 | 1.20 |
| 253.perlbmk | 0.26 | 0.00 | 0.00 | 0.12 |
| 254.gap | 0.00 | 0.00 | 0.00 | 0.00 |
| 255.vortex | 0.00 | 0.93 | 0.00 | 0.98 |
| 256.bzip2 | 0.82 | 4.68 | 1.18 | 7.58 |
| 300.twolf | 0.00 | 0.00 | 0.00 | 0.00 |
| 301.apsi | 0.00 | 0.22 | 0.00 | 0.41 |
| MEAN | 0.57 | 1.24 | 1.74 | 6.84 |

**Table 9. Comparison of *per-structure* power-averaging to direct-temperature measurements.**

Percent of cycles spent in emergency according to our temperature model but not according to the per-structure average-power model (missed-e); and percent of cycles spent in trigger mode according to the per-structure average-power model but not according to our temperature model (false-t). The left-hand pair of data columns use a boxcar average of power measurements over the last 500 K cycles, and the right-hand pair uses 10 K cycles.

Compared to a window of 10 K cycles, averaging power over 500 K cycles usually causes a slight increase in the number of missed emergency cycles but a substantial drop in the number of false triggers. That is because 500 K cycles, which corresponds to 333 microsec., is 2–3 times the per-structure RC constants shown in Table 3—in other words, 500 K-cycles is about how long a sustained increase in power dissipation takes to reach a new, steady-state temperature. Unfortunately, the relationship is not a direct one, as can be seen for *gcc*, where the 10 K window produces many more false emergencies.

Table 10 shows that using *chip-wide* average-power numbers causes large variations for some programs compared to both the per-structure average-power model and the direct-temperature model. Per-structure modeling, regardless of the type of model, gives a much more detailed view of thermal activity and detects hot spots that are hidden by the chip-wide average.

| | missed-e | false-t |
|---|---|---|
| 164.gzip | 0.00 | 23.26 |
| 168.wupwise | 0.00 | 0.00 |
| 175.vpr | 0.00 | 0.00 |
| 176.gcc | 0.57 | 0.00 |
| 177.mesa | 0.27 | 44.78 |
| 179.art | 3.86 | 0.00 |
| 183.equake | 0.36 | 0.00 |
| 186.crafty | 0.00 | 89.02 |
| 187.facerec | 0.00 | 4.35 |
| 191.fma3d | 99.61 | 0.00 |
| 197.parser | 4.03 | 0.00 |
| 252.eon | 0.00 | 0.56 |
| 253.perlbmk | 0.48 | 0.00 |
| 254.gap | 0.00 | 0.00 |
| 255.vortex | 0.00 | 0.00 |
| 256.bzip2 | 1.26 | 0.40 |
| 300.twolf | 0.00 | 0.00 |
| 301.apsi | 0.00 | 0.29 |
| MEAN | 6.14 | 9.04 |

**Table 10. Comparison of *chip-wide* power-averaging to direct-temperature measurements.**

Percent of cycles spent in emergency according to our temperature model but not according to the chip-wide average-power model (missed-e); and percent of cycles spent in trigger mode according to the chip-wide average-power model but not according to our temperature model (false-t). The average-power model uses a boxcar average of power measurements over the last 500 K cycles.

We cannot yet prove that one model is superior to another, although clearly chip-wide power is a poor indicator of thermal behavior. Per-structure power averages, on the other hand, track direct-temperature measurements tolerably closely, within a few percent in most cases. We nevertheless strongly discourage the use of average power as a proxy for temperature for several reasons. Although we are still in the process of validating the temperature model we have developed here, it is founded on basic thermo-electric properties, which cannot be said for a simple boxcar average of power measurements. There is no validation or even established method for accurately converting power measurements into temperature measurements, so there is no reason to prefer average power over direct temperature. A further problem is that it is difficult to choose a *thermal* trigger using an average *power* value. We explored various trigger levels ranging from 47 W to 55 W, and found that different triggers worked best for different programs in terms of most closely replicating the direct-temperature model.

Because the temperature model proposed in this paper provides direct modeling of localized heating in terms of temperature, because it is easy to use and derive, and because it is computationally efficient, we advocate this as the best model currently available for researchers wishing to model temperature at the processor-architecture level.

## 7. Effectiveness of Control-Theoretic Techniques for Thermal Management

Now we evaluate the effectiveness of formal feedback control for DTM. We developed P, PI, and PID controllers for pipeline toggling, and compared them to both the static toggle1 policy and the manually-derived feedback-control policy (M) that is not based on control theory but attempts to set the degree of toggling proportionally to the error. We also developed a PD controller, which is discussed later in this section. As mentioned before, the sampling rate for all the controllers is set to 1000 cycles to match the policy delay for the toggle1 mechanism.

To make the PI and PID controllers effective at eliminating thermal emergencies, we did have to implement anti-windup. Otherwise, the PI and PID controllers failed to prevent thermal emergencies. This behavior was especially prevalent for *perlbmk*, which spends over half its time below the thermal trigger, accumulating a large negative integral. *Gcc*, on the other hand, does not have this problem, because it spends almost 100% of its time with at least one structure above $107^\circ$.

Figure 4 therefore shows the performance of the non-CT toggle1 controller, the manually-derived proportional controller (M), and the CT-toggle controllers (with anti-windup for the PI and PID controllers). Data for the PD controller is omitted for simplicity, since the PD controller is slightly inferior to the PI and PID controllers (but see below for more information). All were able to completely eliminate thermal emergencies, so the metric of interest is performance. Recall that DTM only slows or disables parts of the processor, so the best we can accomplish is to minimize the slowdown. The figure therefore plots the percentage *loss* in performance compared to the baseline case with no DTM.

The adaptive techniques are substantially better than the fixed toggle1 policy, with the PI and PID techniques cutting the performance loss by about 65% compared to toggle1, and by about 37% compared to the M controller. Even the P controller cuts the performance loss by 21% compared to the M controller. Furthermore, the CT-DTM techniques are always better than the toggle1 policy, and almost always better than the M controller except for *equake* and a tiny difference for *fma3d* in the case of the PI and PID controllers, and *gcc* and *equake* for the P controller.

The choice of a higher setpoint ($107.9^\circ$) for the PI and PID controllers is essential to their good performance. We also tried a setpoint of $107.5^\circ$ for these controllers (the same setpoint used by the P controller). The results appear in Figure 5. A PD controller is included in this graph too, for completeness.

The PI and PID controllers still give impressive gains compared to the toggle1 controller, but are actually inferior to the simpler P controller. The reason for this is that these controllers are actually doing *too* good a job of holding temperature to the setpoint. The P controller allows temperatures to rise as high as $107.9^\circ$, while the other controllers hold the temperature extremely close to the setpoint of $107.5^\circ$. Indeed, the temperature never exceeds $107.53^\circ$ for any structure for the PD controller, and never exceeds $107.51^\circ$ for the PI and PID controllers. This means that the setpoint for these controllers should be set to a higher temperature. Ideally, an analysis of maximum overshoot would use the maximum possible power dissipation in each structure, the thermal RC time constant, and the response time of the controller to identify the maximum value for setpoint that still *guarantees* no thermal emergencies. For example, if $0.01^\circ$ were proven to be the maximum overshoot for the PID controller, then the setpoint could actually be placed at $107.99^\circ$ while preserving the guarantee that the emergency level of $108^\circ$ would never be exceeded. That analysis is beyond the scope of this paper. Instead, for our main results in Figure 4, we simply used the maximum observed temperatures to conclude that a setpoint of $107.9^\circ$ is safe for this workload with the PI and PID controllers.

The most dramatic gains for CT-DTM come from the benchmarks with extreme thermal demands and the "medium" benchmarks that operate very close to thermal emergency but rarely actually experience emergencies (*mesa, facerec, eon*, and *vortex*). This latter category is most severely penalized by a fixed policy, and benefits most from the integrating action of the PI and PID controllers because they permit a much higher setpoint, so that these "medium" benchmarks rarely or never engage any toggling.

Of course, such fine control over temperature may be overkill in an environment where the number of temperature sensors on chip is likely to be limited (perhaps four to eight at most), where thermal propagation and the intrinsic response of the sensors themselves imply a lag between heating at hot spots and the time at which this is detected by a sensor, and where imprecision in the sensor may mean accurate sensing and control to within fractions of a degree may be impossible.

Improved methods for distributed temperature sensing can remove most of these problems. But the main point of this paper is broader than the specific mechanisms described here. Rather, we used DTM as a vehicle to show that formal feedback control theory offers a variety of advantages for adaptive techniques in processor architecture, and the results presented here demonstrate that clearly.

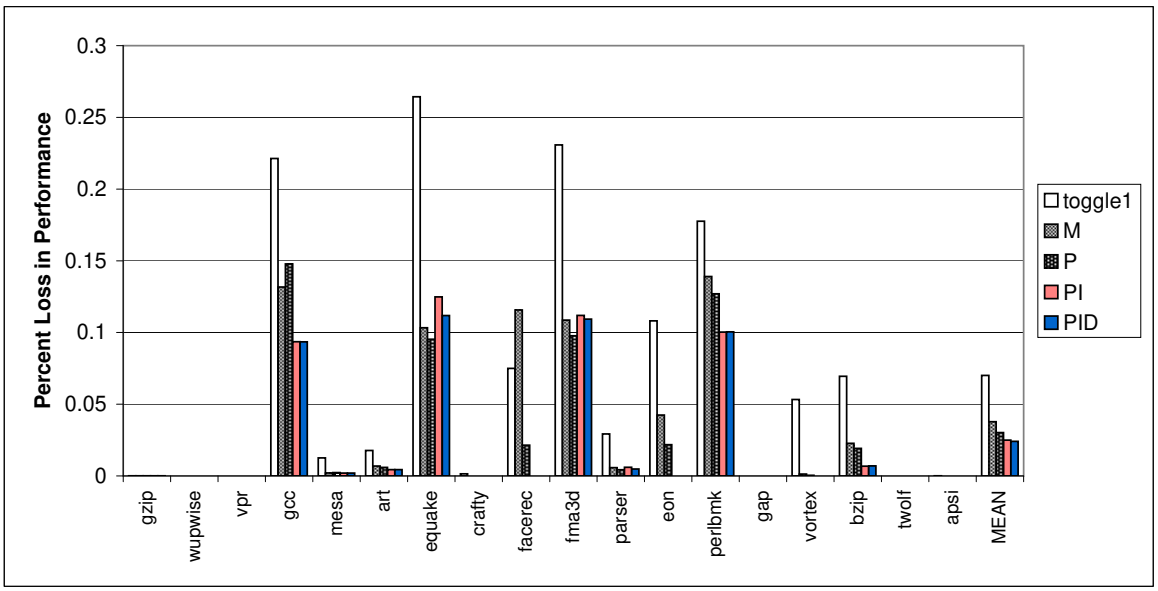


**Figure 4. Performance loss for various DTM techniques relative to the IPC with no DTM. Smaller bars mean less loss in performance.**

The trigger threshold for the PI and PID controllers has been set to $107.8^\circ$ (*i.e.*, a setpoint of $107.9^\circ$). The mean reduction in performance loss compared to toggle1 is 64% for PI and 66% for PID.

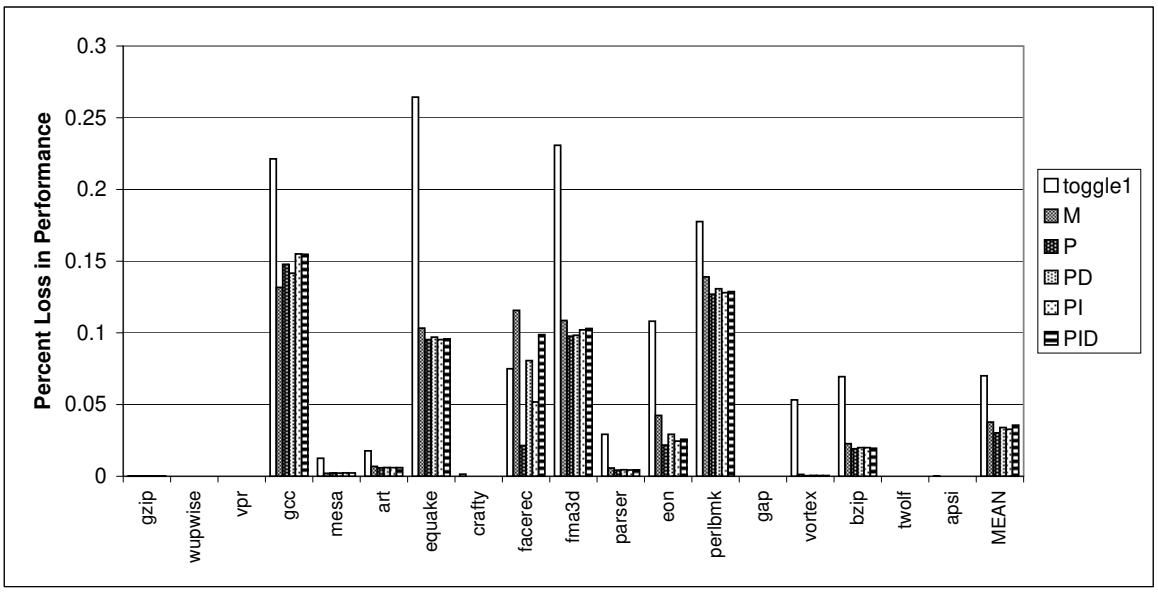


**Figure 5. Performance loss for various DTM techniques relative to the IPC with no DTM, but with a setpoint of $107.5^\circ$ for all the control-theoretic controllers (*i.e.*, a trigger threshold of $107.0^\circ$ for all the mechanisms).**

The mean reduction in performance loss compared to the fixed toggle1 policy is 46% for M, 57% for P, 51% for PD, 53% for PI, and 49% for PID.

## 8. Conclusions and Future Work

DTM pipeline toggling based on control-theoretic techniques is more effective than prior techniques at avoiding thermal emergencies while minimizing performance loss. Our CT-DTM techniques completely prevented thermal emergencies while reducing the loss in performance by 65% compared to the previously proposed "toggle1" technique and 36% over our own hand-designed proportional toggling technique. This shows that the benefits of CT-DTM come not just from their adaptivity, but that they also derive substantial benefit from the formal control theory from which they are derived.

Overall, this paper makes three contributions to the field of architecture-level dynamic thermal management.

1. We introduce the use of a new and more accurate thermal model that models temperature directly and is also computationally inexpensive.

2. We invoke DTM techniques in response to localized hot spots rather than chip-wide effects.

3. We apply control-theoretic techniques to control the thermal-management techniques. These controllers provide much more adaptive control of the temperature and thus minimize performance penalties.

We feel that the most important contribution of this work is our demonstration that formal feedback control theory offers compelling advantages. The control-theoretic techniques are effective because they engage only as much toggling as necessary, while holding the temperature so tightly to the setpoint that we were able to establish a setpoint of $107.9^\circ$ but never exceed the emergency threshold of $108^\circ$. The controllers were also extremely easy to design, requiring almost no tuning.

More generally, our results suggest that applying control-theoretic techniques to other aspects of adaptive processor control is an extremely promising research area. The design of the controllers is itself a rich area for research, because in addition to choosing an overall controller algorithm, the controller parameters can be adjusted to formally determine the choice of thresholds and to guarantee settling times. This, for example, can be used to choose the highest possible setpoint that still guarantees successful regulation of temperature.

This work is just the first step in what we expect to be a long-range effort in thermal modeling and management. Our results so far suggests a variety of avenues for future work: more detailed thermal modeling to account for the second-order effects of thermal coupling among hot spots, thermal modeling and feedback control that account for the presence of only a limited quantity of temperature sensors and the associated sensor lag, validation of the thermal model against a circuit-based power model, new thermal management mechanisms, and application of control theory to other aspects of dynamic power and performance management.

We hope that the material presented here will provide the necessary foundation to help other architects perform more detailed thermal modeling and to help other architects apply control theory in the design of adaptive microarchitecture techniques.

## Acknowledgments

This material is based upon work supported in part by the National Science Foundation under grant nos. CCR-0105626, CCR-0098269, MIP-9703440 (CAREER), and a grant from Intel MRL. We would like to thank George Cai, Margaret Martonosi, and David Brooks for their helpful comments; David Brooks for his extensive help in validating our DTM techniques within Wattch; Zhigang Hu for his help with the SimpleScalar EIO traces; and the anonymous reviewers for many helpful suggestions on how to improve this paper.

## References

[1] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, pages 23–29, Jul.–Aug. 1999.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, pages 171–82, Jan. 2001.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.

[4] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.

[5] J. A. Chavez et al. Spice model of thermoelectric elements including thermal effects. In *Proceedings of the Instrumentation and Measurement Technology Conference*, pages 1019–1023, 2000.

[6] A. Dhodapkar, C. H. Lim, G. Cai, and W. R. Daasch. TEMPEST: A thermal enabled multi-model power/performance estimator. In *Proceedings of the Workshop on Power-Aware Computer Systems*, Nov. 2000.

[7] MIPS R10000 die photo. From website: CPU Info Center. http://bwrc.eecs.berkeley.edu/CIC/die_photos/#mips.

[8] N. Dragone, A. Aggarwal, and L. R. Carley. An adaptive on-chip voltage regulation technique for low-power applications. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 20–24, July 2000.

[9] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A control theoretic analysis of RED. In *Proceedings of IEEE INFOCOM*, Apr. 2001.

[10] R. E. Kessler, E. J. McLellan, and D. A. Webb. The Alpha 21264 microprocessor architecture. In *Proceedings of the 1998 International Conference on Computer Design*, pages 90–95, Oct. 1998.

[11] Al Krum. Thermal management. In Frank Kreith, editor, *The CRC handbook of thermal engineering*, pages 2.1–2.92. CRC Press, Boca Raton, FL, 2000.

[12] F. J. De la Hidalga and M. J. Deen. Theoretical and experimental characterization of self-heating in silicon integrated devices operating at low temperatures. *IEEE Transactions on Electron Devices*, 47(5):1098–1106, May 2000.

[13] C. Lu, J. A. Stankovic, G. Tao, , and S. H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems Journal*, Mar.-Apr. 2002. To appear.

[14] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: speculation control for energy reduction. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 132–41, June 1998.

[15] S. McFarling. Combining branch predictors. Tech. Note TN-36, DEC WRL, June 1993.

[16] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. Stan. Power issues related to branch prediction. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, Feb. 2002.

[17] H. Sanchez et al. Thermal management system for high-performance PowerPC microprocessors. In *COMPCON*, page 325, 1997.

[18] SIA. *International Technology Roadmap for Semiconductors*, 1999.

[19] K. Skadron, D. W. Clark, and M. Martonosi. Speculative updates of local and global branch history: A quantitative analysis. *Journal of Instruction-Level Parallelism*, Jan. 2000. (http://www.jilp.org/vol2).

[20] Standard Performance Evaluation Corporation. SPEC CPU2000 Benchmarks. http://www.specbench.org/osg/cpu2000.

[21] D. C. Steere et al. A feedback-driven proportion allocator for real-rate scheduling. In *Proceedings of the Symposium on Operating System Principles*, Feb. 1999.

[22] K. Wilcox and S. Manne. Alpha processors: A history of power issues and a look to the future. In *Proceedings of the Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design*, pages 16–37, Nov. 1999.

[23] L. S. Y. Wong, S. Hossain, and A. Walker. Leakage current cancellation technique for low power switched-capacitor circuits. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 310–15, Aug. 2001.

[24] T.-D. Yuan and B.-Z. Hong. Thermal management for high performance integrated circuits with non-uniform chip power considerations. In *Proceedings of the Seventeenth SEMI-THERM Symposium*, pages 95–101, Mar. 2001.