Temperature-Aware Microarchitecture: Modeling and Implementation

KEVIN SKADRON, MIRCEA R. STAN, KARTHIK SANKARANARAYANAN, WEI HUANG, SIVAKUMAR VELUSAMY, and DAVID TARJAN
University of Virginia

With cooling costs rising exponentially, designing cooling solutions for worst-case power dissipation is prohibitively expensive. Chips that can autonomously modify their execution and power-dissipation characteristics permit the use of lower-cost cooling solutions while still guaranteeing safe temperature regulation. Evaluating techniques for this *dynamic thermal management* (DTM), however, requires a thermal model that is practical for architectural studies.

This paper describes *HotSpot*, an accurate yet fast and practical model based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. Validation was performed using finite-element simulation. The paper also introduces several effective methods for DTM: "temperature-tracking" frequency scaling, "migrating computation" to spare hardware units, and a "hybrid" policy that combines fetch gating with dynamic voltage scaling. The latter two achieve their performance advantage by exploiting instruction-level parallelism, showing the importance of microarchitecture research in helping control the growth of cooling costs.

Modeling temperature at the microarchitecture level also shows that power metrics are poor predictors of temperature, that sensor imprecision has a substantial impact on the performance of DTM, and that the inclusion of lateral resistances for thermal diffusion is important for accuracy.

Categories and Subject Descriptors: C.5.3 [**Computer Systems Organization**]: Computer Systems Implementation—*Microcomputers and Microprocessors—thermal management*

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Dynamic compact thermal models, dynamic thermal management, dynamic voltage scaling, feedback control, fetch gating

1. INTRODUCTION

In recent years, power density in high-end microprocessors has doubled every three years [Borkar 1999; Mahajan 2002]. Because energy consumed by a chip is converted into heat, this continuing exponential rise in power density creates vast difficulties in cooling costs. For high-performance processors, cooling

This work was conducted while David Tarjan visited University of Virginia during his diploma program at the Swiss Federal Institute of Technology, Zürich.

Authors' address: K. Skadron, K. Sankaranarayanan, S. Velusamy, and D. Tarjan, Department of Computer Science, University of Virginia, 151 Engineer's Way Box 400740, Charlottesville, VA 22904-4740, skadron@cs.virginia.edu; M. R. Stan and W. Huang, Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA.

solutions cost $1–3 or more per watt of heat dissipated [Borkar 1999; Gunther et al. 2001]. Cooling costs are therefore also rising exponentially, threatening industry's ability to deploy cost-effective new systems.

Power-aware design alone has failed to stem this tide, requiring *temperature*-aware design at all system levels, including the processor architecture. Temperature-aware design makes use of power-management techniques, but often in ways that are different from those used to improve battery life or regulate peak power. Localized heating occurs much faster than chip-wide heating; since power dissipation is spatially nonuniform across the chip, this leads to "hot spots" and spatial gradients that can cause timing errors or even physical damage. These effects evolve over time scales of hundreds of microseconds or milliseconds. Power-management techniques, in order to be used for thermal management, must directly target the spatial and temporal behavior of operating temperature. In fact, *many low-power techniques have little or no effect on operating temperature*, because they do not reduce power density in *hot spots*, or because they only reclaim slack and do not reduce power and temperature when no slack is present. Temperature-aware design is therefore a distinct albeit related area of study.

Thermal design techniques to date have mostly focused on the thermal package (heat sink, fan, and so on). If the package is designed for worst-case power dissipation, it must be designed for the most severe hot spot that could potentially arise, which is prohibitively expensive. Yet these worst-case scenarios are rare: the majority of applications, especially for the desktop, do not induce sufficient power dissipation to produce the worst-case temperatures. A package designed for the worst case is excessive.

Dynamic thermal management (DTM) allows the thermal package to be designed for power densities exhibited by *typical* applications, with the chip itself adapting if temperatures approach dangerous levels. For typical applications, the less-expensive package still keeps temperatures within specification and DTM is never engaged. If some atypical application causes the processor to run too hot, on-chip sensors detect the thermal stress and engage some form of runtime response, such as dynamic voltage scaling (DVS) or global clock-gating, while trying to minimize any associated performance loss. This response by the chip itself therefore provides the additional cooling and worst-case protection that is needed for reliability, without the associated system cost of a package designed for worst-case behavior. As long as the threshold temperature that engages DTM (the *trigger threshold*) is based on the hottest temperature in the system, this approach successfully regulates temperature. Gunther et al. [2001] reported that designing the thermal package for the "worst typical" application rather than the true worst case, and using DTM in the form of coarse-grained global clock gating, permitted a 20% reduction in the thermal design power for the Pentium 4. For applications that do engage DTM, some performance loss may be incurred, because reducing temperature typically entails reducing power density, which in turn typically entails slower execution. But per-chip savings can be as high as a hundred dollars or more for very high-end, high-power processors and probably in the tens of dollars for laptop systems that require expensive, compact thermal technologies such as heat pipes [Viswanath et al. 2000]. Improving DTM design will allow greater cost savings with minimal performance cost.

The Need for Architecture-Level Thermal Management. While DTM can substantially reduce cooling costs and still allow typical applications to run at peak performance, DTM tends to reduce performance for applications that exceed the thermal design point—sometimes substantially so. Architectural techniques have an essential role to play. *The architecture domain is unique in its ability to use runtime knowledge of application behavior and the current thermal status of different units of the chip to adjust execution, distribute the workload in order to control thermal behavior, and exploit instruction-level parallelism (ILP).* The architecture has detailed temperature information about hot spots and temperature gradients that can be combined with dynamic information about ILP in order to precisely regulate temperature while minimizing performance loss.

In this paper, we describe two new techniques that outperform prior DTM solutions, but both require microarchitecture-level design and analysis, which in turn requires appropriate modeling capability.

The Need for Architecture-Level Thermal Modeling. To accurately characterize current and future thermal stress, temporal and spatial nonuniformities, and application-dependent behavior—let alone evaluate architectural techniques for managing thermal effects—an accurate and practical dynamic model of temperature is needed. As we show in this paper, estimating thermal behavior from some kind of average of power dissipation is unreliable. It can lead to incorrectly estimating the performance impact of thermal effects and even to developing thermal-management techniques that target areas of the chip that are not hot spots.

An effective architecture-level thermal model must be simple enough to allow architects to reason about thermal effects and trade-offs; detailed enough to model runtime changes in temperature within different functional units; and yet computationally efficient and portable for use in a variety of architecture simulators. Finally, the model should be flexible enough to easily extend to novel computing systems that may be of interest from a temperature-aware standpoint.

Contributions. This paper extends our prior work [Skadron 2004; Skadron et al. 2003a; Stan et al. 2003]. We illustrate the importance of thermal modeling and describe a *compact, dynamic, and portable* thermal model for convenient use at the architecture level; use this model to show that hot spots typically occur at the granularity of architecture-level blocks, and that power-based metrics are not well correlated with temperature; and discusses some remaining needs for further improving the community's ability to evaluate temperature-aware techniques. Our model—which we call *HotSpot*—is publicly available at http://lava.cs.virginia.edu/hotspot. Using this model, we evaluate a variety of DTM techniques using a subset of SPEC2000 benchmarks [Standard Performance Evaluation Corporation] and compare them to current techniques such as DVS and fetch gating that have slowdowns of 27–36%. The most effective technique is “temperature-tracking” dynamic frequency scaling: timing errors due to hotspots can be eliminated with an average slowdown of 4.5%, and, if frequency can be changed without stalling computation, less than 3%. For temperature thresholds where preventing physical damage is also a concern, using a spare register file and migrating computation between the register files in response to heating is best, with an average slowdown of 15–19%. If the extra area for a spare register file cannot be entertained, the next-best technique we found is a hybrid technique that combines fetch gating with DVS in order to exploit ILP in conjunction with the stronger power reductions afforded by DVS. Hybrid DTM yields slowdowns of 20–23%. All our experiments include the effects of sensor imprecision, which significantly handicaps runtime thermal management.

The rest of this paper is organized as follows. The next section provides further background and related work. Then Section 3 describes our proposed model and shows the importance of modeling temperature rather than power. Section 4 describes the DTM techniques we evaluate and discuss the role of thermal-sensor nonidealities. Section 5 describes our experimental setup, and then Section 6 compares the various DTM techniques' ability to regulate temperature. Section 7 concludes the paper.

2. BACKGROUND AND RELATED WORK

High temperatures create a number of problems, because transistors can fail to switch properly (this can lead to soft or hard errors), many aging mechanisms are significantly accelerated (e.g., electromigration), which leads to an overall decrease in reliability, and both the die and the package can even suffer permanent damage. Yet accelerating clock rates and microarchitectural innovations lead to steadily increasing power densities. Since carrier mobility is inversely proportional to temperature, operating temperatures cannot rise and may even need to *decrease* in future generations for high-performance microprocessors. The ITRS actually projects that the maximum junction temperature decreases from 95°C for 180 nm to 85°C for 130 nm and beyond. Spatial temperature gradients exacerbate this problem, because the clock speed must typically be designed for the hottest spot on the chip, and information from our industrial partners suggests that temperatures can vary by 30° or more under typical operating conditions. Such spatial nonuniformities arise both because different units on the chip exhibit different power densities, and because localized heating occurs much faster than chip wide heating due to the slow rate of lateral heat propagation. Yet another temperature-related effect is leakage, which increases exponentially with operating temperature. This in turn dissipates additional heat, which in the extreme can even lead to a destructive cycle called thermal runaway.

A wealth of work has been conducted to design new packages that provide greater heat-removal capacity, to arrange circuit boards to improve airflow, and to model heating at the circuit and board (but not architecture) levels. Compact models are the most common way to model these effects, although computational fluid dynamics using finite-element modeling is often performed when details regarding the flow of air or a liquid are considered. An excellent survey of these modeling techniques is given by Sabry in [2002], but they typically depend on low-level VLSI netlists and structural implementation details or only give steady-state solutions.

Despite the long-standing concern about thermal effects, only a few studies have been published so far in the architecture field. Among existing chips, Gunther et al. [2001] describe the thermal design approach for the Pentium 4, where thermal management is accomplished via global clock gating; Fleischmann [2000] describes thermal management in the Transmeta Crusoe, which uses DVS; and Sanchez et al. [1997] describe thermal management in the PowerPC G3, which uses fetch throttling.

Huang et al. [2000] deploy a sequence of four power-reducing techniques—a filter instruction cache, DVS, subbanking for the data cache, and if necessary, global clock gating—to produce an increasingly strong response as temperature approaches the maximum allowed temperature. Brooks and Martonosi [2001] compared several stand-alone techniques for thermal management: frequency scaling, voltage and frequency scaling, fetch gating, decode throttling (similar to fetch throttling), and speculation control. They also point out the value of having a direct microarchitectural thermal trigger that does not require a trap to the operating system and its associated latency. Unfortunately no temperature models of any kind were available at the time, so both papers use a moving average of chip-wide power dissipation as a proxy for temperature. As we show in Section 3.6, this value does not track temperature reliably. A further problem is that at the time, it was unknown which units on the processor ran the hottest, so some of the proposed techniques do not reduce power density in those areas which industry feedback and our own simulations suggest to be the main hot spots, namely the register files, load-store queue, and execution units. For example, we found that the low-power cache techniques are not terribly effective, because caches are not typically the hottest units.

Lim et al. [2002] propose a heterogeneous dual-pipeline processor for mobile devices in which the standard execution core is augmented by a low-power scalar pipeline that shares the fetch engine, register files, and execution units but deactivates out-of-order components such as the renamer and issue queues. The low-power pipeline is primarily intended for applications that can tolerate low performance and hence is effective at saving energy, but this technique can also help whenever the primary pipeline overheats, although with more performance loss than the best techniques described here [Skadron et al. 2003b]. This work used the TEMPEST thermal model [Dhodapkar et al. 2000], which does model temperature directly, but only at the chip level, and no sensor effects are modeled.

In terms of modeling, a chip-wide model such as TEMPEST allows some exploration of chip-wide techniques such as DVS, fetch toggling, and the Pentium 4's global clock gating, but not more localized techniques, yet it cannot capture the effects of hot spots or changing chip layout. Our prior work [Skadron et al. 2002] proposed a simple model for tracking temperature on a per-unit level, and feedback control to modify the Brooks fetch-toggling algorithm to respond gradually, showing a 65% reduction in performance penalty compared to the all-or-nothing approach. Other than HotSpot and TEMPEST, we are unaware of any compact models that have been developed for use in architecture research. No prior work in the architecture field accounts for imprecision due to sensor noise and placement.

Recently, Srinivasan and Adve [2003] and Heo et al. [2003] have used an early version of our RC model [Skadron et al. 2002]. Srinivasan and Adve showed that multimedia programs have properties that permit predictive rather than reactive DTM, and proposed forms of localized DTM that adapt the issue queue and register file size as well as the number of active ALUs. Heo et al. described an algorithm for migrating computation between spare computational units, similar to our MC technique, but with the insight that doing so proactively at a rate faster than the thermal time constant exploits the latency of heating and can be used to help regulate temperature.

This paper demonstrates the importance of a more detailed thermal model that includes localized heating, thermal diffusion, and coupling with the thermal package, uses this model to evaluate a variety of techniques for DTM, and finds two new techniques that outperform classical techniques such as clock gating and DVS.

3. THERMAL MODELING AT THE ARCHITECTURE LEVEL

3.1 Using an Equivalent RC Circuit to Model Temperature

There exists a well-known duality [Krum 2000] between heat transfer and electrical phenomena. Heat flow can be described as a “current” passing through a thermal resistance, leading to a temperature difference analogous to a “voltage.” Thermal capacitance is also necessary for modeling transient behavior, to capture the time required for a mass to change temperature and hence to account for the delay before a change in power results in the temperature’s reaching steady state. Lumped values of thermal R and C can be computed to represent the heat flow among regions of a chip and from each region to the thermal package. The thermal Rs and Cs together lead to exponential rise and fall times characterized by thermal RC time constants analogous to the electrical RC time constants. The rationale behind this duality is that current and heat flow are described by exactly the same differential equations for a potential difference. In the thermal-design community, these equivalent circuits are called *compact models*, and *dynamic compact models* if they include thermal capacitors. This duality provides a convenient basis for an architecture-level thermal model. For a microarchitectural unit, heat conduction to the thermal package and to neighboring units are the dominant mechanisms that determine the temperature.

3.2 A Parameterized, BICI, Dynamic Compact Model for Microarchitecture Studies

For the kinds of studies we propose, the compact model must have the following properties. It must track temperatures at the granularity of individual microarchitectural units, so the equivalent RC circuit must have at least one node for each unit. It must be parametrized, in the sense that a new compact model is automatically generated for different microarchitectures; and portable, making it easy to use with a range of power/performance simulators. It must be able to solve the RC-circuit's differential equations quickly. It must be validated so that simulated temperatures can be expected to correspond to what would be observed in real hardware. Finally, it must be *BICI*, that is, boundary- and initial-condition independent: the thermal model component values should not depend on initial temperatures or the particular configuration being studied. The model we propose, which we call HotSpot, accomplishes these goals by taking advantage of the duality between thermal and electrical systems. Each unit on the chip is modeled as a heat (power) dissipator; each underlying region of silicon is modeled as part of the RC circuit, with several RC elements representing lateral and thermal heat flow; and the package and convection contribute additional RC elements. The model is a simple library that provides an interface for specifying some basic information about the package and for specifying any floorplan of any desired granularity. HotSpot then generates the equivalent RC circuit automatically, and, supplied with power dissipations over any chosen time step, computes temperatures at the center of each block of interest. The model is BICI by construction since the component values are derived only from material, physical, and geometric values.

Chips today are typically packaged with the die placed against a spreader plate, often made of aluminum, copper, or some other highly conductive material, which is in turn placed against a heat sink of aluminum or copper that is cooled by a fan. This is the configuration modeled by HotSpot. Low-power/low-cost chips often omit the heat spreader and sometimes even the heat sink; and mobile devices often use heat pipes and other packaging that avoid the weight and size of a heat sink. These extensions remain areas for future work.

The equivalent circuit—see Figure 1 for an example—is designed to have a direct and intuitive correspondence to the physical structure of a chip and its thermal package. The RC model therefore consists of three vertical, conductive layers for the die, heat spreader, and heat sink, and a fourth vertical, convective layer for the sink-to-air interface. The die layer is divided into blocks that correspond to the microarchitectural blocks of interest and their floorplan. For simplicity, the example in Figure 1 depicts a die floorplan of just three blocks, whereas a realistic model would have 10–20 or possibly even more. The spreader is divided into five blocks: one that corresponds to the area right under the die (R_{sp}), and four trapezoids corresponding to the periphery that is not covered by the die. In a similar way, the sink is divided into five blocks: one corresponding to the area right under the spreader (R_{hs}); and four trapezoids for the periphery. Finally, the convective heat transfer from the package to the air is represented by a single thermal resistance ($R_{\text{convection}}$). Air is assumed to be at a fixed ambient temperature, which is often assumed in thermal design to be 45°C [Mahajan 2002] (this is not the room ambient, but the temperature inside the computer "box"). Because the perspective in Figure 1 makes it somewhat difficult to distinguish vertical and lateral Rs, Figure 2 shows the RC model for only Rs in the die layer. Note that we have chosen not to divide the spreader's center section into blocks corresponding to each block in the die.

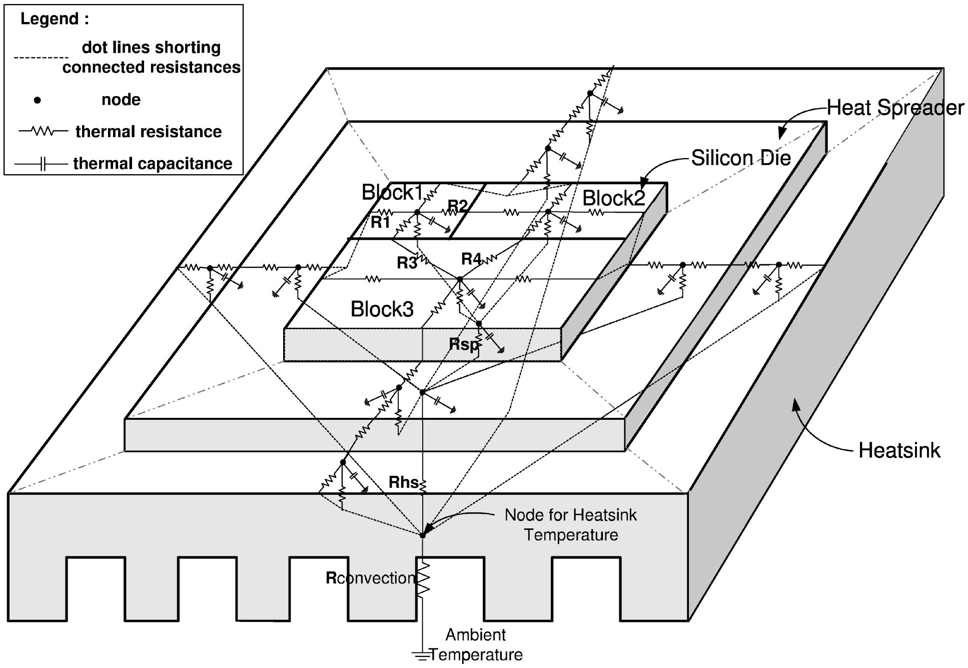


Fig. 1. Example HotSpot RC model for a floorplan with three architectural units, a heat spreader, and a heat sink. The RC model consists of three layers: die, heat spreader, and heat sink. Each layer consists of a vertical RC pair from the center of each block down to the next layer and a lateral RC pair from the center of each edge.

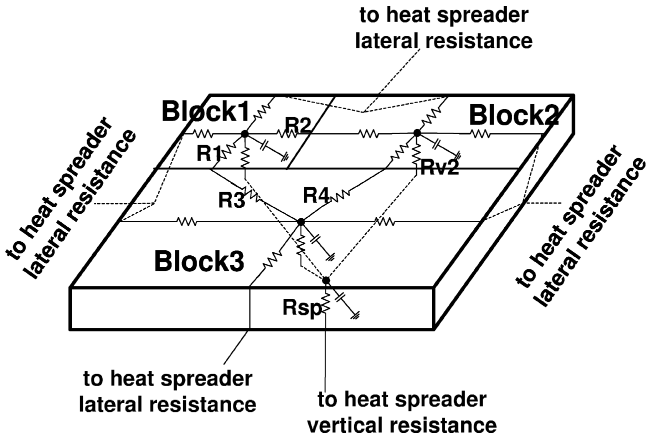

to heat spreader lateral resistance
Block1
Block2
Block3
R1
R2
R3
R4
Rv2
Rsp
to heat spreader lateral resistance
to heat spreader lateral resistance
to heat spreader lateral resistance
to heat spreader vertical resistance

Fig. 2. The RC model for just the die layer.

This is a simplification permitted by the fact that the vertical resistance for the spreader is much less than for the die, so the spreader is relatively isothermal across its surface. Note also that we currently neglect the small amount of heat flowing into the die's insulating ceramic cap and into the I/O pins, and from there into the circuit board, and so on. We also neglect the interface materials between the die, spreader, and sink (these could simply be added as additional vertical layers in the model). Some of these considerations may actually be nontrivial. These are all areas for future work. Perhaps the most important and interesting area for future work is the inclusion of heating due to the clock grid and other interconnect. The effects of wires can be approximated by including their power dissipation in the dynamic, per-block power-density values that drive HotSpot. Our experiments take this approach because the power model we use to drive HotSpot—Wattch [Brooks et al. 2000]—accounts for wires' power this way. A more precise approach would better account for wire lengths and drivers, separately treat self-heating in the wire, and model temperature in each layer of the die.

For the die, spreader, and sink layers, the RC model consists of a vertical model and a lateral model. The vertical model captures heat flow from one layer to the next, moving from the die through the package and eventually into the air. For example, R_{v2} in Figure 2 accounts for heat flow from block 2 into the heat spreader. The lateral model captures heat diffusion between adjacent blocks within a layer, and from the edge of one layer into the periphery of the next area (e.g., $R1$ accounts for heat spread from the edge of block 1 into the spreader, while $R2$ accounts for heat spread from the edge of block 1 into the rest of the chip). At each time step in the dynamic simulation, the power dissipated in each unit of the die is modeled as a current source (not shown) at the node in the center of that block.

3.3 Deriving the Model

In this section we sketch how the values of R and C are computed. The derivation is chiefly based on the fact that thermal resistance is proportional to the thickness of the material and inversely proportional to the cross-sectional area across which the heat is being transferred: $R = t/k \cdot A$, where k is the thermal conductivity of the material per unit volume, 100 $\text{W/m} \cdot \text{K}$ for silicon and 400 $\text{W/m} \cdot \text{K}$ for copper at 85°C. Thermal capacitance, on the other hand, is proportional to both thickness and area: $C = c \cdot t \cdot A$, where c is the thermal capacitance per unit volume, 1.75×10^6 $\text{J/m}^3 \cdot \text{K}$ for silicon and 3.55×10^6 $\text{J/m}^3 \cdot \text{K}$ for copper. Note that HotSpot requires a scaling factor to be applied to the capacitors: capacitors in the spreader and sink should be multiplied by 0.4, and capacitors in the silicon should be multiplied by thickness-dependent factor of about 2–3. These factors are analytical values derived from physical properties and account for some simplifications in our lumped model relative to a full, distributed RC model; the derivation appears in Skadron et al. [2003b]. Typical chip thicknesses are in the range of 0.3–0.9 mm; this paper studies a "thinned" chip of 0.5 mm thickness. Thinning adds to the cost of production but reduces localized hotspots by reducing the amount of relatively higher-resistance silicon and getting the heat-generating areas closer to the higher-conductivity metal package.

In addition to the basic derivations above, lateral resistances must account for *spreading resistance* between blocks of different aspect ratios, and the vertical resistance of the heat sink must account for *constriction resistance* from

Table I. Sample R and C Values. Rs are in K/W, and Cs in J/K and Include the Appropriate Scaling Factor

Block	R_{vert}	R_{lat}	C
IntReg+IntExec	0.420	3.75	0.024
D-Cache	0.250	6.50	0.032
Spreader (Center)	0.025	0.83	0.350
Sink (Center)	0.026	0.50	8.800
Convec.	1.000	na	140.449

the heat-sink base into the fins [Lee et al. 1995]. Spreading and constriction resistances account for the increased heat flow from a small area to a large one and vice versa. These calculations are entirely automated within HotSpot. Again, the derivation appears in Skadron et al. [2003b].

This HotSpot model has several advantages over prior models. In contrast to TEMPEST [Dhodapkar et al. 2000], which models the average chip-wide temperature, HotSpot models heat at a much finer granularity and accounts for hotspots in different functional units. HotSpot has more in common with models from our earlier work [Skadron et al. 2002; Stan et al. 2003], but with some important improvements. Our first work [Skadron et al. 2002] omitted the lateral resistances and the package, both essential for accurate temperatures and time evolution; and used a rather extreme die thickness of 0.1 mm, compressing the spatial temperature gradients too much. Our next model [Stan et al. 2003] corrected these effects but collapsed the spreader and sink, and did not account for spreading resistance, leading to inconvenient empirical fitting factors, and a nonintuitive matching between the physical structure and the model.

Normally, the package-to-air resistance ($R_{\text{convection}}$) would be calculated from the specific heat-sink configuration. In this paper, we instead manually choose a resistance of 1.0 K/W that gives us a good distribution of benchmark behaviors; see Section 5.5. Results with 0.8 K/W are qualitatively similar and have been reported elsewhere [Skadron et al. 2003a, 2003c].

To convey some sense of the R and C values that our model produces, Table I gives some sample values, including those for two different blocks on the die. Vertical Rs for the copper spreader and sink correspond to the middle block, while lateral Rs for the spreader and sink correspond to the inner R for one of the peripheral blocks. As mentioned, the convection resistance of 1.0 K/W has been chosen to provide a useful range of benchmark behaviors, and represents a midpoint in the range 0.1–2.0 that might be found for typical heat sinks [Viswanath et al. 2000] in desktop and server computers. As less expensive heat sinks are chosen—for use with DTM, for example—the resistance increases and temperatures increase, and vice versa.

HotSpot dynamically generates the RC circuit when initialized with a configuration that consists of the blocks' layout and their areas (see Section 3.5). This circuit is then used in a dynamic architectural power/performance simulation by providing dynamic values for power density in each block as the values for the current sources. Power densities are obtained from the power/performance

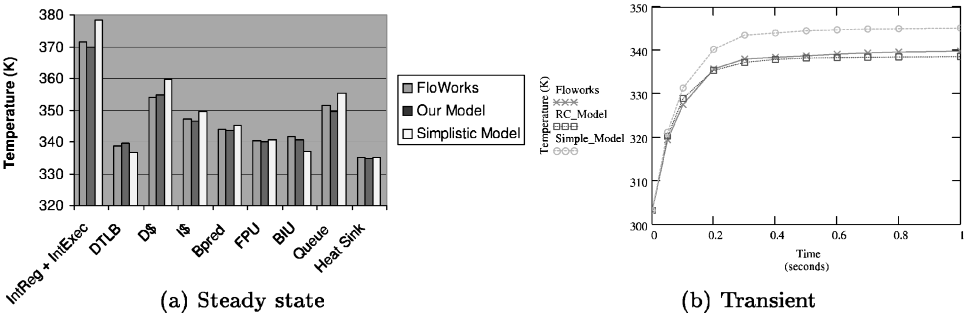
Wattch [Brooks et al. 2000] simulator, averaged over the last 10K clock cycles; see Section 5.1 for more information on the choice of sampling rate. At each time step, the differential equations describing the RC circuit are solved using a fourth-order Runge-Kutta method, returning the new temperature of each block. Each call to the solver takes about 50 μs on a 1.6 GHz Athlon processor, so the simulation-time overhead for temperature modeling is negligible for reasonable sampling rates, less than 1% of total simulation time. The number of iterations for the Runge–Kutta solver is adaptive, to account for the different number of iterations required for convergence at different sampling intervals, an improvement over what was reported in Skadron et al. [2003a], where we used a fixed number of iterations regardless of sampling interval. This slightly improves performance and makes the overhead independent of sampling rate.

HotSpot is already configurable for purposes of modeling new floorplans and variations of the packaging configuration described here, and for use with different simulators and power models. More extensive changes, such as modeling packages with more layers or fans with multiple speed settings, will require minor effort but should remain straightforward.

3.4 Validating the Model

We are not aware of any source of localized, time-dependent measurements of physical temperatures at a microarchitectural granularity that could be used to validate our model. Eventually, we hope to use thermal test chips (e.g., Benedek et al [2001]) or an infrared camera to measure runtime temperatures on a fine temporal and spatial granularity. Until this becomes feasible, we compare against Floworks (http://www.floworks.com), a commercial, finite-element simulator of 3D fluid and heat flow for arbitrary geometries, materials, and boundary conditions that performs full fluid-dynamics calculations, including air flow. A detailed description of our Floworks setup appears in Skadron et al. [2003b].

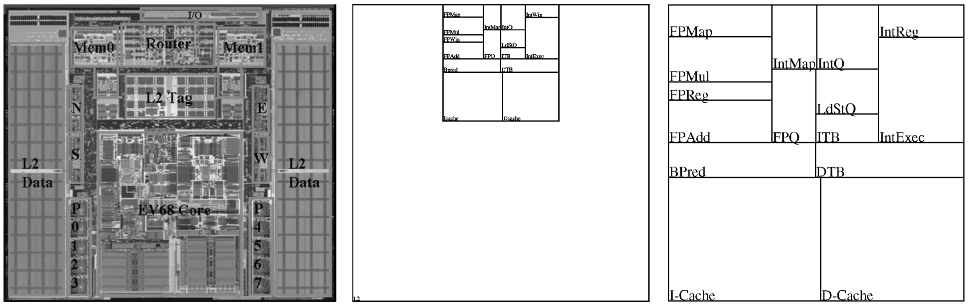
Floworks and HotSpot are not entirely independent. Although all R, C, and scaling factors are determined analytically, without reference to Floworks results; and like HotSpot, our Floworks model omits some details, such as I/O pads and interface materials, which might actually be important. In any case, we can verify that the two obtain similar steady-state operating temperatures and transient response. Figure 3(a) shows steady-state validation comparing temperatures predicted by Floworks, HotSpot, and a “simplistic” model that eliminates the lateral portion of the RC circuit (but not the package, omission of which would yield extremely large errors). HotSpot shows good agreement with Floworks, with errors (with respect to the ambient, 45°C or 318 K) always less than 5.8% and usually less than 3%. The simplistic model, on the other hand, has larger errors, as high as 16%. One of the largest errors is for the hottest block, which means too many thermal triggers will be generated. Figure 3(b) shows transient validation comparing, for Floworks, HotSpot, and the simplistic model, the evolution of temperature in one block on the chip over time. The agreement is excellent between Floworks and HotSpot, but the simplistic model shows temperature rising too fast and too far. Both the steady-state and



(a) Steady state

(b) Transient

Fig. 3. Model validation. (a) Comparison of steady-state temperatures between Floworks, HotSpot, and a "simplistic" model with no lateral resistances. (b) Comparison of the step response in Floworks, HotSpot, and the simplistic model for a single block, the integer register file.



(a) 21364 die photo

(b) Equiv. floorplan

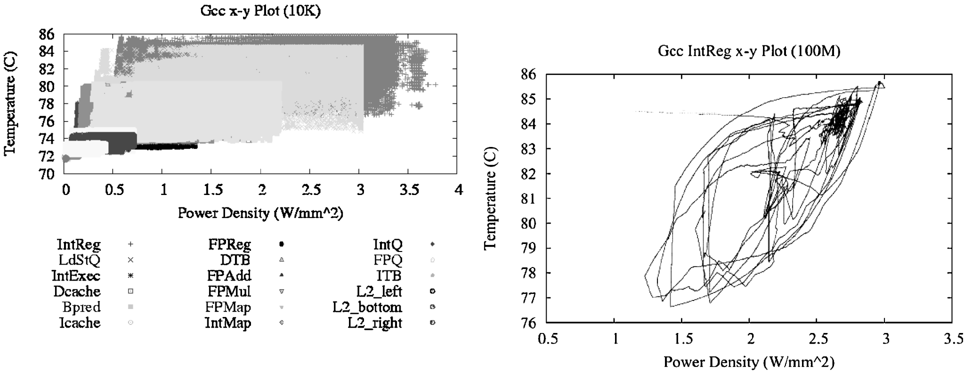
(c) CPU core

Fig. 4. (a) Die photo of the Compaq Alpha 21364 (From CPU Info Center website). (b) Floorplan corresponding to the 21364 that is modeled by our experiments. (c) Closeup of 21364 CPU core.

the transient results show the importance of thermal diffusion in determining on-chip temperatures. We have also validated the scaling factor for the silicon capacitances by testing a 0.1 mm-thick chip with a copper spreader of the same size, and our baseline 0.5 mm chip but with an aluminum (2.58×10^6 J/m$^3 \cdot$ K) spreader of the same size, with similarly small errors. Although we have not yet conducted extensive validations for a wide range of configurations, the HotSpot model is quite general and should give valid results for any reasonable chip thickness, convection resistance, and die/package material.

3.5 Floorplanning: Modeling Thermal Adjacency

The size and adjacency of blocks is a critical parameter for deriving the RC model. In all of our simulations so far, we have used a floorplan (and also approximate microarchitecture and power model) corresponding to that of the Alpha 21364. This floorplan is shown in Figure 4. Like the 21364, it places the CPU core at the center of one edge of the die, with the surrounding area consisting of L2 cache, multiprocessor–interface logic, and so on. Since we model no multiprocessor workloads, we omit the multiprocessor interface logic and treat the entire periphery of the die as second-level (L2) cache. The area of this cache



(a) 10 K

(b) 100 M, I-reg only

Fig. 5. Scatter plots of temperature versus average power density for gcc with averaging intervals of (a) 10K, showing all the major blocks, and (b) 100M cycles, showing only the integer register file.

seems disproportionately large compared to the 21364 die photo in Figure 4(a), because we have scaled the CPU to 130 nm, while keeping the overall die size constant in anticipation of the release of a 130 nm 21364 (we assumed a simple process shrink and kept the die size the same). Note that we do not model I/O pads, because we do not yet have a good dynamic model for their power density—more future work.

Eventually, we envision an automated floorplanning algorithm that can derive areas and floorplans automatically using only the microarchitecture configuration.

3.6 Importance of Directly Modeling Temperature

Due to the lack of an architectural temperature model, a few prior studies have attempted to model temperatures by averaging power dissipation over a window of time. This will not capture any localized heating unless it is done at the granularity of on-chip blocks, but even then it fails to account for lateral coupling among blocks, the role of the heat sink, the nonlinear rate of heating, and so on. We have also encountered the fallacy that temperature corresponds to instantaneous power dissipation, when in fact the thermal capacitance acts as a low-pass filter in translating power variations into temperature variations.

Computing correlation between average-power samples and temperatures, as we did in Skadron et al. [2003a], turns out to overstate the accuracy of power metrics: even though statistical *correlation* is reasonably high for averages over 100 million cycles, power still cannot be used to effectively infer operating *temperature* with any useful precision. Figures 5(a) and (b) present, for two different averaging intervals, scatter plots for each value of average power density versus the corresponding temperature. Although the 100 M interval does show correlation, large temperature ranges (y-axis) are observed for any given value of power density (x-axis). This is partly due to the exponential nature of heating and cooling, which can be observed in the exponentially rising and falling curves. These data come from *gcc*, a representative benchmark, with the reference *expr* input, simulated to completion.

4. TECHNIQUES FOR ARCHITECTURAL DTM

This section describes the various architectural mechanisms for dynamic thermal management that are evaluated in this paper, and then discusses design and modeling considerations for on-chip temperature sensing. The general philosophy behind all but one of these techniques is to react to thermal stress by reducing power density just long enough to bring the temperature to a safe level. The exception is temperature-tracking dynamic frequency scaling (TTDFS), which can be applied when the only risk from excessive temperature is timing errors. In this case it is sufficient to simply reduce the clock speed.

Most of the DTM techniques described below entail minimal extra hardware and should have a negligible effect on power density. The feedback controllers required by some techniques are probably the most complex and require a few registers, an adder, and a multiplier, along with a state machine to drive them, but single-cycle response is not needed, so the controller can be made with minimum-sized circuitry. The datapath width in the controller can also be narrow, since only limited precision is needed. Other techniques require state machines with single-cycle response, but their state space is trivial.

4.1 DTM Techniques

4.1.1 *Temperature-Tracking Dynamic Frequency Scaling.* Independently of the relationship between frequency and voltage, the *temperature* dependence of carrier mobility means that frequency is linearly dependent on the operating temperature. Garrett and Stan [2001] report an 18% variation over the range 0–100°. This suggests that the standard practice of designing the nominal operating frequency for the maximum allowed operating temperature is too conservative. When applications exceed the temperature specification, they can simply scale frequency down in response to the rising temperature. Because the temperature dependence is mild within the interesting operating region, the performance penalty of preventing timing errors is also mild—indeed, negligible.

It might seem odd to only scale frequency. The reason is that the dependence of frequency on temperature is independent of its dependence on voltage: any change in voltage requires an additional reduction in frequency. This means that, unlike traditional DFS, TTDFS does not allow reductions in voltage without further reductions in frequency.

A processor must typically stall for anywhere from 10–50 μs to accommodate resynchronization of the clock's phase-locked loop (PLL), but if the transition is gradual enough, the processor can execute through the change without stalling, as the Xscale is believed to do [Semeraro et al. 2002]. We examine a discretized frequency scaling with 10 MHz steps and 10 μs stall time for every change in the operating frequency; and an ideal version that does not incur this stall but where the change in frequency does not take effect until after 10 μs has elapsed. We call these "TTDFS" and "TTDFS-i(deal)." Larger step sizes do not offer enough opportunity to adapt, and smaller step sizes create too much adaptation and invoke too many stalls. No feedback control is needed for TTDFS, since the frequency is simply a linear function of the current operating temperature.

This technique is unique among our other techniques in that the operating temperature may legitimately exceed the 85° threshold that other techniques must maintain. As long as frequency is adjusted before temperature rises to the level where timing errors might occur, there is no violation.

4.1.2 *Dynamic Voltage Scaling.* DVS has long been regarded as a solution for reducing energy consumption, because it gives cubic reductions in power density relative to performance loss. It has recently been proposed as one solution for thermal management [Brooks and Martonosi 2001; Huang et al. 2000], and is used for this purpose in Transmeta's Crusoe processors [Fleischmann 2000]. When changing the processor voltage, frequency must be reduced in conjunction with voltage, because circuits switch more slowly as the operating voltage approaches the threshold voltage. We used Cadence with BSIM 100 nm low-leakage models to derive appropriate frequency settings for each voltage step [Skadron et al. 2003b].

We model two possible scenarios for the overhead of switching voltage/frequency settings. In the first ("DVS"), the penalty to change the DVS setting is 10 μs, during which the pipeline is stalled. In the second, idealized scenario ("DVS-i"), the processor may continue to execute through the change but the change does not take effect until after 10 μs have elapsed.

Different implementations of DVS offer various numbers of steps for the voltage and frequency, ranging from two with Intel's SpeedStep to at least ten for Transmeta's LongRun, and forty for the Intel Xscale. For thermal management, we found that multiple steps are unnecessary. We tried a variety of step granularities: continuous, ten, five, three, and two (which we call "binary"). For all but binary DVS, we use a proportional-integral (PI) controller to select the highest voltage that regulates temperature.[1] Although multiple step sizes can be beneficial for balancing battery life and performance, for DTM they all give almost exactly the same performance, differing by less than 0.4% for DVS-stall and less than 0.01% for DVS-ideal. These results mean that the PI control reported in our prior work [Skadron et al. 2003a] is unnecessary. Only two voltages are needed for DTM: the maximum voltage, and a low voltage that eliminates all possible thermal violations.

The reasons for this behavior are twofold. First, when more than two voltages are available, safety requires DTM to be conservative, and so the minimum voltage is often used anyway, obviating the benefit of multiple steps. Second, even when multiple steps are available and a higher voltage is used, it takes longer to reduce thermal stress, eliminating the advantage of conferred by the higher frequency; while lower voltages take less time to reduce thermal stress so the lower frequency is used for a shorter time.

[1] When the controller is near a boundary between DVS settings, small fluctuations in temperature can produce too many changes in setting, incurring costly overhead. To prevent this, we apply a simple low-pass filter to decide whether to increase the voltage [Skadron et al. 2003b]. Filtering cannot be used for lowering the voltage, because that is compulsory in response to thermal stress.

Instead of step size, what does matter for DTM is the value of the lowest voltage. With our heat sink and benchmarks, we use 85% of the nominal voltage for the low setting with binary DVS. This is the highest voltage that still eliminates thermal violations.

4.1.3 *Fetch and Clock Gating.* With fetch gating, fetch is prevented at some duty cycle, reducing instruction activity through the pipeline and hence power density. The choice of duty cycle is a feedback-control problem, for which we use a PI controller, with settings confirmed by exhaustive search.

Clock gating might seem more attractive, because it attains extra power reduction by eliminating power dissipation in the clock tree. But stopping and starting the entire clock tree on a rapid basis (required to exploit ILP) may be infeasible, especially given voltage-stability concerns. The mild levels of fetch gating that we employ maintain activity throughout the pipeline and should present less of a voltage-stability problem. To bound the potential extra savings that can be attained with these gating techniques, we show results for both fetch gating (FG) and global clock gating (GCG) in order to provide approximate lower and upper bounds on the power savings possible with a realistic implementation of one of these gating schemes. We refer to these schemes as FG-PI and GCG-PI.

4.1.4 *Local Toggling.* As a possibly more gentle way to implement gating, individual domains of the processor can have their activity gated or “toggled” at the gentlest duty cycle that successfully regulates temperature—“LTOG-PI.” The choice of duty cycle is again a feedback-control problem.

Only domains in thermal stress are toggled. When a unit is toggled in any given cycle, it is prevented from performing its normal activity. Note that this is not a multiple-clock-domain organization; all units operate with the same clock and when a unit is toggled, other units continue operating normally. This means that the units must be decoupled in some fashion, most likely with the queues that already exist in a superscalar processor. The domains that we considered for toggling are the fetch engine, integer engine, floating-point engine, and load-store/data-cache engine. Decoupling buffers between the domains, such as the issue queues, will still dissipate some power even when toggled off, in order to allow neighboring domains to continue operating; for example, allowing the data cache to write back results even though the integer engine is stalled that cycle. Note that as the toggling duty cycle for some domain exceeds available ILP, the entire processor will tend to operate at that duty cycle, and local toggling becomes similar to global fetch gating.

4.1.5 *Migrating Computation.* Two units that run hot by themselves will tend to run even hotter when adjacent. On the other hand, separating them will introduce additional communication latency that is incurred regardless of operating temperature. This suggests the use of spare units located in cold areas of the chip, to which computation can *migrate* only when the primary units overheat.

We developed a new floorplan that includes an extra copy of the integer register file, shown in Figure 6. When the primary register file reaches 81.6°, issue is stalled, instructions ready to write back are allowed to complete, and

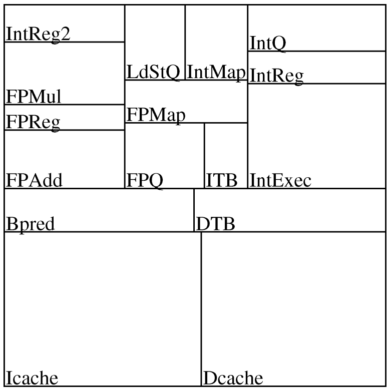


Fig. 6. Floorplan with spare integer register file for migrating computation.

the register file is copied, four values at a time. Then all integer instructions use the secondary register file, allowing the primary register file to cool down, while computation continues unhindered except for the extra computational latency incurred by the greater communication distance. The extra distance is accounted for by charging two extra cycles for every register-file access.[2] When the primary register file returns to 81.5°, the process is reversed and computation resumes using the primary register file. We call this scheme "migrating computation" (MC). Note that, because there is no way to guarantee that MC will prevent thermal violations, a failsafe mechanism is needed, for which we use FG-PI.

Copying values between the register file of course dissipates power, both due to the values that must be copied and to the extra cycles of execution time. Accessing the distant register file, whether for use in computation or to copy values between register files, involves additional power dissipation due to the fact that the signal has to be driven over a longer distance. For each register file read, the issue queue sends register addresses to the register file, and the register file responds to the execution units. Results are then written back to the register file. Assuming that the register file is connected to the issue queue and the execution units via top level metal, we can compute the extra power dissipated in these wires with the formula $aCV^2 f$. We assume an activity factor of 0.5 corresponding to random switching. The capacitance can be computed using Wattch parameters and the wires' length (obtained from our floorplanning tool). For now, we make the simple assumption that the power dissipation in the global interconnect is allocated equally between the sender and receiver (to account for repeaters), yielding 0.003–0.004 W/bit for the processor model and clock frequency modeled in our experiments (see Section 5).

It is also important to note that the different floorplan will have some direct impact on thermal behavior even without the use of any DTM technique. The entire integer engine runs hot, and even if the spare register file is never used,

[2] This is accounted for in the issue logic so that when one operand will be provided via bypassing, the other operand is obtained from the register file early enough so that the register-file operand is ready at the same time as the bypassed operand.

the MC floorplan rearranges the hot units, especially by moving the load-store queue (typically the second- or third-hottest block) farther away from the primary register file. The design space here is rich, but we were limited in the number of floorplans that we could explore, because developing floorplans that fit in a rectangle with no white space is a laborious process. Eventually, we envision an improved floorplanning algorithm that can derive floorplans automatically using some simple specification format.

The dual-pipeline scheme proposed by Lim et al. [2002] could actually be considered another example of migrating computation. MC could also be considered a limited form of multiclustered architecture e.g. [Canal et al. 1999].

4.1.6 *Hybrid DTM.* Finally, we introduce a hybrid technique that combines fetch or clock gating with DVS [Skadron 2004]. The problem with most existing DTM approaches is that different hardware techniques may be better suited to different degrees of thermal stress. This is not just a matter of finding the optimal setting for some technique and matching its response to the degree of thermal stress, such as finding the best voltage and frequency setting that safely cools the chip while minimizing slowdown. Rather, completely different *mechanisms* may be needed. We show that when thermal stress is severe, an aggressive DTM response based on voltage scaling is likely best, because this obtains approximately cubic reductions in power density relative to the reduction in performance. On the other hand, when thermal stress is mild and only a mild DTM response is needed, we show that an architectural response that exploits ILP has less overhead than DVS—possibly even no overhead if sufficient ILP is present. These observations argue for a *hybrid* DTM technique that uses the most effective type of response according to the degree of thermal stress: DVS for aggressive DTM response, and ILP techniques for mild DTM response. Once the required DTM response is aggressive enough that ILP techniques no longer adequately exploit ILP, DVS is engaged. This is the point at which DVS's cubic impact becomes dominant.

We mentioned earlier that DVS by itself does not require multiple settings and feedback control, while FG or GCG by itself does. When combined into a hybrid technique, multiple settings and feedback control can be eliminated for both component techniques with only a 0.5% average loss in performance. We therefore only present results for the simpler scheme ("Hyb") that omits multiple settings and feedback control. This scheme responds to thermal stress by using a fixed duty cycle at full voltage for the ILP-exploiting gating technique (FG or GCG), and when the temperature is too far above the trigger point, lowering the voltage and returning the FG/GCG duty cycle back to 100%. This approach is appealing because it eliminates imprecision, oscillation, and so on that arise with controllers, and because it dovetails with binary DVS.

Composing a hybrid technique requires a way to find the crossover point at which the choice of best technique changes between the "ILP technique" and DVS. To conduct such measurements, we would eventually like a figure of merit that is an a priori measure of cooling, independent of the specific experimental thermal setup. For now, we simply conducted a search across a range of FG and GCG duty cycles.

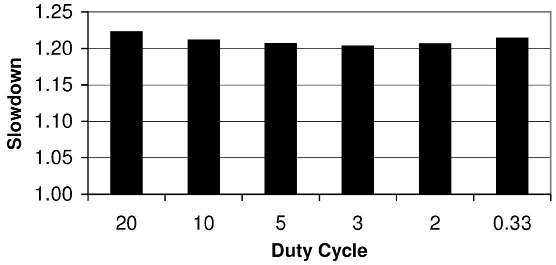


Fig. 7. DTM slowdown for different PI-Hyb configurations with DVS-stall.

As Figure 7 shows, the best hybrid configuration uses a maximum duty cycle of 3 (i.e., skip a fetch cycle once every three cycles) for PI-Hyb with DVS-stall. This figure plots, for PI-Hyb, the slowdown as a function of the duty cycle. A value of x on the x-axis indicates that fetch is gated every $1/x$ cycles, so larger values mean that DVS is engaged sooner. (0.33 means that fetch is gated two out of every three cycles.) A duty cycle of 3 represents the crossover point because beyond this point, it becomes difficult for the ILP technique to successfully exploit instruction-level parallelism, and slowdown rises sharply. In the meantime, beyond this point, DVS's cubic reduction in power density relative to slowdown overcomes the stalls associated with switching settings. In contrast, for the idealized DVS without any stalls, the maximum duty cycle is 20. Because no stalls are incurred to switch DVS settings, only the mildest fetch gating, where ILP hides almost all performance impact, can give better results than DVS.

We performed the same analysis for different low-voltage settings, and with and without the PI controller, and always found the same crossover points. This suggests that the interaction of fetch duty cycle with ILP is purely an architectural phenomenon and remains the same even as the low-voltage varies.

4.2 Sensors

Runtime thermal management requires real-time temperature sensing. So far, all prior published work of which we are aware has assumed omniscient sensors, which we show in Section 6 can produce overly optimistic results. Sensors that can be used on chip for the type of localized thermal response, we contemplate, are typically based on analog CMOS circuits using a current reference. An excellent reference is Bakker and Huijsing [2000]. The output current is digitized using a ring oscillator or some other type of delay element to produce a square wave that can be fed to a counter. Although these circuits produce nicely linear output across the temperature range of interest, and respond rapidly to changes in temperature, they are sensitive to lithographic variations and supply–current variations. These sources of imprecision can be reduced by making the sensor circuit larger, at the cost of increased area and power. Another constraint that is not easily solved by up-sizing is that of sensor bandwidth—the maximum sampling rate of the sensor.

Industry contacts tell us that CMOS sensors which would be reasonable to use in moderate quantity of say 10–20 sensors would have at best a precision of ±2°C and sampling rate of 10 ms. This matches the results in Bakker and Huijsing [2000]. We place one sensor per architectural block.

We model the imprecision by randomizing at each node the true temperature reading over the specified range $\pm 2^\circ$. We assume that the hardware reduces the sensor noises at runtime by using a moving average of the last ten measurements, because averaging reduces the error as the square root of the number of samples. This, of course, assumes that the measured value is stationary, which is not true for any meaningful averaging window. This means we must also account for the potential change in temperature over the averaging window, which we estimate to be potentially as much as 0.4° if temperatures can rise 0.1° per $30\,\mu s$. For $\pm 2^\circ$, we are therefore able to reduce the uncertainty to $S = \frac{2}{\sqrt{10}} + 0.4 = \pm 1^\circ$. An averaging window of ten samples was chosen because the improved error reduction with a larger window is offset by the larger change in the underlying value.

There is one additional nonideality that must be accounted for when modeling sensors and cannot be reduced by averaging. If a sensor cannot be located exactly coincident with every possible hotspot, the temperature observed by the sensor may be cooler by some spatial-gradient factor G than at the hotspot. If, in addition to the random error discussed above, there is also a systematic or offset error in the sensor that cannot be canceled, this increases the magnitude of the fixed error G. Based on simulations in our finite-element model and the assumption that sensors can be located near but not exactly coincident with hotspots, we choose $G = 2^\circ$.

It can therefore be seen that for any runtime thermal-management technique, the use of sensors lowers the emergency threshold by $G + S$ (3° in our case). This must be considered when comparing to other low-power design techniques or more aggressive and costly packaging choices. This extra temperature margin is also strong motivation for finding temperature-sensing techniques that avoid this overhead, perhaps based on clever data fusion among sensors, or the combination of sensors and performance counters.

5. SIMULATION SETUP

In this section, we describe the various aspects of our simulation framework and how they are used to monitor runtime temperatures for the SPECcpu2000 benchmarks [Standard Performance Evaluation Corporation].

5.1 Integrating the Thermal Model

HotSpot is completely independent of the choice of power/performance simulator. Adding HotSpot to a power/performance model merely consists of two steps. First, initialization information must be passed to HotSpot. This consists of an adjacency matrix describing the floorplan (the floorplan used for the experiments in this paper is included in the HotSpot release) and an array giving the initial temperatures for each architectural block. Then at runtime, the power dissipated in each block is averaged over a user-specified interval and passed to HotSpot's RC solver, which returns the newly computed temperatures.

Although it is feasible to recompute temperatures every cycle, this is wasteful, since even at the fine granularity of architectural units, temperatures take at least 100K cycles to rise by 0.1°C. We chose a sampling rate of 10K cycles as the best trade-off between precision and overhead. For sampling intervals of 10K and less, the error is less than 0.01°—the same magnitude as the rounding error due to significant digits in the Runge–Kutta solver.

5.2 Power-Performance Simulator

We use a power model based on power data for the Alpha 21364 [Bannon 2002]. The 21364 consists of a processor core identical to the 21264, with a large L2 cache and (not modeled) glueless multiprocessor logic added around the periphery. An image of the chip is shown in Figure 4, along with the floorplan schematic that shows the units and adjacencies that HotSpot models. Because we study microarchitectural techniques, we use Wattch version 1.02 [Brooks et al. 2000], which is based on a widely used and convenient cycle-accurate microarchitecture simulator, SimpleScalar [Burger and Austin 1997]. Our power data for were 1.6 V at 1 GHz in a 0.18μ process, so we used Wattch's linear scaling to obtain power for 0.13μ, V_{dd}=1.3 V, and a clock speed of 3 GHz. These values correspond to the recently announced operating voltage and clock speed that for the Pentium 4 [Robertson 2002]. We assume a die thickness of 0.5 mm. Our spreader and sink are both made of copper. The spreader is 1 mm thick and 3 cm × 3 cm, and the sink has a base that is 7 mm thick and 6 cm × 6 cm. Power dissipated in the per-block temperature sensors and the DTM control logic are not modeled.

We augmented SimpleScalar's sim-outorder to model an Alpha 21364 as closely as possible, with four-wide integer and two-wide floating-point issue queues, an 80-entry integer and floating-point merged physical/architectural register file, and an 80-entry active list. First-level caches are 64 KB, 2-way, write-back, with 64 B lines and a 2-cycle latency; the second-level is 4 MB, 8-way, with 128 B lines and a 12-cycle latency; and main memory has a 225-cycle latency. The only major features of the 21364 that we do not model are the register-cluster aspect of the integer box, way prediction in the I-cache, and speculative load-use issue with replay traps (which may increase power density in blocks that are already quite hot). We also modified SimpleScalar/Wattch to account for dynamic frequency and voltage scaling and to report execution time in seconds rather than cycles as the metric of performance. Finally, we augmented the issue logic to properly handle the extra latency to MC's spare register file, by taking advantage of bypassing when possible and otherwise waking instructions up early enough to read the register file early enough to be ready when other operands become available.

5.3 Modeling the Temperature-Dependence of Leakage

Because leakage power is an exponential function of temperature, these power contributions may be large enough to affect the temperature distribution and the effectiveness of different DTM techniques. Furthermore, leakage is present regardless of activity, and leakage at higher temperatures may affect the efficacy of thermal-management techniques that reduce only activity rates. Eventually, we plan to combine HotSpot with our temperature/voltage-aware "HotLeakage" model [Li et al. 2004] to more precisely track dynamic leakage-temperature interactions, which we believe are an interesting area for future work. For now, to make sure that leakage effects are modeled in a reasonable way, we use a simpler model: like Wattch, leakage in each unit is simply treated as a percentage of its power when active, but this percentage is now determined based on the temperature and technology node using figures from ITRS data [2001].

The original model assumes that idle architectural blocks are clock gated and leak a fixed percentage of the dynamic power that they would dissipate if active. The default value of 10% happens to roughly correspond to the leakage percentage that would be seen at 85°C in the 130 nm generation, according to our calculations from the ITRS projections. To better incorporate leakage effects in a simple way, we improve Wattch's model to account for the temperature dependence of leakage. We use ITRS data [SIA 2001] to derive an exponential distribution for the ratio R_T of leakage power to dynamic power as a function of temperature T, and recompute the leakage ratio at every time step. This gives

$$R_T = \frac{R_0}{V_0 T_0^2} e^{\frac{B}{T_0}} \cdot V T^2 \cdot e^{\frac{-B}{T}} \tag{1}$$

where T_0 is the ambient temperature and R_0 is the ratio at T_0 and nominal voltage V_0. B is a process technology constant that depends on the ratio between the threshold voltage and the subthreshold slope. This ratio was computed using the leakage current and saturation drive current numbers from ITRS 2001. Only the $VT^2 \cdot e^{\frac{-B}{T}}$ term varies with temperature and/or operating voltage. This expression replaces the fixed leakage factor in the original Wattch.

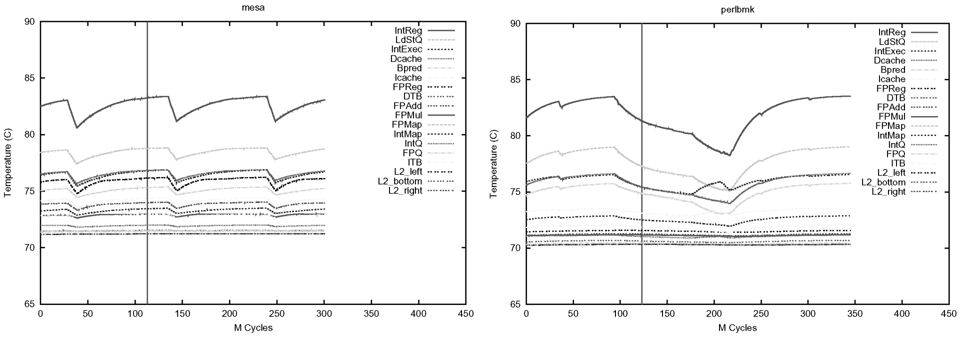
It is desirable to eventually model leakage in more detail, to account for structural details, and permit studies of the interactions between temperature and leakage-management techniques. The interaction of leakage energy, leakage control, and thermal control is beyond the scope of this paper, but is clearly an interesting area for future work, and since we do not study leakage-control techniques here, the simpler temperature-dependent function given in Equation (1) seems adequate for our current work.

5.4 Benchmarks

We evaluate our results using benchmarks from the SPECcpu2000 suite. 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 but no operating system or multiprogrammed behavior. For each program, we fast-forward to a single representative sample of 500 million instructions. The location of this sample is chosen using the data provided by Sherwood et al. [2001]. Simulation is conducted using SimpleScalar's EIO traces to ensure reproducible results for each benchmark across multiple simulations.

Table II. Benchmark Summary, Heat Sink R = 1.0 K/W. "I" = Integer, "F" = Floating Point

	IPC	Average Power (W)	% Cycles in Thermal Violation	Dynamic Max Temp. (°C)
Severe Thermal Stress (medium)				
mesa (F)	2.7	31.5	100.0	90.9
perlbmk (I)	2.3	30.4	100.0	94.3
gzip (I)	2.3	31.0	100.0	90.9
bzip2 (I)	2.3	31.7	100.0	93.8
Extreme Thermal Stress (hot)				
eon (I)	2.3	33.2	100.0	92.0
crafty (I)	2.5	31.8	100.0	92.0
vortex (I)	2.6	32.1	100.0	91.4
gcc (I)	2.2	32.2	100.0	93.2
art (F)	2.4	38.1	100.0	95.6



(a) mesa

(b) perlbmk

Fig. 8. Operating temperature as a function of time (in terms of millions of clock cycles) for two example benchmarks.

Due to the extensive number of simulations required for this study and the fact that many did not run hot enough to be interesting thermally, we used only nine of the total 26 SPEC2k benchmarks. A mixture of integer and floating-point programs with intermediate, and extreme thermal demands were chosen. Table II provides a list of the benchmarks we study along with their basic performance, power, and thermal characteristics. It can be seen that IPC and peak operating temperature are only loosely correlated with average power dissipation. For most SPEC benchmarks, and all those in Table II, the hottest unit is the integer register file—interestingly, this is even true for most floating-point and memory-bound benchmarks. It is not clear how true this will be for other benchmark sets.

For the benchmarks that have multiple reference inputs, we chose one. For *perlbmk*, we used splitmail.pl with arguments "957 12 23 26 1014"; *gzip*—graphic; *bzip2*—graphic; *eon*—rushmeier; *vortex*—lendian3; *gcc*—expr; and *art*—the first reference input with "-startx 110."

To more clearly illustrate the time-varying nature of programs' thermal behavior, in Figure 8 we present two plots of programs' operating temperature (with no DTM) in each unit as a function of time. In each plot, the vertical line toward the left-hand side of the plot indicates when the warmup period ends.

Mesa (Figure 8(a)) deserves special comment because it shows clear program phases. At each drop in its sawtooth curve, we found (not shown) a matching sharp rise in L1 and L2 data misses and a sharp drop in branch mispredictions. The rate of rise and fall exactly matches what we calculate by hand from the RC time constants. The temperatures are only varying by a small amount near the top of their range. So the increase in temperature occurs slowly, like a capacitor that is already close to fully charged, and the decrease in temperature is quite sharp, like a full capacitor being discharged. *Perlbmk* (Figure 8(b)) shows a smaller version of this periodic sawtooth behavior.

5.5 Package, Warmup, and Initial Temperatures

The correct choice of convection resistance and heat-sink starting temperature are two of the most important determinants of thermal behavior over the relatively short time scales than can be tractably simulated using SimpleScalar. We chose a low-cost heat sink whose convection resistance we compute to be 1.0 K/W. This represents a medium-cost heat sink, with a modest savings of perhaps $10 [Viswanath et al. 2000] compared to the 0.7 K/W convection resistance that would be needed without DTM. This is larger than the resistance of 0.8 K/W that we chose manually in Skadron et al. [2003a], but has the advantage that it corresponds to a computed model of a real heat sink and fan configuration. The relative performance of the various DTM schemes is mostly unchanged, whether 0.8 or 1.0 K/W is used.

The initial temperatures that are set at the beginning of simulation play a large role in thermal behavior. The most important temperature is that of the heat sink. Its time constant is on the order of several minutes, so its temperature barely changes and certainly does not reach steady state in our simulations. For experiments with DTM (except TTDFS), the heat-sink temperature should be set to a value commensurate with the maximum tolerated die temperature (81.8° with our sensor architecture): the DTM response ensures that chip temperatures never exceed this threshold, and heat sink temperatures are correspondingly lower than with no DTM. We have not yet accounted for multi programmed behavior: a “hot” application that begins executing when the heat sink is cool may not generate thermal stress before its time slice expires. Rohou and Smith [1999] used this to guide processor scheduling and reduce maximum operating temperature. Indeed, although we have not explored scheduling policy, the fact that temperatures evolve on time scales of milliseconds suggests that this is another interesting area for future work for which HotSpot should be useful.

To avoid transient artifacts as on-chip structures reach representative temperatures, it is necessary to warm up the state of large structures such as caches and branch predictors, and then to literally warm up HotSpot. We first set the blocks’ initial temperatures to the steady-state temperatures calculated using the per-block average power dissipation for each benchmark. When we start simulations, we run the simulations in full-detail cycle-accurate mode

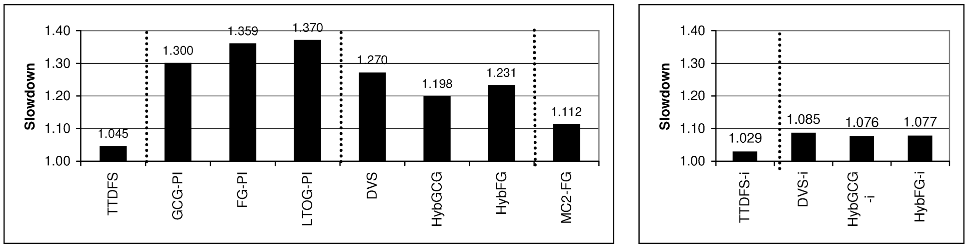


Fig. 9. Slowdown for DTM. The left-hand graph presents techniques that do not depend on "ideal" voltage/frequency scaling, and the right-hand graph presents techniques that assume ideal scaling.

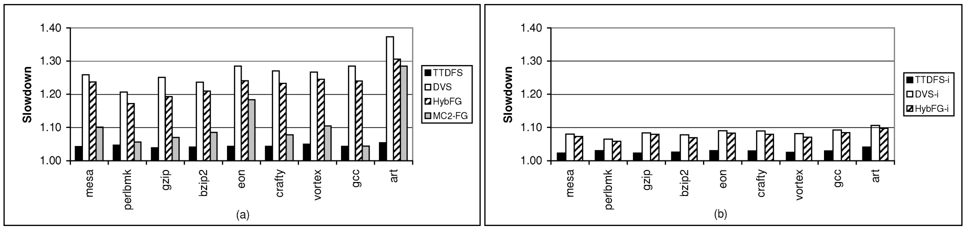


Fig. 10. Per-benchmark slowdowns for selected DTM techniques.

(but without statistics gathering) for 100 million cycles to train the caches—including the L2 cache—and the branch predictor. (These two issues must be treated sequentially, because otherwise cold-start cache effects would affect temperatures.) Finally, we continue in full-detail cycle-accurate mode for another 200 million cycles to allow temperatures to reach truly representative values. Only after these warmup phases have completed do we begin to track any experimental statistics.

6. RESULTS FOR DTM

In this section, we use the HotSpot thermal model to evaluate the performance of the various techniques described in Section 4. First we assume realistic, noisy sensors, and then consider how much the noise degrades DTM performance.

6.1 Results with Sensor Noise Present

Figure 9 presents the slowdown (execution time with thermal management divided by original execution time) for each of the thermal management techniques, averaged across all the benchmarks. None of the techniques incur any thermal violations (Figure 10). All the average performance differences compared to the baseline are significant at the 99% confidence level.

6.1.1 *TTDFS versus Threshold-Based Techniques.* The best technique for thermal management by far is TTDFS, with the TTDFS-i version of course being slightly better. The performance penalty for even the hottest benchmarks is small; the worst is *art* with only a 4.1% slowdown for TTDFS-i and a 5.4% slowdown for TTDFS. If the maximum junction temperature of 85° is strictly based on timing concerns, and somewhat higher temperatures than 85° can be tolerated without unduly reducing operating lifetime, then TTDFS is vastly superior because its impact is so gentle. Even if other factors (e.g., different technology parameters) make the temperature dependence of frequency larger, TTDFS remains an attractive solution.

It might seem there should be some benefit with TTDFS from *increasing* frequency when below the trigger threshold, but we did not observe any noteworthy speedups—even for TTDFS-i with the coldest benchmark, *mcf*, we observed only a 2% speedup, and the highest speedup we observed was 3%. A few benchmarks actually experienced a 1% slowdown, and the highest speedup we observed was 2%. The reason for the lack of speedup is partly that the slope is so small—this helps to minimize the slowdown for TTDFS with warm and hot benchmarks, but minimizes the benefit for cold ones. In addition, for higher frequency to provide significant speedup, the application must be CPU bound, but then it will usually be hot and frequency cannot be increased.

If the junction temperature is dictated not only by timing but also physical reliability, then TTDFS is not a viable approach: the specified junction temperature must be enforced. All the remaining techniques do this, and among these, the techniques using ideal DVS (DVS-ideal and the two hybrid-ideal techniques) are vastly superior. If changing frequency entails a 10 μs stall, only the left-hand graph in Figure 9 applies, and MC and the two hybrid techniques are best.

6.1.2 *Gating and Toggling.* The gating/toggling techniques—GCG, FG, and LTOG—all perform much worse than the other techniques. Although in principle these techniques should capitalize on ILP, this only succeeds at mild duty cycles where enough instructions are active to maintain throughput. Beyond duty cycles of about 1/3, slowdown becomes proportional to the duty cycle—and these more aggressive duty cycles are often required. In contrast, DVS and the hybrid techniques can take advantage of the fact that DVS allows cubic reductions in power density relative to performance loss. The hybrid techniques can still exploit ILP when only mild duty cycles are needed.

Note that LTOG actually performs worse than the two global gating techniques. This contradicts the results in Skadron et al. [2003a]. We found that the gain on the GCG and FG controllers was too high, engaging unnecessarily strong duty cycles. The reason that LTOG is no better is that for all but the very mildest of duty cycles, toggling one domain tends to proportionally reduce the throughput of the entire pipeline; but these mild duty cycles are almost never seen. The reason that FG is slightly better than LTOG—rather than simply being equivalent—is that FG's immediate chip-wide impact does a slightly better job of cooling down domains neighboring the hot one that triggered the gating. This also means that subsequent heating takes slightly longer too.

GCG outperforms FG and LTOG because it eliminates power dissipation in the clock tree and allows the chip to cool faster.

6.1.3 *Hybrid versus DVS.* Although DVS provides cubic reductions in power density relative to performance loss, the hybrid techniques outperform

DVS because they capitalize on the gating component's ability to exploit ILP and reduce stalls to change the DVS setting. Hybrid DTM can improve performance by 5.5–6% compared to DVS alone, which represents about a 25% reduction in DTM overhead. When an idealized DVS is available with no overhead to change settings, hybrid DTM is less helpful, improving performance by only about 1% compared to DVS-i, which represents about an 11% reduction in DTM overhead. Hybrid DTM is still able to attain a small benefit even with ideal DVS because DVS almost always imposes some performance overhead due to the use of a lower frequency. This means that, even though hybrid DTM with ideal DVS uses a very low crossover and the ILP technique is rarely used, there are occasional mild temperature excursions where the ILP technique can still do a better job of reducing temperature with less performance loss. It is not clear whether this advantage is significant enough to be worth the effort. But with the overheads found in typical DVS implementations today, hybrid DTM does offer impressive benefits.

Another important point is that both DVS and hybrid techniques eliminate the need for feedback control to select a duty cycle, eliminating a major source of difficulty in tuning and making the DTM technique more robust. We did also evaluate hybrid techniques with PI control to set the duty cycle on the gating component, and found that it conferred no benefit. The explanation is the same as for DVS's insensitivity to the number of voltage steps: less aggressive fetch gating takes longer to reduce the temperature, and vice versa. It might seem that this argument should apply to FG, GCG, and LTOG too as standalone techniques, but here the PI control is needed. If only one FG/GCG/LTOG duty cycle were available, it would have to be too high—a duty cycle of 50% for FG—to eliminate all thermal violations; and this is beyond the ILP-DVS crossover point. Eliminating PI control for DVS works because its cubic nature compensates for using a low voltage, and eliminating PI control for hybrid DTM works because the fixed ILP response can be matched to the crossover point.

6.1.4 *MC.* Except for the ideal DVS and ideal hybrid schemes, MC is the best DTM technique that ensures physical reliability. MC works well for three reasons. First, the floorplan we used by itself is enough to reduce the operating temperature of the primary integer register file. This shows the importance of considering on-chip thermal diffusion in designing the floorplan. Second, MC is able to exploit instruction-level parallelism. Third, the complete elimination of activity in the primary register file allows it to cool quickly, minimizing the time during which the slower secondary register file is needed.

It is interesting to note that MC alone is not able to prevent all thermal violations. Our MC technique engaged the fallback technique for four of the benchmarks, *gzip, bzip2, gcc*, and *art*: 6–9% of the time for *gzip* and *bzip* and 90–100% of the time for *gcc* and *art*. Even for these benchmarks where the fallback technique is used, performance with MC–FG is better than with the toggling, DVS, or hybrid techniques, so the migration does provide clear value added. This is even true for *art*. Yet it also means that the choice of fallback technique can be important to performance. Compared to GCG or FG as the fallback, results are much worse if we use DVS as the fallback, again because of the overhead of stalls required when engaging DVS. Ideal DVS does not have this problem, and performs slightly better as a fallback than GCG or FG.

It might be that issue logic which can adapt to the differing latencies of the primary and secondary register files would be too complex. If the secondary register file is not accessed early to account for its extra latency, the performance overhead rises from 11.2% to 18.8%.

Other floorplans that accommodate the spare register file may give different results, and spare copies of other units may be useful as well, especially for programs that cause other hotspots. We have not yet had a chance to explore these issues. Another study we have not had a chance to perform is the cost-benefit analysis of whether the extra die area for the spare register file would be better used for some other structure, with hybrid DTM as the thermal-management technique. But these results and recent work by Heo et al. [2003] suggest the potential of trading-off area for thermal control. These results suggest that migration and explicit consideration of floorplan issues are promising ways to manage heat. And once alternate floorplans and extra computation units are contemplated, the interaction of performance and temperature for microarchitectural clusters e.g. [Canal et al. 1999] becomes an interesting area for further investigation. Our MC results also suggest the importance of modeling lateral thermal diffusion.

6.1.5 *Summary of DTM Behavior.*. The results reported here clearly show the value of DTM techniques that can exploit ILP. Both the Hyb and MC techniques obtain substantial benefits from ILP, although MC requires extra die area for a spare copy of the register file. New techniques for more effectively extracting ILP and for migrating computation are promising areas for future work. Our results also show the drawback of using DVS unless the associated stalls can be drastically reduced.

6.2 Role of Sensor Error

Sensor noise hurts in two ways; it generates spurious triggers when the temperature is actually not near violation, and it forces a lower trigger threshold. Both reduce performance. Figure 11 shows the impact of both these effects for our DTM techniques (for TTDFS and DVS, we look at the nonideal versions) in terms of what percentage of each techniques slowdown can be attributed to sensor effects. Sensor error clearly has a significant impact on the effectiveness of thermal management. MC, DVS, Hyb, and TTDFS are the most sensitive to sensor noise because each incurs significant overhead for changes in DTM setting—overhead that is wasted on spurious changes caused by noise. On the other hand, even though TTDFS is sensitive to noise, the impact of the different threshold on TTDFS was negligible. That is because the TTDFS change in frequency for 1° is negligible.

Overall, these results also indicate not only the importance of modeling temperature in thermal studies, but also the importance of modeling realistic sensor behavior and of finding new ways to determine on-chip temperatures more precisely.

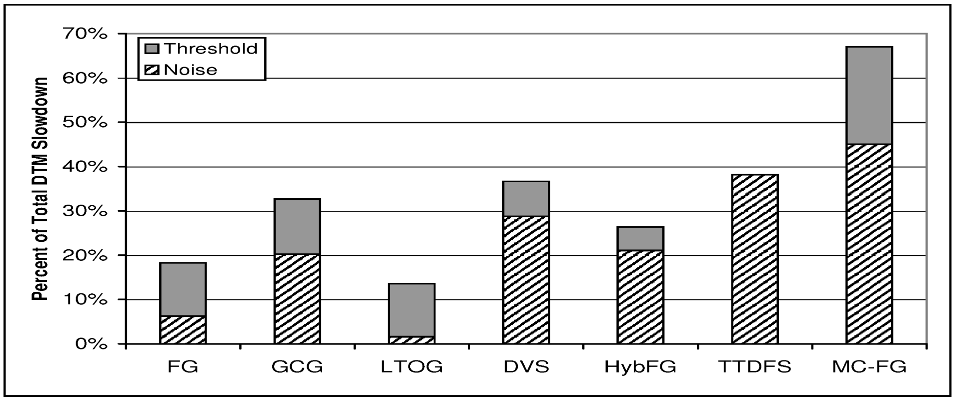


Fig. 11. Slowdown for DTM from eliminating sensor noise, and from the consequent increase in trigger threshold to 82.8°.

7. CONCLUSIONS AND FUTURE WORK

This paper has presented HotSpot, a practical and computationally efficient approach to modeling thermal behavior in architecture-level power/performance simulators. Our technique is based on a simple network of thermal resistances and capacitances that have been combined to account for heating within a block due to power dissipation, heat flow among neighboring blocks, and heat flow into the thermal package. The model has been validated against finite-element simulations using Floworks, a commercial simulator for heat and fluid flow. HotSpot is publicly available at http://lava.cs.virginia.edu/hotspot.

Using HotSpot, we can determine which are the hottest microarchitectural units; understand the role of different thermal packages on architecture, performance, and temperature; understand programs' thermal behavior; and evaluate a number of techniques for regulating on-chip temperature. When the maximum operating temperature is dictated by timing and not physical reliability concerns, "temperature-tracking" frequency scaling lowers the frequency when the trigger temperature is exceeded, with average slowdown of only 4.5%, and only 3% if the processor need not stall during frequency changes. When physical reliability concerns require that the temperature never exceed a specified value—85° in our studies—the best solutions we found were to provide a second register file and migrate computation between them (11.2% slowdown) and a hybrid technique that combines fetch gating with DVS (23.1% slowdown, and only 7.7% slowdown if no stalls are needed during frequency changes). These schemes perform substantially better than global clock gating, fetch gating, and DVS—which exhibit slowdowns of 27–36%—because they exploit the ability of instruction-level parallelism to hide small disruptions in instruction flow.

A significant portion of the performance loss of all these schemes is due to sensor error, which invokes thermal management unnecessarily. Even with a mere ±1° noise margin, sensor error was responsible for as much as 67% of the slowdowns observed.

We feel that these results make a strong case that runtime thermal management is an effective tool in managing the growing heat dissipation of processors, and that microarchitecture DTM techniques must be part of any temperature-aware system. But to obtain reliable results, architectural thermal studies must evaluate techniques based on *temperature* and must include the effects of sensor noise as well as lateral thermal diffusion.

We hope that this paper conveys an overall understanding of thermal effects at the architecture level, and of the interactions of microarchitecture, power, sensor precision, temperature, and performance. This paper only touches the surface of what we believe is a rich area for future work. The RC model must still be validated against test chips, and can be refined in many ways, most importantly to include the effects of heating in the metal layers due to wire self-heating, heating in the I/O pads, the role of thermal interface layers in the thermal package, and the impact of more sophisticated cooling techniques such as heat pipes and variable-speed fans. The model can also be extended to multiprocessor, chip-multiprocessor, and simultaneous multithreaded systems; many new workloads remain to be explored; and a better understanding is needed for how programs' execution characteristics and microarchitectural behavior determine their thermal behavior. In terms of techniques for improving runtime thermal management, many new DTM techniques are possible. Our results suggest that the most effective ones will exploit ILP as well as the choice of floorplan. Clever data-fusion techniques for sensor readings are also needed to allow more precise temperature measurement and reduce sensor-induced performance loss. In addition to work in the microarchitecture domain, temperature-awareness is also an interesting research area for other components of a computer system as well as for scheduling policies within the operating system. Finally, another important problem is to understand the interactions among dynamic management techniques for active power, leakage power, current variability, and thermal effects, which together present a rich but poorly understood design space where the same technique may possibly be used for multiple purposes but at different settings and points in time.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grant nos. CCR-0133634 and MIP-9703440, a grant from Intel MRL, and an Excellence Award from the University of Virginia Fund for Excellence in Science and Technology. We would also like to thank Peter Bannon, Pradip Bose, David Brooks, Howard Davidson, Antonio González, Jose González, Margaret Martonosi, Avi Mendelson, Ronny Ronen, and Gad Sheaffer for their helpful comments.

REFERENCES

BAKKER, A. AND HUIJSING, J. 2000. *High-Accuracy CMOS Smart Temperature Sensors*. Kluwer Academic, Boston, MA.

BANNON, P. 2002. Personal communication.

BENEDEK, Z., COURTOIS, B., FARKAS, G., KOLLÁR, E., MIR, S., POPPE, A., RENCZ, M., SZÉKELY, V., AND TORKI, K. 2001. A scalable multi-functional thermal test chip family: Design and evaluation. *Transactions of the ASME, Journal of Electronic Packaging 123,* 4 (Dec.), 323–330.

BORKAR, S. 1999. Design challenges of technology scaling. *IEEE Micro 19,* 4 (Jul.–Aug.), 23–29.

Brooks, D. and Martonosi, M. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, 171–182.

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

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

Canal, R., Parcerisa, J.-M., and González, A. 1999. A cost-effective clustered architecture. In *Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques*, 160–168.

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

Fleischmann, M. 2000. Crusoe power management: Cutting ×86 operating power through LongRun. In *Embedded Processor Forum*.

Garrett, J. and Stan, M. R. 2001. Active threshold compensation circuit for improved performance in cooled CMOS systems. In *Proceedings of the International Symposium on Circuits and Systems*, 410–413.

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

Heo, S., Barr, K., and Asanovic, K. 2003. Reducing power density through activity migration. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*.

Huang, W., Renau, J., Yoo, S.-M., and Torellas, J. 2000. A framework for dynamic energy efficiency and temperature management. In *Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture*, 202–213.

Krum, A. 2000. Thermal management. In *The CRC Handbook of Thermal Engineering*, F. Kreith, Ed. CRC Press, Boca Raton, FL, 2.1–2.92.

Lee, S., Song, S., Au, V., and Moran, K. 1995. Constricting/spreading resistance model for electronics packaging. In *Proceedings of the ASME/JSME Thermal Engineering Conference*, 199–206.

Li, Y., Parikh, D., Zhang, Y., Sankaranarayanan, K., Skadron, K., and Stan, M. 2004. State-preserving vs. non-state preserving leakage control in caches. In *Proceedings of the 2004 Design, Automation and Test in Europe Conference*, to appear.

Lim, C.-H., Daasch, W., and Cai, G. 2002. A thermal-aware superscalar microprocessor. In *Proceedings of the International Symposium on Quality Electronic Design*, 517–522.

Mahajan, R. 2002. Thermal management of CPUs: A perspective on trends, needs and opportunities. Keynote presentation at the *8th Int'l Workshop on THERMal INvestigations of ICs and Systems*.

Robertson, J. 2002. Intel hints of next-generation security technology for mpus. *EE Times*.

Rohou, E. and Smith, M. 1999. Dynamically managing processor temperature and power. In *Proceedings of the 2nd Workshop on Feedback-Directed Optimization*.

Sabry, M.-N. 2002. Dynamic compact thermal models: An overview of current and potential advances. In *Proceedings of the 8th Int'l Workshop on THERMal INvestigations of ICs and Systems*. Invited paper.

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

Semeraro, G., Magklis, G., Balasubramonian, R., Albonesi, D. H., Dwarkadas, S., and Scott, M. L. 2002. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, 29–40.

Sherwood, T., Perelman, E., and Calder, B. 2001. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, 3–14.

SIA 2001. *International Technology Roadmap for Semiconductors*. SIA.

Skadron, K. 2004. Hybrid architectural dynamic thermal management. In *Proceedings of the 2004 Design, Automation and Test in Europe Conference*, to appear.

SKADRON, K., ABDELZAHER, T., AND STAN, M. R. 2002. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, 17–28.

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003a. Temperature-aware microarchitecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, 2–13.

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003b. Temperature-aware microarchitecture: Extended discussion and results. Tech. Rep. CS-2003-08, University of Virginia Department of Computer Science. Apr.

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003c. Temperature-aware computer systems: Opportunities and challenges. *IEEE Micro 23,* 6 (Nov.–Dec.), 52–61.

SRINIVASAN, J. AND ADVE, S. V. 2003. Predictive dynamic thermal management for multimedia applications. In *Proceedings of the 2003 International Conference on Supercomputing*, 109–120.

STAN, M. R., SKADRON, K., BARCELLA, M., HUANG, W., SANKARANARAYANAN, K., AND VELUSAMY, S. 2003. Hotspot: A dynamic compact thermal model at the processor-architecture level. *Microelectronics Journal: Circuits and Systems 34,* 12 (Dec.), 1153–1165.

STANDARD PERFORMANCE EVALUATION CORPORATION. SPEC CPU2000 Benchmarks. http://www.specbench.org/osg/cpu2000.

VISWANATH, R., VIJAY, W., WATWE, A., AND LEBONHEUR, V. 2000. Thermal performance challenges from silicon to systems. *Intel Technology Journal*.

Received October 2003; revised January 2004; accepted January 2004