TEMPERATURE-AWARE COMPUTER SYSTEMS: OPPORTUNITIES AND CHALLENGES

TEMPERATURE-AWARE DESIGN TECHNIQUES HAVE AN IMPORTANT ROLE TO PLAY IN ADDITION TO TRADITIONAL TECHNIQUES LIKE POWER-AWARE DESIGN AND PACKAGE- AND BOARD-LEVEL THERMAL ENGINEERING. THESE AUTHORS DEFINE THE ROLE OF ARCHITECTURE TECHNIQUES AND DESCRIBE HOTSPOT, AN ACCURATE YET FAST THERMAL MODEL SUITABLE FOR COMPUTER ARCHITECTURE RESEARCH.

Kevin Skadron
Mircea R. Stan
Wei Huang
Sivakumar Velusamy
Karthik Sankaranarayanan
David Tarjan
University of Virginia

In recent years, power density in microprocessors has doubled every three years, and experts expect this rate to increase within one to two generations as feature sizes and frequencies scale faster than operating voltages. Because a microprocessor consumes energy and converts it into heat, the corresponding exponential rise in heat density is creating significant difficulties in maintaining reliability and low manufacturing cost. Any design must remove heat from the surface of the microprocessor die, and for all but the lowest-power designs today, such cooling solutions have become very expensive.

Power-aware design alone has failed to stem this tide, requiring *temperature*-aware design at all system levels, including the processor architecture. Localized heating occurs much faster than chip-wide heating; because 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, to be useful for thermal management, must directly target the spatial and temporal behavior of the operating temperature. In fact, *many low-power techniques have insufficient impact 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 in the absence of slack. Temperature-aware design is therefore a distinct, albeit related, area of study.

Temperature-specific design techniques to date have mostly focused on the thermal package (heat sink, fan, and so on). Because 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 absolute worst case is excessive. To reduce packaging cost without unnecessarily limiting performance, it has been suggested that the package should be designed for the worst *typical* application.[1,2] Any applications that dissipate more heat than this cheaper pack-

Published by the IEEE Computer Society
0272-1732/03/$17.00 © 2003 IEEE

age can manage should engage an alternative *runtime* thermal-management technique (*dynamic thermal management* or DTM). Because typical high-power applications still operate 20 percent or more below the absolute worst case,[2] this can lead to dramatic savings. DTM is the philosophy behind the Intel Pentium 4's thermal design.[2] Should operating temperature ever exceed a safe temperature, the clock stops (we refer to this as *global clock gating*) until the temperature returns to a safe zone. As long as the threshold temperature that stops the clock (the *trigger threshold*) is based on the hottest temperature in the system, this approach successfully regulates temperature.

The need for architecture-level thermal management

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 to control thermal behavior, and exploit instruction-level parallelism (ILP). The architecture has detailed temperature information about hot spots and temperature gradients that can combine with dynamic information about ILP to precisely regulate temperature while minimizing performance loss.

Here, we describe two new techniques that outperform prior DTM solutions, but both require design and analysis in the microarchitecture domain, which in turn requires an appropriate modeling capability. Although our work has focused on microarchitecture, system-architecture and operating system techniques have an important and complementary role to play. For example, the operating system can use knowledge of different processes' thermal characteristics to guide scheduling decisions,[3,4] which again requires an appropriate modeling capability.

The need for architecture-level thermal modeling

Researchers have already proposed a variety of microarchitecture and some process-scheduling techniques,[1,3-9] so there is clearly interest in this topic within the broad computer architecture field. To accurately characterize current and future thermal stresses, temporal and spatial nonuniformities, and application-dependent behavior—let alone evaluate architectural techniques for managing thermal effects—we need a suitable model of temperature, yet the architecture community still lacks reliable and practical tools for thermal modeling.

An effective architecture-level thermal model must be

- simple enough to allow architects to reason about thermal effects and tradeoffs,
- detailed enough to model runtime changes in temperature within different functional units,
- yet computationally efficient and portable for use in a variety of architecture simulators.

Contributions

Based on our work, first presented at ISCA 30,[8] we illustrate here the importance of thermal modeling for research on temperature-aware design by describing a compact, dynamic, and portable thermal model for convenient use in architecture research, which we call HotSpot.[8,10] It is publicly available at http://lava.cs.virginia.edu/HotSpot. Using this model, we evaluated several DTM techniques, including two new ones that we proposed earlier[8] but which we reevaluate here with a more refined experimental setup. All of our experiments include the effects of sensor imprecision, which our work finds to significantly handicap runtime thermal management in current technology. Sensors exhibit offset errors that calibration and testing cannot eliminate, as well as runtime noise from the environment.[11] In addition, if we cannot locate a sensor exactly at every possible hot spot, the temperature observed by the sensor might be cooler by some additional error factor.

Because thermal constraints are becoming so severe, we expect that temperature-aware computing will be a rich area for research, drawing from the fields of architecture, circuit design, compilers, operating systems, packaging, and thermodynamics.

Thermal modeling at the architecture level

Our technique starts from the parallels between heat transfer and electrical circuits,

Using an Equivalent RC Circuit to Model Temperature

There exists a well-known duality, shown in Table A, between heat transfer and electrical phenomena: You can consider heat flow to be a "current" passing through a thermal resistance, leading to a temperature difference analogous to voltage.[1] Thermal capacitance is also necessary for modeling transient behavior, to capture the delay before a change in power results in the temperature reaching a steady state. You can compute lumped values of thermal R and C to represent the heat flow among units and from each unit 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 electrical RC time constants. The rationale behind this duality is that you can describe current and heat flow using exactly the same differential equations as those for a potential difference. 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 determining the temperature.

You can compute lumped values of thermal R and C to represent the heat flow among units and from each unit to the thermal package. In the thermal-design community, these equivalent circuits are called *compact models*, and are called *dynamic compact models* if they include thermal capacitors. All thermal capacitors are with respect to ground.

Figure A shows a very simple example of such a model, with Figure A1 showing a typical IC package with a heat sink and Figure A2 showing an equivalent dynamic compact model similar to the one used in Tempest.[2]

References

1. A. Krum, "Thermal Management," F. Kreith, ed., *The CRC Handbook of Thermal Engineering*, CRC Press, 2000, pp. 2.1-2.92.
2. A. Dhodapkar et al., "TEMPEST: A Thermal Enabled Multi-Model Power/Performance Estimator," *Proc. Workshop on Power-Aware Computer Systems*, 2000.

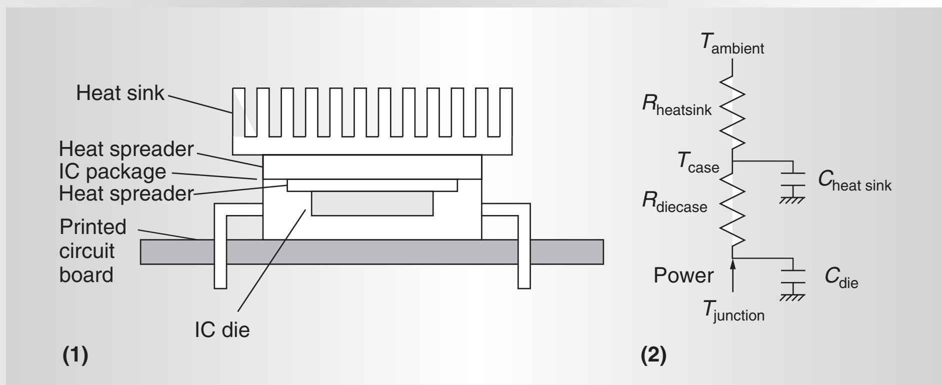

Figure A. IC package with heat sink: physical structure (1) and simple compact thermal model (2).

Table A. Duality between thermal and electrical quantities.

Thermal quantity	Units	Electrical quantity	Units
P, heat flow, power	Watts	*I*, current flow	Amperes
T, temperature difference	Degrees Kelvin	*V*, voltage	Volts
ρ_{th}, thermal resistivity	(meter × degrees Kelvin)/watts	ρ, electrical resistivity	Meters × Ω
R_{th}, thermal resistance	Degrees Kelvin/watts	*R*, electrical resistance	Ω = volts/ampere
C_{th}, thermal mass, capacitance	Joules/Kelvin	*C*, electrical capacitance	Farad = ampere/volt
$R_{th} \times C_{th}$, thermal RC constant	Seconds	$R \times C$, electrical RC constant	s

which we discuss in the "Using an Equivalent RC Circuit to Model Temperature" sidebar. With this as a theoretical basis, we developed the HotSpot model.

HotSpot: A parameterized, BICI, dynamic compact model for microarchitecture studies

For the 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 parameterized, in the sense that it can generate a new compact model for different microarchitectures.
- It must be able to solve the RC circuit's differential equations quickly.

- Finally, it must be *BICI*, that is, boundary- and initial-condition independent: The thermal model's component values should not depend on initial temperatures or the particular configuration under study.

The HotSpot model we have developed meets all these conditions.[10]

Chips today are typically packaged with the die next to a spreader plate, often made of copper or some other highly conductive material, which is in turn next to a fan-cooled, aluminum or copper heat sink. This is the configuration modeled by HotSpot; Figure A1 shows a typical example. Low-power, low-cost chips often omit the heat spreader and sometimes even the heat sink. Mobile devices often use heat pipes and other packaging that avoids the weight and size of a heat sink. These extensions remain areas for future work in HotSpot.

We designed HotSpot to produce an equivalent circuit with a direct and intuitive correspondence to the physical structure of a chip and its thermal package. The RC model therefore consists of three vertical layers for the die, heat spreader, and heat sink, and a fourth vertical layer for the sink-to-air interface. We divide the die layer into blocks that correspond to the microarchitectural blocks of interest and their floorplan. The spreader has five blocks: one that corresponds to the area right under the die (R_{sp}), and four trapezoids corresponding to the periphery that the die does not cover. Similarly, the sink has five blocks: one corresponding to the area directly under the spreader (R_{hs}); and four trapezoids for the periphery. Finally, we use a single thermal resistance ($R_{convection}$) to represent the convective heat transfer from the package to the air. We assume air to be at a fixed ambient temperature (this is not the room ambient temperature, but the temperature inside the computer box)—45° C is a reasonable value. Figure 1 shows the *RC* model for only *R*s in the die layer. 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 are areas for future work in HotSpot.

For the die, spreader, and sink layers, the *RC* model consists not only of a vertical model but also 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 1 accounts for heat flow from block 2 into the heat spreader. Each time step in the dynamic simulation models the power dissipated in each unit of the die as a current source (not shown) at the node in the center of that block.

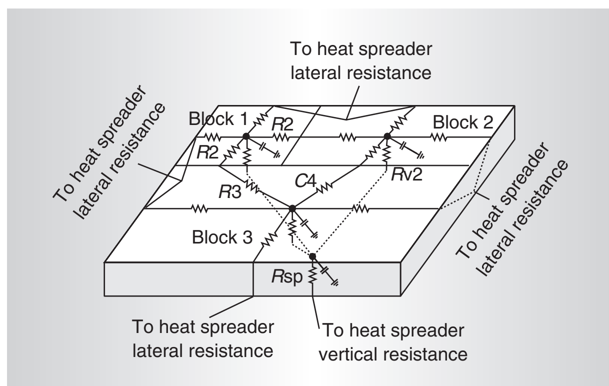

Figure 1. *RC* model for just the die layer.

Deriving the model

Thermal resistance is proportional to material thickness *t* and inversely proportional to cross-sectional area *A* across which the heat is transferring: $R = t / k \times A$, where *k* is the thermal conductivity of the material per unit volume; 100 W/K-m for silicon and 400 W/K-m for copper at 85° C.

Thermal capacitance, on the other hand, is proportional to both thickness and area: $C = c \times t \times A$, where *c* is the thermal capacitance per unit volume, 1.75×10^6 J-K/m^3 for silicon and 3.55×10^6 J-K/m^3 for copper. The capacitors require additional scaling to account for the use of a lumped model rather than a full distributed model, and the resistors must account for spreading and constriction resistances between blocks of different aspect ratios.[12]

HotSpot automates all these calculations. Normally, we would calculate package-to-air resistance $R_{convection}$ from the specific heat-sink configuration as well. Here, however, we instead manually choose a resistance of 0.8 K/W that gives us a good distribution of benchmark behaviors, and represents a reasonable midpoint in the range 0.1 to 2.0 K/W, which characterizes typical heat sinks.[13] As designs employ less-expensive heat sinks—for use with DTM, for

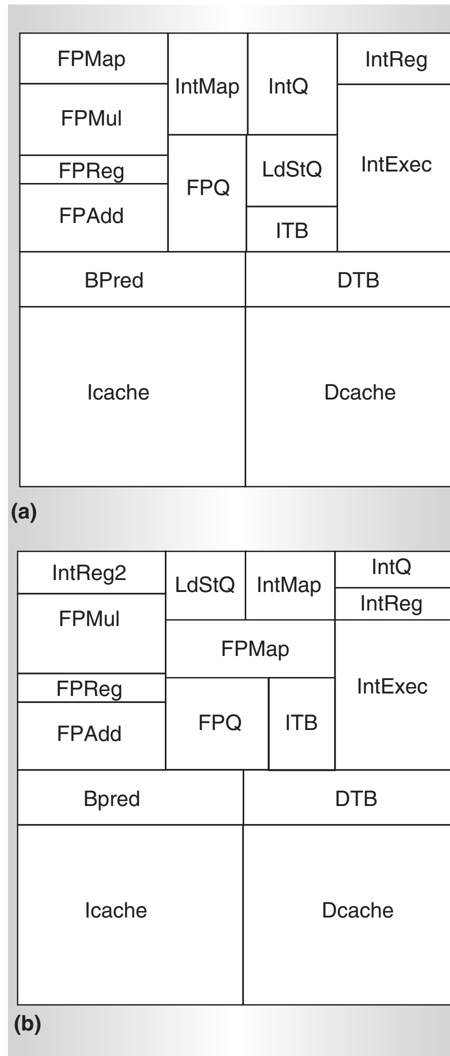

Figure 2. Floorplan corresponding to the 0.13-micron Alpha 21364 used in our simulations: close-up of CPU core (a), and the core with a spare register file (b) used for the migrating-computation DTM technique.

example—this thermal resistance can increase.

Power densities come from the power/performance Wattch simulator,[14] averaged over the last 10,000 clock cycles, which represents a good tradeoff between simulation speed and minimizing sampling error; this tradeoff imposes a simulation slowdown of less than 1 percent. At each time step, HotSpot solves the differential equations describing the *RC* circuit using a fourth-order Runge-Kutta method, returning the new temperature of each block.

The size and adjacency of blocks is a critical parameter for deriving the *RC* model. In all of our simulations thus far, we have used a floorplan (and also approximate microarchitecture and power model) corresponding to that of the Alpha 21364. Like the 21364, this model places the CPU core (shown in Figure 2a) at the center of one edge of the die, with the surrounding area consisting of L2 cache, multiprocessor-interface logic, and so on.

Validating the model

We compare our model against FloWorks (http://www.floworks.com), a commercial, finite-element simulator of 3D fluid and heat flow for arbitrary geometries, materials, and boundary conditions. We are now also in the process of further validating our model against fabricated test chips.

Steady state validation comparing temperatures predicted by FloWorks and HotSpot shows good agreement, with errors (with respect to the ambient, 45° C or 318 K) always less than 5.8 percent and usually less than 3 percent. Transient validation comparing temperature evolution in one block over time, for a sudden increase in power dissipation, shows almost perfect agreement (see our earlier work[8] for further details and results).

Importance of directly modeling temperature

A common fallacy in early thermal investigations is to estimate temperatures by averaging *power* dissipation over a window of time. This relationship is indeed true for steady-state temperatures, but fails to account for lateral coupling among blocks, the role of the heat sink, the nonlinear rate of heating, and other effects. Another common fallacy is to consider temperatures to correspond to instantaneous power dissipation, when in fact the thermal capacitance acts as a low-pass filter in translating power variations into temperature variations. As an example of the poor correlation between average power and localized transient temperature, Figure 3 shows a scatter plot of temperature versus average power for the integer register file during execution of the SPECcpu2000 benchmark gcc for an averaging interval of 0.033 seconds.

Comparison of techniques for architectural DTM

Several microarchitecture techniques target runtime temperature regulation. Here, we

evaluate and compare just a few of them using HotSpot, including the effects of sensor noise, and describe the two new techniques we introduced earlier: *temperature-tracking frequency scaling* and *migrating computation.*[8] Our simulations assume the maximum tolerated operating temperature in any location is 85° C. To model sensors, we assume one sensor per architectural block, random noise of ± 1° C after averaging, and an offset error of ± 2° C. We also allow an extra 0.2° C margin to accommodate the time required for DTM to achieve the lower temperature. Altogether, this yields a trigger threshold of ± 81.8° C.

For our experiments, we modify Wattch[14] to obtain a power model based on power data for the Alpha 21364. We assume an aggressive configuration running at 1.3 V and 3 GHz. We have also updated Wattch's power model to incorporate the temperature-dependence of leakage.

We simulate nine benchmarks from the SPECcpu2000 suite, compiling and statically linking them to the Alpha instruction set using the Compaq Alpha compiler with SPEC *peak* settings. These simulations 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. Simulation must actually begin 300 million instructions prior to this sample to purge cold-start bias in the caches and to allow the entire chip to reach steady-state temperatures. Only after this warm-up process does statistics gathering commence. Heat sink temperatures are lower with DTM, because it prevents all temperatures in the system from rising as high. Simulations must account for this effect by using the correct initial temperature for the heat sink.

We chose our nine benchmarks to represent a mixture of integer and floating-point programs with intermediate and extreme thermal demands; all those omitted operate below the 81.8° C trigger threshold and are uninteresting for DTM studies. Table 1 provides a list of the benchmarks we study along with their basic performance, power, and thermal characteristics. The table shows that IPC and peak operating temperature only loosely correlate with average power dissipation.

Figure 4 presents the slowdown (execution time with thermal management divided by original execution time) for each of the thermal-management techniques, averaged across all of the benchmarks. None of the techniques incur any thermal violations. All the performance differences compared to the baseline are significant at the 99 percent confidence level. A description and comparison of the various DTM techniques follows.

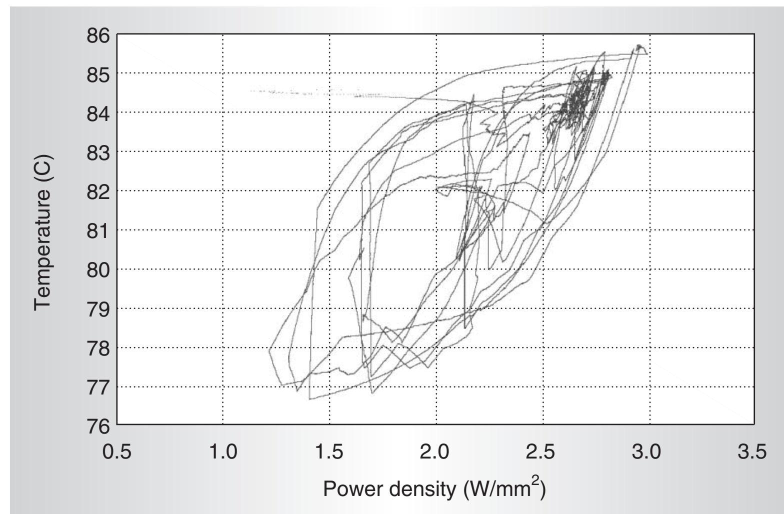

Figure 3. Temperature versus average power density for gcc with a power-averaging interval of 0.033 seconds.

Temperature-tracking dynamic frequency scaling

Independently of the relationship between frequency and voltage, the temperature-dependence of carrier mobility in CMOS means that frequency is also linearly dependent on the operating *temperature*. 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 this temperature dependence is mild within the operating region of interest, the performance penalty for doing so is almost negligible.

When changing frequency, the processor must typically stall for anywhere from 10 μs to 50 μs to accommodate resynchronization of the clock's phase-locked loop (PLL). We examine a discretized frequency scaling with 10 MHz steps and a 10 μs stall time for every change in the operating frequency, and an ideal version that does not incur this stall but

Table 1. Summary characteristics for benchmarks exhibiting thermal stresses.

Benchmark	IPC	Average power (W)	Cycles in thermal violation (percentage)	Dynamic maximum temperature (° C)	Steady-state temperature (° C)
mesa (F)*	2.7	31.5	40.6	83.4	82.6
perlbmk (I)	2.3	30.4	31.1	83.5	81.6
gzip (I)	2.3	31.0	66.9	84.0	83.1
bzip2 (I)	2.3	31.7	67.1	86.3	83.3
eon (I)	2.3	33.2	100	84.1	84.0
crafty (I)	2.5	31.8	100	84.1	84.1
vortex (I)	2.6	32.1	100	84.5	84.4
gcc (I)	2.2	32.2	100	85.5	84.5
art (F)	2.4	38.1	100	87.3	87.1

* F indicates floating-point; I, integer.

the change in frequency does not take effect until after 10 μs has elapsed. We call these TT-DFS and TT-DFS-i (for ideal).

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

From the simulation results, we can see that TT-DFS is one of the two best techniques for thermal management, with the TT-DFS-i version, of course, being slightly better. The performance penalty for even the hottest benchmarks is small; the worst is art with only a 2.8 percent slowdown. If the maximum junction temperature of 85° C is strictly based on timing concerns, and the chip can tolerate somewhat higher temperatures without an undue reduction in its operating lifetime, then TT-DFS is attractive because its impact is so gentle. We observe similar excellent performance even for a much less capable heat sink of 1.0 K/W, for which operating temperatures will be considerably higher.

If both timing and physical reliability dictate the junction temperature of 85° C, then TT-DFS is not a viable approach: It cannot enforce the specified junction temperature. All the remaining techniques can do so.

Fetch gating and global clock gating

Fetch gating (FG) alternates between fetching and preventing fetch, reducing instruction activity through the pipeline and hence power density. The choice of ratio or duty cycle between cycles spent fetching versus not fetching is a feedback control problem, for which we use a proportional-integral (PI) controller (hence the name FG-PI) with settings confirmed by exhaustive search.

Global clock gating might seem more attractive, because it attains extra power reduction by eliminating power dissipation in the clock tree. But rapidly stopping and starting the entire clock tree (required to exploit ILP) can 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 for GCG-PI. We found that localized toggling confers negligible benefit over fetch gating; 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 has almost the same performance behavior as fetch gating.

The gating/toggling techniques—GCG and FG—all perform much worse than the other techniques in simulation. GCG outperforms FG because it eliminates power dissipation in the clock tree and allows the chip to cool faster. In principle, these techniques should capitalize on ILP; however, that is only possible 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 techniques often require many of these aggressive duty cycles.

Dynamic voltage scaling

Designers have long regarded DVS as a solution for reducing energy consumption; researchers have recently proposed it as one solution for thermal management[1,5] and Transmeta's Crusoe processors use it for this purpose. When changing the processor voltage, a processor must reduce frequency in conjunction with voltage, because circuits switch more slowly as the operating voltage approaches the threshold voltage.

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 can 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 10 for Transmeta's LongRun, and 40 for the Intel XScale. With our heat sink and benchmarks, 93 percent of the nominal voltage is the maximum voltage reduction that eliminates thermal violations, so this is always the lowest voltage setting that we model. We tried a variety of step sizes, but for DTM they all give almost exactly the same performance, differing by less than 0.4 percent for DVS and less than 0.01 percent for DVS-i. These results mean that the PI feedback control reported in our prior work[8] is unnecessary. DTM only needs two voltages: the maximum voltage, and a low voltage that eliminates all possible thermal violations. The main reason is that the time required to reduce thermal stress is proportional to how far you reduce the voltage: Lower voltages require a greater reduction in clock frequency but take less time to reduce thermal stress.

From the simulation results, we can see that DVS can be much better than the gating techniques, but if resynchronizing the clock requires stalls, the associated overhead is substantial and wipes out much of DVS's benefit. The gating techniques, in contrast, can activate within a few cycles if they use on-chip sensors and microarchitectural trigger.

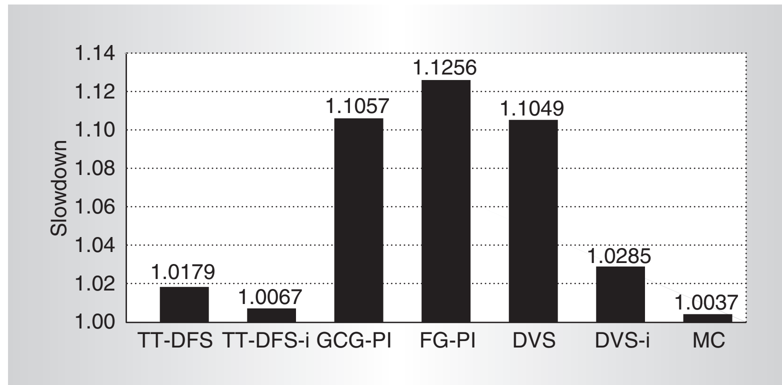

Figure 4. Slowdown for various DTM techniques.

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 an additional latency, which a communication incurs 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.

Because our experiments consistently show the integer register file to be the hottest unit on the chip (even for many floating-point programs), we developed a new floorplan that includes an extra copy of the integer register file, shown in Figure 2b. When the primary register file overheats, the processor stalls instruction fetch, lets instructions ready to write back finish, and copies the register file 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 performance and power model accounts for the extra distance by charging two extra cycles for every register file access. Accessing the distant register file also involves additional power dissipation because the signal has to travel over a longer distance, which we have approximated by estimating the capacitance of the requisite wires. When the primary register file returns to a safe temperature, computation resumes using the primary register file. We call this scheme *migrating computation*. Note that, because MC will not prevent thermal violations, it requires a fail-safe mechanism should

MC be insufficient to regulate temperature. We use FG-PI for this purpose.

From the results, MC is the best DTM technique at 0.8 K/W; it 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 is the dominant effect for a heat sink of 0.8 K/W, where maximum temperatures are not far above 85° C; indeed, the benchmarks we simulated rarely used the spare register file. This shows the importance of considering on-chip thermal diffusion in designing the floorplan and in simulating DTM techniques.

Second, MC can exploit ILP to hide the extra latency of the spare register file. Third, the complete elimination of activity in the primary register file allows it to cool quickly, minimizing the use of the slower secondary register file.

Although MC is the best technique with a heat sink of 0.8 K/W, that is not the case for a less-expensive heat sink. At 1.0 K/W, operating temperatures are higher, the special floorplan is no longer enough to prevent thermal violations in the register file and uses the spare register file more often. Although MC remains superior to DVS and the gating techniques, a less-expensive heat sink reduces its performance advantage, and MC is no longer nearly as attractive as TT-DFS.

Although we were unable to explore a wider variety of floorplans, the success of these floorplan-based techniques suggests an appealing way to manage heat.

Research in temperature-aware architecture is still in its infancy, and there are many topics that require additional study. We mention just a few.

In terms of modeling, perhaps the most important and interesting area for future work is the inclusion of heating from the clock grid and other interconnect. HotSpot currently approximates the effects of wires by including their power dissipation in the dynamic, per-block power density values that drive the RC model. 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. Another issue that requires further study is the appropriate granularity at which to derive the RC model. This depends on the underlying power model. Our current power model assumes a uniform power density for each microarchitectural block.

This research area also needs better objective functions. From a reliability standpoint, today's thermal design rules based on maximum temperature might be insufficient or require unnecessarily conservative design margins: Localized hot spots on the chip can far exceed the average chip temperature. Although it is understood that spatial gradients and temperature cycles over time can be more important than absolute junction temperatures, a simple junction-temperature specification doesn't capture either effect adequately. Indeed, it might be more useful to regulate temperature to control expected lifetime rather than using fixed rules-of-thumb on maximum temperatures and gradients.

Better runtime temperature sensing would help to translate runtime techniques into hardware. It might also be helpful to export these sensors' readings—probably at some low sampling rate—to the operating system. But, as mentioned, today's CMOS sensors are noisy and difficult to calibrate, so this work would benefit from data fusion techniques to reduce sensor imprecision.

Finally, industry needs techniques in all design domains—circuit, microarchitecture, system architecture, and operating system—to allow the chip to autonomously regulate its temperature. Each domain can control temperature in different ways and at different timescales. It might even be possible to combine techniques across domains into a cooperating hierarchy, using operating system knowledge of workload requirements and behavior to provide hints to the architecture, which, for example, might let the microarchitecture respond proactively rather than reactively.[9] The microarchitecture can in turn use runtime knowledge to regulate circuit techniques like the choice of threshold voltage.

Another important problem is to understand the interactions among dynamic management techniques for active power, leakage power, voltage stability, and thermal effects. Together, these areas present a rich but poorly understood design space where the same technique can possibly serve multiple purposes but in different settings. MICRO

Acknowledgments

This work is supported in part by NSF under grant numbers CCR-0133634 and MIP-9703440; two grants from Intel MRL; and an Excellence Award from the University of Virginia's Fund for Excellence in Science and Technology. We thank Peter Bannon, Shekhar Borkar, Pradip Bose, David Brooks, Howard Davidson, Vivek De, Antonio González, José González, Konrad Lai, Margaret Martonosi, Avi Mendelson, Ronny Ronen, and Gad Sheaffer for their helpful comments.

References

1. D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *Proc. 7th Int'l Symp. High-Performance Computer Architecture* (HPCA 01), IEEE CS Press, 2001, pp. 171-182.
2. S. Gunther et al., "Managing the Impact of Increasing Microprocessor Power Consumption," *Intel Technology J.*, Feb. 2001; http://www.intel.com/technology/itj/q12001/articles/art_4.htm.
3. F. Bellosa et al., "Event-Driven Energy Accounting for Dynamic Thermal Management," *Proc. 2003 Workshop on Compilers and Operating Systems for Low Power*, 2003.
4. E. Rohou and M. Smith, "Dynamically Managing Processor Temperature and Power," *Proc. 2nd Workshop Feedback-Directed Optimization*, 1999.
5. W. Huang et al., "A Framework for Dynamic Energy Efficiency and Temperature Management," *Proc. 33rd Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO-33), IEEE CS Press, 2000, pp. 202-213.
6. C.-H. Lim, W. Daasch, and G. Cai, "A Thermal-Aware Superscalar Microprocessor," *Proc. Int'l Symp. Quality Electronic Design* (ISQED 02), IEEE Press, 2002, pp. 517-522.
7. K. Skadron, T. Abdelzaher, and M.R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," *Proc. 8th Int'l Symp. High-Performance Computer Architecture* (HPCA 02), IEEE CS Press, pp. 17-28.
8. K. Skadron et al., "Temperature-Aware Microarchitecture," *Proc. 30th Ann. Int'l Symp. Computer Architecture* (ISCA 03), IEEE CS Press, 2003, pp. 2-13.
9. J. Srinivasan and S.V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," *Proc. 17th Int'l Conf. Supercomputing* (ICS 03), ACM Press, 2003, pp. 109-120.
10. M.R. Stan et al., "Hotspot: A Dynamic Compact Thermal Model at the Processor-Architecture Level," to be published in *Microelectronics J.: Circuits and Systems*, Elsevier, 2003.
11. A. Bakker and J. Huijsing, *High-Accuracy CMOS Smart Temperature Sensors*, Kluwer Academic, 2000.
12. S. Lee et al., "Constricting/Spreading Resistance Model for Electronics Packaging," *Proc. ASME/JSME Thermal Eng. Conf.*, American Soc. of Mechanical Eng., 1995, pp. 199-206.
13. R. Viswanath et al., "Thermal Performance Challenges from Silicon to Systems," *Intel Technology J.*, Aug. 2000; http://www.intel.com/technology/itj/q32000/articles/art_4.htm.
14. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. 27th Ann. Int'l Symp. Computer Architecture* (ISCA 00), ACM Press, 2000, pp. 83-94.

Kevin Skadron is an assistant professor in the Department of Computer Science at the University of Virginia. **Mircea R. Stan** is an associate professor of electrical and computer engineering at the University of Virginia. **Wei Huang** is a PhD student in the Department of Electrical and Computer Engineering at the University of Virginia. **Sivakumar Velusamy** is a PhD student in the Department of Computer Science at the University of Virginia. **Karthik Sankaranarayanan** is a PhD student in the Department of Computer Science at the University of Virginia. **David Tarjan** is a student finishing his diploma at the ETH Zurich and will start as a PhD student in the University of Virginia's Department of Computer Science in 2004; he also visited the University of Virginia for five months in 2002.

Direct questions and comments about this article to Kevin Skadron, Dept. Computer Science, Univ. of Virginia, 151 Engineer's Way, PO Box 400740, Charlottesville, VA 22904-4740; skadron@cs.virginia.edu.