Hybrid Architectural Dynamic Thermal Management

Kevin Skadron
Department of Computer Science, University of Virginia
Charlottesville, VA 22904
skadron@cs.virginia.edu

Abstract

When an application or external environmental conditions cause a chip's cooling capacity to be exceeded, dynamic thermal management *(DTM) dynamically reduces the power density on the chip to maintain safe operating temperatures. The challenge is that even though this reduction in power density reduces heat dissipation and can be used to regulate temperature and reduce the need for expensive thermal packages, reducing power density may come at a cost in execution speed. This paper shows the importance of processor-architecture techniques for DTM, and proposes a new, "hybrid," low-overhead implementation based on combining fetch gating and dynamic voltage scaling (DVS). When thermal stress is low, fetch gating is superior because it exploits instruction-level parallelism (ILP). Once thermal stress becomes severe enough that fetch gating degrades ILP, DVS is engaged instead to take advantage of its greater ability to reduce power density. We show that under a variety of assumptions about DVS implementation, a hybrid policy reduces DTM performance overhead by 25% on average compared to DVS, and is easy to design.*

1. Introduction

In recent years, power density in microprocessors is doubling every three years, and this rate is expected to increase as feature sizes and frequencies scale faster than operating voltages [1, 16]. Because energy consumed by the microprocessor is converted into heat, the corresponding exponential rise in heat density is creating vast difficulties in reliability and manufacturing costs. For high-performance processors, cooling solutions are rising at $1–3 or more per watt of heat dissipated [1, 6], making it more difficult to deploy new systems. Cooling costs are exacerbated by the fact that cooling solutions must typically be designed for the worst-case. Yet the worst case usually far exceeds the typical case, so even though worst-case design is necessary, the resulting solutions are substantially over-engineered for typical operating conditions.

Dynamic thermal management (DTM) allows the thermal package to be designed for power densities exhibited by *typical* applications, with the chip itself adapting its runtime *behavior* 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, like dynamic voltage scaling (DVS) or global clock-gating. 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. Gunther et al. [6] reported that targeting the thermal package for the "worst typical" application rather than the true worst case, and using DTM in the form of 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 power density often entails slower execution. But in the near future, the 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 which require more expensive, compact thermal technologies like heat pipes [20]. Improving DTM design will allow greater cost savings with minimal performance cost.

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, like finding the best voltage and frequency setting that safely cools the chip while minimizing slowdown. That can be accomplished using feedback control. 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 frequency. 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 instruction-level parallelism (ILP) has less overhead than DVS—possibly even no overhead if ILP is sufficient. (We call these ILP-exploiting techniques *ILP techniques*.) DVS also carries inherent overhead due to step size and switching time that make it unattractive for mild thermal stress. We are not aware of any prior work examining tradeoffs between architectural ILP techniques and traditional DVS techniques for thermal management.

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. In fact, we show that this approach is so effective and so insensitive to tuning parameters that the need for feedback control over the ILP and DVS settings can be eliminated with negligible effect on DTM performance overhead. Hybrid DTM is therefore easier to design and much more *robust* than traditional DTM techniques, an attractive property compared to many techniques that are highly sensitive to tuning parameters. Overall, we show that a hybrid policy can reduce DTM overhead by about 25% compared to the best existing technique, DVS; and can also outperform even an idealized DVS that has no switching overhead. Specifically, we make the following contributions:

- We propose the notion of a hybrid DTM technique, and show how to understand what characteristics dictate the crossover point between the ILP technique and DVS. We find that hybrid schemes are the best DTM technique proposed so far. This result shows the value of an architectural approach to thermal management.
- We show how to eliminate the need for feedback control in a hybrid DTM technique, making hybrid DTM more robust than previous adaptive techniques.
- We show that "binary" DVS, with only two voltage levels, is just as good for DTM as more sophisticated DVS schemes. This is attractive because the fewer the voltages supported, the less the testing overhead; indeed many chips use only two voltages.

The rest of this paper is organized as follows. The next provides further background and discusses related work. Then Section 3 describes our modeling setup, and Section 4 describes the DTM techniques we study. Section 5 evaluates hybrid DTM, and Section 6 concludes the paper.

2. Background

Power-aware design alone has failed to stem the tide of rising operating temperature for a variety of reasons, including continuing demand for higher performance, the fact that much power-aware design focuses on energy efficiency and battery life rather than peak operating temperature, and the fact that on-chip temperatures exhibit hotspots—spatial gradients due to variations in power density among different functional units, and temporal gradients due to variations in computational activity among different phases of a program and among different programs. Many low-power techniques have insufficient or no effect on operating temperature, because they do not reduce power density in hotspots, or because they only reclaim slack and do not reduce power and temperature when no slack is present. Controlling temperature with in-chip hardware responses requires power-management techniques that directly target the spatial and temporal behavior of operating temperature.

One of the most common techniques discussed for DTM is dynamic voltage scaling, possibly with feedback control [12, 17], and a variety of processors offer DVS today. There are three reasons why ILP techniques can outperform DVS despite the cubic reduction in power density that DVS provides with respect to the reduction in frequency. The first is that ILP techniques can exploit ILP while DVS cannot (because the clock speed itself is changing while the cycle count stays mostly unchanged). The second is that when DVS only provides discrete steps, each step imposes a minimum quantum in performance loss due to the associated change in frequency. Finally, ILP techniques do not incur any stall time to engage or switch DTM settings.

Besides DVS, the other two DTM techniques that to our knowledge have been implemented in current processors are clock gating in the Pentium 4 [6], in which the entire processor clock is stopped for 2 microseconds at a time; and fetch throttling in the PowerPC G3 [13], in which the number of instructions fetched per cycle is reduced.

Other DTM techniques that have been proposed in the research literature are fetch gating [2], in which the instruction-fetch rate is slowed or stopped; local toggling [17], in which the processor domain(s) in thermal stress are slowed or stopped; and "migrating computation" [7, 11, 17]. We have found that local toggling confers little advantage over fetch gating and do not consider it further, and the cost-benefit concerns of adding extra hardware for migration make its study beyond the scope of this paper.

A distinction should be made between fallback techniques like the DEETM hierarchy of Huang et al. [8], and the hybrid techniques we propose here. Fallback techniques use a DTM technique until its ability to control temperature is exhausted and an additional or alternative technique is needed to prevent thermal violations. In contrast, the hybrid technique we propose uses an ILP technique *only while doing so is optimal* and then switches to DVS. As we show, this crossover point is well before the ILP technique's cooling capability has been exhausted.

3. Simulation Setup

In this section, we give a brief overview of the various aspects of our simulation framework. We use the same simulator, benchmark, and setup as in our prior work [17], with minor exceptions noted below.

Power-Performance Simulation. In order to study the temporal and spatial evolution of temperature over interesting time periods in real workloads, we simulate at the microarchitecture level, using a power model based on the Alpha 21364 [17] that is implemented using the SimpleScalar/Wattch [3, 4] toolkits. The 21364 consists of a superscalar, out-of-order-issue processor core identical to the 21264, with a large L2 cache and (not modeled) multiprocessor logic added around the periphery. We only consider uniprocessor benchmarks, so we replace the multiprocessor logic with additional cache to fill out the remaining die area. The 21364 power data was for 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}=1.3V, and a clock speed of 3 GHz. We updated Wattch's leakage model to model leakage as a function of temperature using ITRS [16] projections for the 0.13μ node—see [18].

Thermal Simulation. It is convenient to define several terms: the *emergency threshold* is the temperature above which the chip is in *thermal violation*; we use 85° based on 2001 ITRS recommendations for future technology nodes. We assume that the chip should never violate the emergency threshold. The *trigger threshold* is the temperature above which runtime thermal management begins to operate; obviously, trigger < emergency.

For modeling temperature, we use our HotSpot model [17], in which a "dynamic compact model" of thermal resistances and capacitances is derived from the layout of microarchitectural blocks. The various RC pairs represent heat flow in both the lateral and vertical directions. This model has been validated against a commercial finite-element model. An example of the RC model is shown in Figure 1, with a simple floorplan of just three blocks for better legibility—our experiments use a floorplan corresponding to the 0.13μ 21364 floorplan, shown in Figure 2. One of the important features of HotSpot is that only microarchitectural parameters and estimates of block areas are needed to derive the equivalent RC circuit, making it useful for early, planning-stage research before detailed design and layout has been completed. Following [17], we use time steps of 10,000 cycles, average the power calculated by Wattch for each block during each time step, and use that average power for each block as the current source at each block's node in the RC circuit. Because temperature evolves over microseconds to milliseconds, this procedure keeps sampling error below 0.1% in temperature, with simulation overhead of less than 1%. For the package, we assume a die thickness of 0.5mm, the same copper heat spreader and heat sink as [17], and an equivalent thermal resistance for sink-to-air heat transfer of 1.0 K/W, corresponding to a low-cost package. This package was selected to push some of the SPECcpu2000 benchmarks into thermal stress in order to evaluate DTM with real programs.

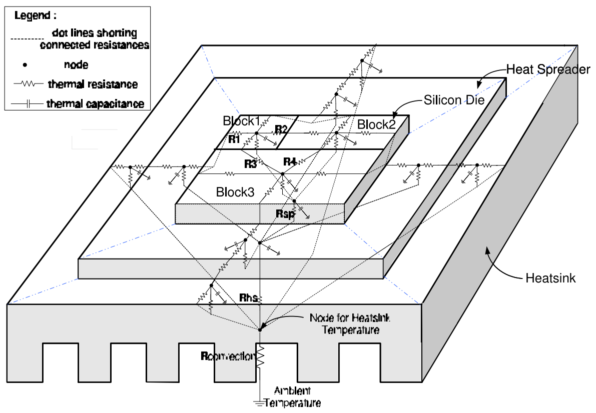
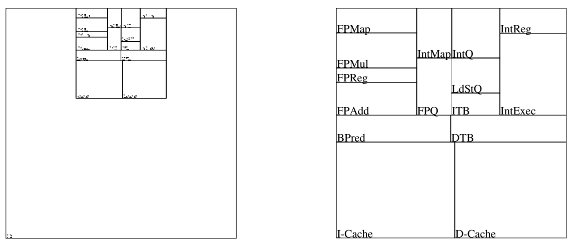


Figure 1. Example HotSpot RC model for a floorplan with three architectural units, a heat spreader, and a heat sink.

To model sensor effects, we assume that each architectural block has one sensor in the middle of the block. The effective precision after averaging is $\pm 1°$ and the sensor may also have a fixed offset of as much as 2°. For the 85°C



(a) 21364 floorplan (b) CPU core

Figure 2. (a): Floorplan corresponding to the 21364. (b): Closeup of 21364 core.

maximum allowed *true* operating temperature, this leads to a practical limit of 82°. To allow enough time for the DTM response to begin cooling the chip, we set a trigger temperature of 81.8°. The sampling rate for reading the sensors is 10 kHz, an aggressive but reasonable value according to [17]. Note that this limits the rate at which DTM can observe changes in temperature and adjust its setting. Sensor *placement* is also important: if the critical transistors in a sensor are not co-located with potential hotspots, the observed temperature may be cooler than the actual hotspots which we are attempting to regulate. This requires an additional design margin that can be added to the sensor's fixed offset.

Brooks and Martonosi [2] pointed out that for fast DTM response, interrupts are too costly. We adopt their suggestion of on-chip circuitry that directly translates any signal of thermal stress into actuating the thermal response. We assume that it simply consists of a comparator for each digitized sensor reading.

Benchmarks. We evaluate our results using the nine hottest benchmarks from the SPEC CPU2000 suite, representing a mixture of integer and floating-point programs with intermediate and extreme thermal demands: *mesa, perlbmk, gzip, bzip2, eon, crafty, vortex, gcc*, and *art*. All operate above 85° 100% of the time and above 90° most of the time. The benchmarks are compiled and statically linked for the Alpha instruction set using the Compaq Alpha compiler with SPEC *peak* settings and include all linked libraries. For each program, we simulate a single representative sample of 500 million instructions using UCSD's SimPoints [15]. When we start simulations, we initialize all temperatures to their steady-state values and then run the simulations in full-detail cycle-accurate mode (but without statistics gathering) for 300 million cycles to bring caches to steady-state miss ratios and bring operating temperatures to accurate runtime values. Only then do we begin to track any experimental statistics. For all the benchmarks we study, the hottest unit is the integer register file.

Note that over these time scales, the heat sink temperature changes little. Temperature changes in the silicon, on the other hand, take place as fast as 1°/ms, so many phases of program behavior can be observed and many cycles of DTM response can be modeled.

4. Techniques for Architectural DTM

This section describes the various architectural mechanisms for dynamic thermal management that are evaluated in this paper: three existing techniques, DVS, fetch gating, and clock gating; and our new hybrid techniques. All are simulated at levels that eliminate thermal violations.

4.1. Existing DTM Techniques

Dynamic Voltage Scaling. When changing the processor voltage, frequency must be reduced in conjunction with voltage, a capability many processors offer today. We used Cadence with BSIM 100nm low-leakage models to simulate the period of a 101-stage ring oscillator to determine the frequency for each voltage step—for more details, see [18]. Different implementations of DVS offer various step sizes for the voltage and frequency, ranging from two with Intel's SpeedStep [9] to at least ten for Transmeta's LongRun [5], and forty for the Intel Xscale [14]. For thermal management, we found that multiple step sizes are unnecessary. We tried a variety of step sizes: continuous, ten, five, three, and two. For two steps, if the temperature dictates that DVS must be engaged, the low voltage is used. This type of response simply entails comparators on the sensor readings. For the other schemes, we use a PI controller to set the voltage to the highest level that regulates temperature. A problem arises when the controller is near a boundary between DVS settings and stalls are required on changes, because small fluctuations in temperature can produce too many changes in setting, along with the associated overhead. To prevent this, we apply a simple low-pass filter to decide whether to increase the voltage. Filtering is not used for lowering the voltage, because that is compulsory in response to thermal stress. We model two possible scenarios for the overhead of switching voltage/frequency settings. In the first ("stall"), the penalty to change the DVS setting is 10μs, during which the pipeline is stalled. In the second ("ideal") the processor may continue to execute through the change but the change does not take effect until after $10\mu s$ have elapsed.

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. The reason for this behavior is 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.

Instead of step size, what does matter for DTM is the value of the lowest voltage. With our heat sink and benchmarks, 85% of the nominal voltage is the largest value for the low-voltage setting that eliminates thermal violations. Based on these results, the rest of this paper only presents data for binary DVS.

Fetch Gating. With fetch gating, fetch is prevented at some duty cycle, reducing the number of instructions flowing through the pipeline and hence the unit activities and the power densities. This entails gating both the I-cache accesses and branch/target predictions. The choice of duty cycle is a feedback-control problem, for which we use an integral controller, with settings confirmed by exhaustive search. The hardware to implement this controller is minimal. A few registers, an adder, and a multiplier are needed, along with a state machine to drive them. Single-cycle response is not needed, so the controller can be made with minimum-sized circuitry. The datapath width in this circuit can also be fairly narrow, since only limited precision is needed.

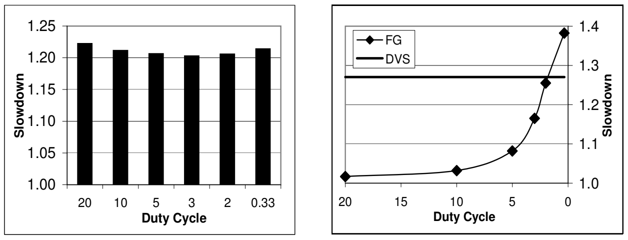
Clock gating might seem more attractive, because it attains extra power reduction by eliminating power dissipation in the clock tree. But stopping and starting the entire clock tree on a rapid basis (required to exploit ILP) may be infeasible, especially given voltage-stability concerns. The mild levels of fetch gating that we employ maintain activity throughout the pipeline and should present less of a voltage-stability problem. If fine-grained clock gating is in fact feasible, our results here represent a lower bound on the benefits of hybrid DTM.

4.2. Hybrid DTM

Our hybrid techniques use fetch or clock gating as the ILP techniques when a mild DTM response is required, because when DTM response is mild, the overhead for these ILP techniques is *lower* than DVS. This is even true for the ideal DVS schemes, because they still reduce clock frequency, while mild fetch gating may be mostly hidden by ILP. Naturally, the stalls incurred by non-ideal DVS substantially increase the DTM range for which the ILP techniques are superior.

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.

There are two ways to implement a hybrid scheme. The most obvious perhaps is to use a feedback-controlled ILP technique until the crossover point is reached, and then use DVS. This outperforms both DVS-ideal and DVS-stall, and we call this "PI-Hyb." A much simpler approach is to use a single fixed level of DTM response for the ILP technique, and when the temperature is too far above the trigger point, to lower the voltage instead. We call this "Hyb." This latter approach is appealing because it eliminates any risks of imprecision, oscillation, etc. with the controllers, and because it is compatible with a binary DVS. In fact, we show that it sacrifices negligible performance compared to feedback-controlled (PI) hybrid DTM. Implementation is slightly more complex than for binary DVS, because comparison is required against two thresholds rather than just one, but this remains simpler than feedback control.



(a) (b)

Figure 3. DTM slowdowns for (a) different PI-Hyb configurations and (b) FG and DVS.

5. Evaluation of Hybrid DTM

5.1. Finding the Crossover Point

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; developing such a metric is an interesting and important area for future work. Instead, we simply conducted a search across a range of FG duty cycles. Since binary DVS is sufficient for the DVS component and since the ILP techniques have a reasonably limited range of duty cycles, this is not overly burdensome.

As Figure 3a shows, the best hybrid configuration uses a maximum duty cycle of 3 (i.e., skip fetch once every three cycles) for PI-Hyb with DVS-stall. This figure plots, for PI-Hyb, the slowdown as a function of the fetch-gating 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.) All these DTM configurations fully prevent thermal emergencies. A duty cycle of 3 represents the crossover point. Beyond this point, it becomes difficult for the ILP technique to successfully exploit instruction-level parallelism, and slowdown rises sharply; whereas DVS's cubic advantage overcomes the stalls associated with switching settings. This is shown in Figure 3b, which plots the slowdown of stand-alone fetch gating, with the overhead of stand-alone DVS superimposed as a straight line for comparison purposes. Of course, most of these duty cycles are insufficient to eliminate all thermal violations. Only a duty cycle of 0.33 does that, although it is overly harsh much of the time—that is why FG needs PI control. Figure 3b merely shows the linear relationship between duty cycle and slowdown that sets in at a duty cycle of about 3, the point at which all ILP has been exhausted. It might seem that a duty cycle of 2 would be even better for hybrid DTM, because the overhead of FG and DVS is approximately equal at this point. But Figure 3b does not distinguish the effectiveness of each technique according to the severity of thermal stress, and hence does not reflect the savings achieved by combining FG and DVS. For example, Figure 3b includes the high overhead costs of DVS for even very mild thermal emergencies, where FG would be better, and the high overhead costs of FG for severe thermal emergencies, where DVS is better.

For idealized DVS without any stalls, the gentlest duty cycle of 20 is preferred. Because no stalls are incurred to switch DVS settings, only the mildest fetch gating, where ILP hides almost all performance impact from FG, can give better results than DVS.

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

5.2. Performance Comparison

Figure 4 compares the performance of the various DTM schemes we have described: fetch gating, DVS, PI-Hyb, and Hyb (without PI control). DVS is consistently better than fetch gating, but the hybrid schemes are even better. When DVS incurs overhead to change settings, hybrid DTM can improve performance by 5.5–6%, 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%. This represents about an 11% reduction in DTM overhead. All the performance differences compared to DVS are significant at the 99% confidence level. These results show that with the overheads found in typical DVS implementations today, hybrid DTM offers impressive benefits, but that much of the benefit comes from minimizing changes in DVS setting and the associated overhead.

Figure 4 also shows that eliminating PI control sacrifices almost no performance, and in fact it performs slightly better with DVS stall. (The small difference is also significant at the 99% confidence level.) 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 too, but here the PI control is needed. If only one FG duty cycle were available, it would have to be too high—a duty cycle of 2—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. Conclusions and Future Work

In this paper, we have pointed out that dynamic thermal management can be made more efficient by applying a combination of different thermal responses. The key insight is that at mild levels of thermal stress, "ILP techniques" incur less overhead, because they take advantage of the microarchitecture's ability to exploit instruction-level parallelism to mask bubbles in the pipeline. This hybrid DTM approach differs from prior work by switching away from the ILP

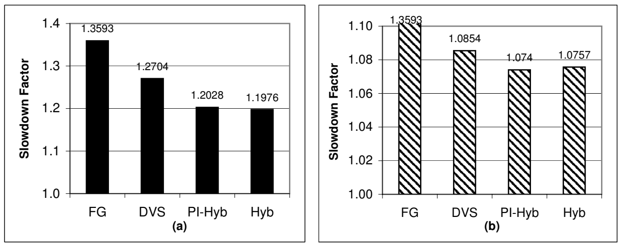


Figure 4. DTM slowdown, averaged across nine SPECcpu2000 benchmarks, comparing fetch gating, DVS, PI-Hyb, and Hyb for (a) DVS-stall and (b) DVS-ideal.

technique when instruction-level parallelism no longer sufficiently hides the overhead of the DTM technique—rather than waiting until the DTM technique fails to cool the chip sufficiently. When combining fetch gating with traditional DVS schemes, we find that the proper crossover point between is when most of the ILP has been exploited, slowdown begins to rise in proportion to the duty cycle, and DVS is better able to reduce power density despite the overhead of changing voltage settings. On the other hand, when combining fetch gating with an idealized DVS scheme that has no overhead for switching voltage and frequency, only the mildest levels of fetch gating are justified, where ILP almost completely hides the fetch gating. In both cases, hybrid DTM confers benefits, another example of the importance of architectural approaches for thermal management.

We showed that for typical DVS implementations where changing the voltage does require the processor to stall, the hybrid DTM approach outperforms DVS by 5.5–6% on average. This represents a 25% reduction in DTM overhead. When compared to an idealized DVS with no overhead, hybrid DTM still outperforms DVS by 1%.

We also showed that a hybrid technique can eliminate the need for sophisticated feedback control on adaptive techniques like DVS or fetch gating. This creates a robust technique that avoids tuning difficulties, while still preserving the performance benefits of hybrid DTM. And we showed that “binary” DVS with only two voltage settings is just as good as more sophisticated DVS schemes.

We hope that the hybrid technique proposed here stimulates further work on architectural solutions to temperature-aware design. A variety of important areas remain for future work. A figure of merit is needed to help in analyzing DTM performance and cooling capability. New architectural techniques may allow produce even better DTM performance. Techniques for predicting thermal stress and responding proactively, rather than waiting for actual thermal stress and responding reactively, may further reduce the overhead of DTM [19]. Thermal management on multi-threaded and multi-core systems remains poorly understood. And our results in combining microarchitectural techniques and DVS suggest the potential value of hybrid DTM for thermal management in globally asynchronous/locally synchronous processors with independent voltage domains, like [10, 14].

Acknowledgments

This work is supported in part by the National Science Foundation under grant no. CCR-0133634, a grant from Intel MRL, and an Excellence Award from the Univ. of Virginia Fund for Excellence in Science and Technology. I would also like to thank Mircea Stan, Karthik Sankaranarayanan, Wei Huang, and the anonymous reviewers for their helpful comments.

References

[1] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, pages 23–29, Jul.–Aug. 1999.
[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. HPCA-7*, pages 171–82, Jan. 2001.
[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. ISCA-27*, pages 83–94, June 2000.
[4] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. *ACM SIGARCH CAN*, 25(3):13–25, June 1997.
[5] M. Fleischmann. Crusoe power management: Cutting x86 operating power through LongRun. In *Embedded Processor Forum*, June 2000.
[6] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Tech. J.*, Q1 2001.
[7] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *ISLPED 2003*, Aug. 2003.
[8] W. Huang, J. Renau, S.-M. Yoo, and J. Torellas. A framework for dynamic energy efficiency and temperature management. In *Proc. Micro-33*, pages 202–13, Dec. 2000.
[9] Intel Corp. *Mobile Pentium III processor in BGA2 and Micro-PGA2 pacakages*, 2001. Datasheet Order no. 283653-002.
[10] A. Iyer and D. Marculescu. Power and performance evaluation of globally asynchronous locally synchronous processors. In *Proc. ISCA-29*, pages 158–68, May 2002.
[11] C.-H. Lim, W. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. In *Proc. ISQED*, pages 517–22, Mar. 2002.
[12] M. Ma et al. Enhanced thermal management for future processors. In *Proc. of the 2003 Int'l Symp. on VLSI Circuits*, pages 201–04, June 2003.
[13] H. Sanchez et al. Thermal management system for high-performance PowerPC microprocessors. In *Proc. COMPCON*, page 325, 1997.
[14] G. Semeraro et al. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proc. HPCA-8*, pages 29–40, Feb. 2002.
[15] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proc. PACT*, Sept. 2001.
[16] SIA. *International Technology Roadmap for Semiconductors*, 2001.
[17] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. ISCA-30*, June 2003.
[18] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture: Extended discussion and results. Technical Report CS-2003-08, U.Va. Dept. of Computer Science, Apr. 2003.
[19] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proc. 17th ICS*, June 2003.
[20] R. Viswanath, W. Vijay, A. Watwe, and V. Lebonheur. Thermal performance challenges from silicon to systems. *Intel Tech. J.*, Q3 2000.