A Computer-Architecture Approach to Thermal Management in Computer Systems: Opportunities and Challenges

†Kevin Skadron, ‡Mircea R. Stan, ‡Wei Huang, †Karthik Sankaranarayanan, ‡Zhijian Lu, ‡John Lach
†Dept. of Computer Science, ‡Dept. of Electrical and Computer Engineering
University of Virginia, Charlottesville, VA, 22904, USA
E-mail: {skadron,karthick}@cs.virginia.edu, {mircea,wh6p,zl4j,jlach}@virginia.edu

Abstract

Cooling costs for notebook, desktop, and server computer systems are rising exponentially as power densities for high-performance chips continue to double every three years. Research has led to a range of advances in modeling and design of thermal packaging and circuit boards. Yet a major front that has been absent is thermal design in the computer architecture *domain, where processor utilization, the interleaving of different computation processes, and the flow of instructions through the CPU are controlled. The architecture domain presents a rich opportunity for thermal management that complements other advances by controlling temperature* at runtime *in response to the dynamic behavior of the computer's current workload. This paper describes a compact modeling algorithm that is appropriate for this domain, presents some architecture techniques for runtime thermal management, and describes some interesting problems that remain to be solved.*

1. Introduction

The architecture domain is unique in its ability to use runtime knowledge of application behavior and the thermal status of the chip to control execution rate, distribute the workload, and extract instruction-level parallelism (ILP). On-chip temperature sensors can provide information about local hot spots and temperature gradients that can be combined with dynamic information about ILP in order to precisely regulate temperature while minimizing performance loss.

The main focus of research on temperature-aware architecture to date has been on *dynamic thermal management* or *DTM*. DTM recognizes that if the thermal package is designed for worst-case power dissipation, it must be designed for the most severe hot spot that could potentially arise. Yet these worst-case scenarios are rare and lead to over-design. A less expensive package can be employed, designed for the worst "typical" or "interesting" workload. Excessive heat dissipation must then be handled by an autonomous, runtime response in the chip itself that guarantees to reduce power densities far enough and fast enough to maintain temperature regulation. For example, the Intel Pentium 4 [12] follows this approach, designing the thermal package for a power density 20% less than the theoretical worst case. If environmental or workload characteristics cause temperatures to rise too close to the maximum allowable, the Pentium 4 responds by stopping the clock, and hence all dynamic power, until temperature returns to a safe level [15]. As another example, the Transmeta Crusoe uses dynamic voltage scaling to reduce power density when temperatures rise too high [10]. Yet current techniques such as these are too simplistic and impose unnecessary performance penalties. Architectural design can take advantage of instruction-level parallelism to implement DTM with substantially less performance overhead. This paper describes three techniques, "temperature-tracking dynamic frequency scaling," "migrating computation" [29], and "hybrid DTM" [26], that reduce overhead by 25–35% compared to techniques like the Pentium 4's and the Transmeta Crusoe's.

Recent work in the architecture community [2, 4, 13, 14, 19, 21, 23, 27, 31] demonstrates growing interest in thermal management and shows that architectural techniques offers substantial benefits. Accurately characterizing thermal behavior and evaluating architectural techniques requires an appropriate thermal model. An effective model for architects must be simple enough to allow them to reason about thermal effects and tradeoffs; parameterized; capable of modeling runtime changes in temperature within different functional units on the die; and yet computationally efficient and portable.

We have developed and publicly released a dynamic compact model—*HotSpot* [29]—that achieves all of these requirements. It differs from prior dynamic compact models for computer chips in that this new model requires less design detail and hence can be used earlier in the design cycle. It has been validated both against an independent finite-element model and a test chip. It also accounts for imprecision due to sensor noise and placement.

The rest of this paper is organized as follows. The next section describes some example architectural thermal-management techniques and presents preliminary results evaluating their potential benefits. Section 3 describes the proposed modeling approach. Then Section 4 describes what we see as some important research questions in this area, and Section 5 concludes the paper.

0-7803-8420-2/04/$ 20.00 ©2004 IEEE

2. Architectural Thermal Management

2.1. DTM Techniques

Several existing and proposed architectural techniques can provide runtime temperature regulation. Here, we describe and evaluate just a few of them here; additional techniques are described in [28]. The evaluation is done via simulation with various industry-standard uniprocessor benchmark programs from the SPECcpu2000 suite [33]. These are real programs with realistic inputs and data sets representing a range of integer and floating-point programs. We have chosen the nine hottest benchmarks, ranging from the *gcc* compiler and two data-compression programs to several numerical computations from the sciences. The remaining 17 SPEC benchmarks generate less heat, with some running 10-20° cooler. The CPU is modeled at cycle-level accuracy using the SimpleScalar simulation toolkit [6]: that is, the flow of instructions is tracked through each stage of the CPU's pipeline, including faithful modeling of all stall cycles. This simulation obviously runs vastly slower than real hardware; simulating the SPEC benchmarks to completion takes days or weeks, so we instead simulate 500M-instruction samples (about 75–150 msec., whereas interesting thermal behavior develops over timescales of 10s of msec.) that have been determined representative according to the methodology of Sherwood et al. [24]. We have customized the simulator to model an Alpha 21264/21364 [16], minus its multiprocessor logic, which is not used for these programs. The pipeline simulation is used to derive activity factors that drive the Wattch [5] power model. These power statistics are then used to track dynamic power densities, which are the inputs to the proposed thermal model.

These simulations model the 0.13μ version of the 21364 at 1.3 V and 3 GHz. The floorplan of our modeled processor is shown in Figures 1a and b. Simulations assume a CMOS temperature sensor located in the center of each architectural block with random noise of ±1°C after averaging and ±2°C offset error. We also allow an extra 0.2°C margin to accommodate the time required for the DTM technique to be engaged—this assumes an on-chip trigger mechanism as opposed to interrupts, which are much slower. The maximum tolerated temperature for these simulations is 85°C, in accordance with projections in the 2001 International Technology Roadmap [25]. This means that DTM must be engaged whenever the temperature exceeds 81.8°C. Of course, reliability concerns are not serious at such low temperatures, and in reality 85°C need not be rigidly enforced as long as timing violations can be prevented. We use this threshold for studying all the DTM techniques so that their potential performance overhead can be compared under the same conditions.

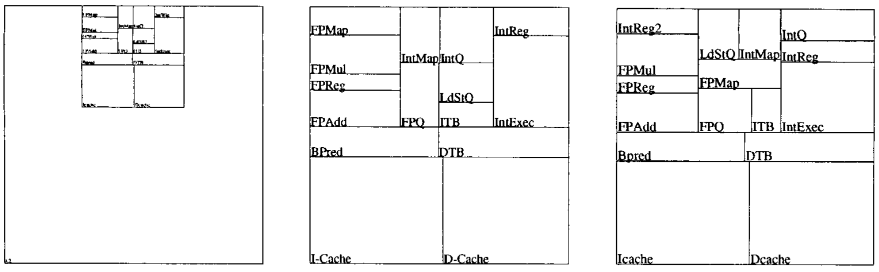
Temperature-Tracking Dynamic Frequency Scaling (TTDFS) was proposed in [29] and recognizes that carrier mobility and hence frequency are linearly dependent on the operating temperature. Garrett and Stan [11] report an 18% variation over the range 0-100°C. This suggests that the standard practice of designing 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. When reliability is not a concern, this is the technique with the least overhead. These simulations assume that any time the frequency is adjusted, resynchronizing the phase-locked loop (PLL) takes 5 $\mu sec.$, and the processor is stalled during this time.

Dynamic Voltage Scaling (DVS) reduces voltage and hence frequency in response to high temperatures. Because $P \propto CV^2 f$, this yields a cubic reduction in power density relative to the reduction in frequency (i.e., performance). These simulations assume that any time the DVS setting is changed, the processor must stall for 5 $\mu sec.$ for the PLL and another 5 $\mu sec.$ to change the voltage. In these simulations, only two voltage settings are available. The low setting is 1.1V.

Fetch Gating (FG) was proposed in [4] and reduces the rate at which the processor can fetch instructions. This reduces activity factors within the pipeline. It has the advantage that the ability of a superscalar, out-of-order execution processor like the Alpha can exploit instruction-level parallelism (ILP) and compensate for the performance impact of some of these wasted fetch cycles. The ratio of fetch vs. non-fetch cycles is determined by a proportional-integral feedback controller.

Hybrid DTM (Hyb) was proposed in [26] and combines DVS and fetch gating. The advantage of fetch gating at low levels of thermal stress is that the fetch duty cycle remains high and the availability of ILP compensates for the impact of wasted fetch cycles. In contrast, DVS incurs a performance loss proportional to the reduction in frequency. At higher levels of thermal stress, more fetch cycles are skipped and ILP can no longer compensate. At this point DVS becomes more attractive because of the cubic relationship between power density and performance loss.

Migrating Computation (MC) was proposed in [29] and employs additional spare units located in colder parts of the chip, for use when a primary unit overheats. In our simulations the integer register file is always the hottest, so here we consider a spare register file. The location of the spare register file and the requisite changes to the CPU's floorplan are shown in Figure 1c. When the primary register file reaches 81.6°C, the processor stalls and drains the pipeline, and the register file is copied to



(a) Equiv. floorplan

(b) CPU core

(c) CPU core

Figure 1. (a) Floorplan approximating the 21364 as modeled by our experiments. (b) Closeup of CPU floorplan. (c) Floorplan with a spare integer register file, for use with the MC technique.

the spare register file. 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, and the power model has been augmented to account for the extra power due to the long cross-chip wires and for copying the register values. When the primary register file returns to 81.5°C, the process is reversed and computation resumes using the primary register file. Note that, because there is no way to guarantee that MC will prevent thermal violations, a failsafe mechanism is needed, for which we use fetch gating. Also note that MC adds extra chip resources; raising an opportunity-cost question of whether that additional area is better used for spare resources or enhancing the core architecture. Finally, note that migrating among single units is but one, small-granularity version of this "migration" philosophy. Migration can also occur among pipeline clusters [7] or distinct CPUs that share the same die [13].

2.2. Results

Figure 2 gives the performance overhead (relative to no DTM) for these DTM techniques with a 1.0 K/W thermal package (including the equivalent sink-to-air convection resistance). DTM always incurs some slowdown because it operates by reducing power density, which usually entails some technique that reduces performance. Note that the absolute magnitude of the slowdown is highly dependent on the package and operating voltage and frequency that are modeled, but the relative behavior of the different DTM techniques remains fairly similar to the specific data reported here.

The best technique for thermal management by far is TTDFS. The performance penalty for even the hottest benchmarks is small; the worst is *art* with only 5.4% slowdown for TTDFS. If the maximum junction temperature of 85°C is strictly based on timing concerns, and somewhat higher temperatures than 85°C 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.

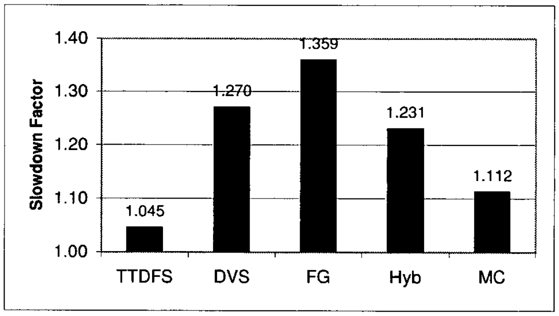


Figure 2. Slowdown factor for DTM relative to execution with no DTM and no limitation on maximum temperature.

If the junction temperature of 85°C were dictated not only by timing but also physical reliability, then TTDFS is not a viable approach, as the specified junction temperature must be enforced. All the remaining techniques do this, and among these, hybrid DTM and migrating computation are the best. DVS performs poorly due to the overhead incurred on each change in voltage. As this overhead is reduced, DVS becomes more attractive. In an ideal case in which DVS incurs no overhead, its performance loss is reduced to 8.5%. 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.

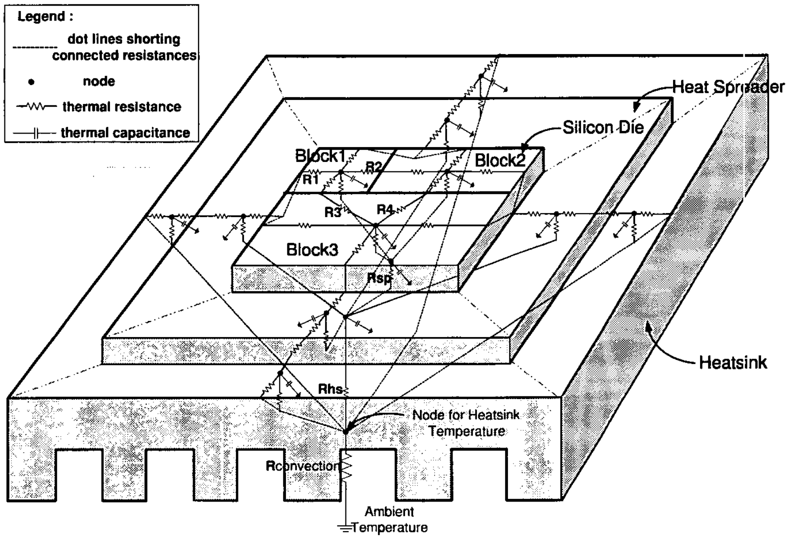


Figure 3. Example HotSpot RC model for a floorplan with three architectural units, a heat spreader, and a heat sink. The RC model consists of three layers: die, heat spreader, and heat sink. Each layer consists of a vertical RC pair from the center of each block down to the next layer and a lateral RC pair from the center of each block to the center of each edge.

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 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%.

This section has only been a quick survey. These simulation results demonstrate some of the possible architectural approaches to runtime thermal management, their relative effectiveness, and some of the implementation issues that arise.

3. A Dynamic Compact Model for Architects

3.1. Model Description

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 parameterized, 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.

Compact models are the most common way to model thermal phenomena in computer systems, although computational fluid dynamics using finite-element modeling is often performed when the flow of air or a liquid is considered. An excellent survey of these modeling techniques is given by Sabry in [22]. Batty *et al.* [1], Cheng and Kang [8], Koval and Farmaga [17], Székely *et al.* [20, 34], and Torki and Ciontu [35] all describe techniques for modeling localized heating within a chip due to different power densities of various blocks, but none of these tools are easily adapted to architectural exploration for a variety of reasons. Architectural modeling typically precludes direct thermal-response measurements, *e.g.* [34, 35], the use of analytic power models obviates the need for joint electro-thermal modeling, *e.g.* [34], and these techiques typically depend on low-level VLSI netlists and structural implementation details or only give steady-state solutions, neither of which are compatible with the type of architecture studies we wish to enable.

The model we propose was first described in primitive form in [27], expanded to account for lateral heat transfer and package effects in [29, 32], and most recently refined in [28] and [30]. It accomplishes its design goals by taking advantage of the duality between thermal and electrical systems. Each unit on the chip is modeled as a heat (power) dissipater; 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. It 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.

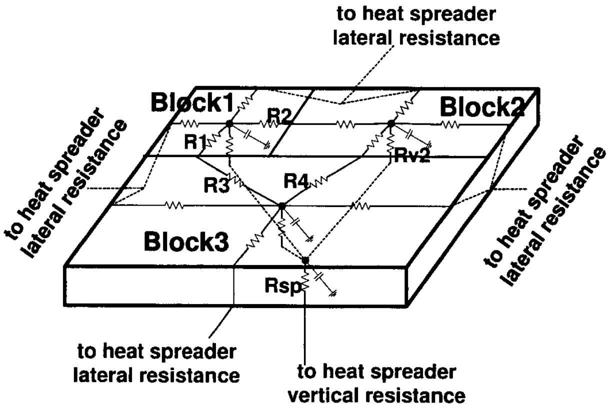


Figure 4. The RC model for just the die layer.

The equivalent circuit—see Figure 3 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 vertical, conductive layers for the die, heat spreader, heat sink, interface materials, and a final vertical, convective layer for the sink-to-air interface. For each layer except the convective layer, 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 4 accounts for heat flow from Block 2 into the interface material between the die and 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). The die layer is divided into blocks that correspond to the microarchitectural blocks of interest and their floorplan. For simplicity, the example in Figure 3 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 ($R_{convection}$). Air is assumed to be at a fixed ambient temperature. Because the perspective in Figure 3 makes it somewhat difficult to distinguish vertical and lateral R's, Figure 4 shows the RC model for only R's in the die layer. Note that all thermal resistances must account for spreading and constriction resistances. Note also that we have chosen not to divide the spreader's center section into blocks corresponding to each block in the die. This is a simplification permitted by the fact that the vertical resistance for the spreader is much less than for the die, so the temperature gradient across its far surface is fairly small. We also 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, etc. The time scales for these is much larger than the short time scales at which we envision DTM's occurring. Of course, for studies of temperature-aware process scheduling or DTM on a system-wide scale, these effects would need to be accounted for.

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. The exact derivation of the RC model is described in further detail in [28].

Another version of this model has been developed that uses a regular grid rather than the architecture-dependent blocks as described above. This entails more computational cost but improves spatial resolution and is helpful for work involving spatial gradients. It is also likely to be more useful for integration into some types of CAD tools. Certainly a grid-based model can be obtained by specifying a grid as the floorplan for the previously described architectural model, but knowing at the outset that the model is constrained to a grid helps improve the efficiency of the solver and thus helps offset the extra computational overhead of the finer resolution.

3.2. Validation

Validation of the extended compact thermal model is performed by comparing the estimated silicon surface temperatures from the model against a commercial thermal testing chip [3]. The thermal testing chip has a 9x9 grid of power dissipating resistors, which can be turned on or off individually. Each cell also contains a temperature sensor. The sensors can be used to measure both steady-state and transient temperatures for each of the cells. We build the same 9x9 grid-like chip structure in our thermal model. In our validation experiments, we turn on specific sets of power dissipaters in the testing chip and assign exactly the same power values at the same locations in the thermal model. Figure 5 shows the steady-state thermal plots using measurements from the testing chip and results from our thermal model. Transient temperature data from the thermal model is also compared with the testing chip transient measurements, see Figure 6. Power density in this experiment is $50\text{W}/\text{cm}^2$ in the heat dissipating area. As can be seen, our compact thermal model is reasonably accurate, with the worst case percentage errors for steady-state temperatures and transient temperatures less than 5% and 7%, respectively.

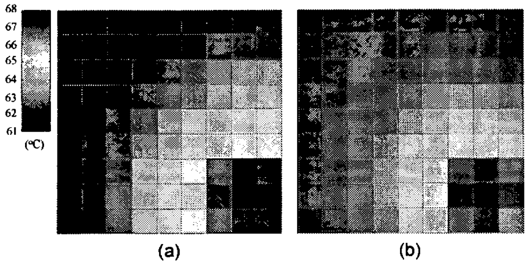


Figure 5. Steady-state validation of the compact thermal model—(a) Testing chip measurements (b) Results from the model with errors less than 5%. (Lower right 3x3 dissipators are turned on with 0.5W/mm^2.)

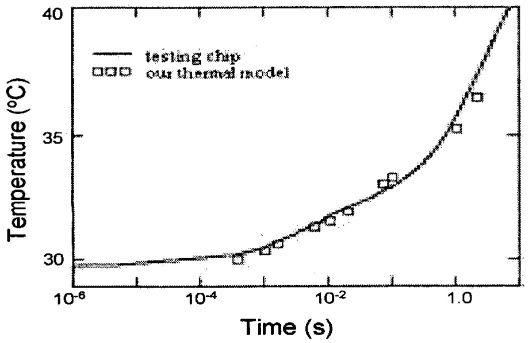


Figure 6. Transient validation of the model (one power dissipator is shown here). Percentage error is less than 7%.

Although further validation is required against real microprocessors, these results suggest that the proposed model has correctly accounted for the major heat-flow paths, and can be used with confidence in lieu of more computationally expensive approaches. Each recomputation of temperatures using the model takes approximately $50\,\mu sec.$, and with a simulated time step of $3\,\mu sec.$ or larger, thermal modeling adds less than 1% to processor simulation times with our SimpleScalar/Wattch simulator.

3.3. Future Modeling Needs

The proposed model can be used to estimate how different workloads affect the spatial distribution of heating across the chip over time. Extensions to multiple processors ("cores") on a chip and heterogeneous systems-on-a-chip should be straightforward extensions. To study system-wide heating patterns and runtime temperature management in the system as a whole, however, requires a model of heat dissipation in other major components, or at least knowledge of their average operating temperatures.

Even at the chip level, further accuracy can be obtained by accounting for self-heating in wires, and localized heating effects due to I/O pads and heat flow through the pins. So far we have neglected these effects, assuming that their heat contributions are roughly uniform. To model these effects requires a more detailed estimate of the layout and routing of the chip, and validation will require a more sophisticated test chip or the ability to perform infrared imaging.

Because the proposed model is intended to be useful during early planning stages, before detailed designs—let alone layout and routing—are available, modeling the impact of interconnect requires some *a priori* estimate of wire-length distributions and via locations. One possibility is to manually determine connectivity among high-level architectural units, approximate wire locations and lengths, and approximate via locations, based on a preliminary floorplan (this might be feasible for global interconnect and the power/ground/clock planes); another is to adopt a model like the one based on Rent's Rule, proposed by Davis et al. [9]. With information about wires in hand, we can estimate self-heating power according to $P_{self} = I^2R$. Total current through the wiring net is given by $Power/V_{dd}$.

As for I/O pads, a similar problem exists in early design stages with estimating each pad's activity. Heat transfer from I/O pads to PCB can be modeled as a series of thermal RC pairs, each of which represents the thermal resistance and capacitance of pad-bumps/underfill, ceramic substrate, ball/lead array, and PCB convection. As for the derivation, R's and C's can be calculated in a similar way as other R's and C's in the model.

4. Future Challenges

Work on temperature-aware architecture is still in its infancy, and a host of challenges remain. The two that we see as most interesting are to develop DTM algorithms for a variety of systems, especially embedded systems, systems with multi-core and multi-threaded chips, and all types of real-time systems; and techniques for managing reliability as a function of temperature at runtime.

Unlike high-performance desktop, server, and laptop processors, where the processor is typically the major source of heat (with the graphics processor sometimes a close second), embedded systems often use low-cost, low power chips, are often I/O bound, and in the case of wireless devices, RF activities may dominate. We suspect that the philosophy of autonomous runtime temperature management, in lieu of worst-case cooling design, can be applied to other components besides the CPU. Within conventional, high-performance computer systems, the advent of multi-core and multi-threaded chips means that multiple programs, possibly with very different characteristics, adds new dimensions to the design of DTM. DTM can now respond to thermal stress by altering the way the system schedules its workload, and in the case of multicore chips, by migrating hot applications from core to core to distribute their heat dissipation over a wider area. One version of this migration was recently proposed by Heo et al. [13]. Temperature-aware scheduling has been studied for single-threaded processors [2, 21] but not for multi-threaded systems.

Real-time systems are especially challenging from the standpoint of implementing DTM, because the maximum slowdown due to DTM must be bounded and factored into the scheduling. In many real-time systems, however, the behavior of the workload can be well characterized. Srinivasan and Adve [31], for example, have used knowledge about MPEG frame characteristics to develop predictive DTM algorithms with lower overhead than conventional reactive techniques. Another mitigating consideration is that in soft-real-time applications, occasional deadline misses can be tolerated.

Circuit reliability is heavily dependent upon operating temperature, creating a reliability dimension to runtime temperature management. For example, interconnect electromigration and gate-oxide breakdown rates both increase with temperature. The current approach to handling the reliability-temperature relationship is to designate a worst-case operating temperature that may never be exceeded, which unnecessarily hampers system performance. By instead modeling circuit lifetime as a resource that is consumed over time by temperature, dynamic thermal management techniques can be developed that maximize circuit performance while still meeting reliability requirements. For example, a system may allow the previously designated maximum operating temperature to be exceed for a short time (thereby increasing the circuit lifetime consumption rate), so long as the circuit lifetime is later compensated by lower operating temperatures. This would be particularly useful in realtime workloads. Naturally, reliability is a probabalistic phenomenon, so these calculations must be done in terms of expected value. There are significant opportunities for exploring such reliability-aware dynamic thermal management techniques. We are currently in the process of developing reliability models that can track changes in expected lifetime while the system runs, accounting for changes in the spatial temperature map over time. As Lasance has noted [18], both spatial and temporal gradients are clearly important in determining reliability.

5. Conclusions

This paper argues that chips can and should manage their operating temperature at runtime in order to improve performance and reduce cooling costs. This argument is motivated by the fact that worst-case operating conditions are rarely present, and hence worst-case design is unnecessary and wasteful. Instead, the chip can operate at a higher performance level and/or use a less expensive cooling solution; when the workload or operating conditions exceed the cooling solution's capacity, the chip can autonomously reduce its power dissipation (albeit with some possible loss in performance) to maintain safe temperatures. This sets up a tradeoff between performance gains for average-case behavior and performance loss for worstcase behavior. Simulation results with realistic benchmarks suggest that several dynamic thermal management techniques can keep the performance loss for hot workloads reasonably small.

To study dynamic thermal management—especially in the architecture domain, where much work must be done before detailed design can begin—requires a dynamic compact model that can be derived from material, physical, and geometric values. We described such a model and its validation, and suggested several possible avenues for further research to improve the precision of this approach. The current version of the HotSpot model is publicly available at http://lava.cs.virginia.edu/HotSpot.

Acknowledgments

This work is supported in part by the U.S. National Science Foundation under grant nos. CCR-0133634, CCR-0105626, grants from Intel MRL, and an Excellence Award from the Univ. of Virginia Fund for Excellence in Science and Technology.

References

[1] W. Batty et al. Global coupled EM-electrical-thermal simulation and experimental validation for a spatial power combining MMIC array. *IEEE Transactions on Microwave Theory and Techniques*, pages 2820–33, Dec. 2002.

[2] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel. Event-driven energy accounting for dynamic thermal management. In *Proceedings of the 2003 Workshop on Compilers and Operating Systems for Low Power*, Sep. 2003.

[3] Z. Benedek, B. Courtois, G. Farkas, E. Kollár, S. Mir, A. Poppe, M. Rencz, V. Székely, and K. Torki. A scalable multi-functional thermal test chip family: Design and evaluation. *Transactions of the ASME, Journal of Electronic Packaging*, 123(4):323–30, Dec. 2001.

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

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

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

[7] R. Canal, J.-M. Parcerisa, and A. González. A cost-effective clustered architecture. In *Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques*, pages 160–68, Oct. 1999.

[8] Y.-K. Cheng and S.-M. Kang. A temperature-aware simulation environment for reliable ULSI chip design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1211–20, Oct. 2000.

[9] J. A. Davis, V. K. De, and J. D. Meindl. A stochastic wire-length distribution for gigascale integration (G SI)—part I: Derivation and validation. *IEEE Transactions on Electron Devices*, 45(3):580–89, Mar. 1998.

[10] M. Fleischmann. Crusoe power management: Cutting x86 operating power through LongRun. In *Embedded Processor Forum*, June 2000.

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

[12] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal*, Q1 2001.

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

[14] W. Huang, J. Renau, S.-M. Yoo, and J. Torellas. A framework for dynamic energy efficiency and temperature management. In *Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 202–13, Dec. 2000.

[15] Intel Corp. *Intel Pentium 4 Processor In the 423-pin Package: Thermal Design Guidelines*, Nov. 2000. Order no. 249203-001.

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

[17] V. Koval and I. W. Farmaga. MONSTR: A complete thermal simulator of electronic systems. In *Proceedings of the 31st Design Automation Conference*, June 1994.

[18] C. J. M. Lasance. Thermally driven reliability issues in microelectronic systems: status-quo and challenges. *Microelectronics Reliability*, 34(12):1969–74, Dec. 2003.

[19] C.-H. Lim, W. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. In *Proceedings of the International Symposium on Quality Electronic Design*, pages 517–22, Mar. 2002.

[20] M. Rencz, V. Székely, A. Poppe, and B. Courtois. Friendly tools for the thermal simulation of power packages. In *Proceedings of the International Workshop On Integrated Power Packaging*, pages 51–54, July 2000.

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

[22] M.-N. Sabry. 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*, Oct. 2002. Invited paper.

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

[24] T. Sherwood, E. Perelman, and B. Calder. 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*, pages 3–14, Sept. 2001.

[25] SIA. *International Technology Roadmap for Semiconductors*, 2001.

[26] K. Skadron. Hybrid architectural dynamic thermal management. In *Proceedings of the 2004 Design, Automation and Test in Europe Conference*, Feb. 2004. To appear.

[27] K. Skadron, T. Abdelzaher, and M. R. Stan. 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*, pages 17–28, Feb. 2002.

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

[29] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 2–13, Apr. 2003.

[30] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware computer systems: Opportunities and challenges. *IEEE Micro*, 23(6):52–61, Nov-Dec. 2003.

[31] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proceedings of the 2003 International Conference on Supercomputing*, pages 109–20, June 2003.

[32] M. R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, and S. Velusamy. Hotspot: A dynamic compact thermal model at the processor-architecture level. *Microelectronics Journal: Circuits and Systems*, 34(12):1153–65, Dec. 2003.

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

[34] V. Székely, A. Poppe, A. Páhi, A. Csendes, and G. Hajas. Electro-thermal and logi-thermal simulation of VLSI designs. *IEEE Transactions on VLSI Systems*, 5(3):258–69, Sept. 1997.

[35] K. Torki and F. Ciontu. IC thermal map from digital and thermal simulations. In *Proceedings of the 2002 International Workshop on THERMal Investigations of ICs and Systems (THERMINIC)*, pages 303–08, Oct. 2002.