Improved Thermal Management with Reliability Banking

USING A FIXED TEMPERATURE FOR THERMAL THROTTLING IS PESSIMISTIC. REDUCED AGING DURING PERIODS OF LOW TEMPERATURE CAN COMPENSATE FOR ACCELERATED AGING DURING PERIODS OF HIGH TEMPERATURE. RUNTIME TRACKING OF THE TEMPERATURE-DEPENDENT AGING RATE MEANS THAT THROTTLING IS ENGAGED ONLY WHEN NECESSARY TO MAINTAIN RELIABILITY.

Zhijian Lu
John Lach
Mircea R. Stan
Kevin Skadron
University of Virginia

The advance of technology scaling (along with resulting increases in power density) has made thermal-related reliability a major concern in modern IC design. For example, in the deep-submicron region, experts widely regard electromigration (a temperature-enhanced aging process in metal interconnects caused by the exchange of momentum between electrons and atoms) as a dominant failure mechanism. Designers must therefore rely on temperature-dependent reliability models to derive the expected lifetime of their circuits, increasing design margins (for example, wire width) as necessary to meet requirements. Traditionally, designers use a worst-case temperature to evaluate system reliability; excessive design margins are often the result.

In addition, under such pessimistic assumptions, the system might engage dynamic thermal management (DTM) techniques[1,2] (and incur performance penalties) unnecessarily. (The "Dynamic thermal management" sidebar summarizes these techniques.) In fact, during execution many programs or workloads exhibit temperature fluctuations caused by inherently phased behaviors. In this article, we show that the effect of cool (low-temperature) phases can compensate for that of hot (high-temperature) phases on reliability. Existing DTM techniques ignore the effects of temperature fluctuations on chip lifetime and can unnecessarily impose performance penalties for hot phases. The disadvantages of these techniques become more obvious in systems such as Web servers, in which hot phases usually imply an increased number of service requests. The engagement of cooling mechanisms then affects the server's quality of service.

Using electromigration as the targeted failure mechanism, we apply a dynamic reliability model (described in the "Dynamic electromigration reliability model" sidebar) and propose a dynamic reliability management (DRM) technique to dynamically track the consumption of chip lifetime during operation. Variations in operating temperature have a major impact on the expected lifetime of an IC. By accounting for such variations, we can model lifetime as a resource that is consumed over time at a temperature- and voltage-dependent rate. We use temperature

Published by the IEEE Computer Society

variability and lifetime resource models to develop novel DRM techniques that reduce performance penalties associated with existing DTM techniques while maintaining the required IC reliability lifetime. Srinivasan et al. introduced the DRM concept.[3] They proposed a chip-level reliability model and showed the potential benefits of trading reliability for performance in individual applications. They assumed an oracular algorithm for runtime management and did not consider the effects of interapplication thermal behaviors on reliability. In later work, the authors refined their reliability model and improved reliability by using redundant components.[4] Our work focuses on practical runtime management techniques for the worst-case on-chip component (the hottest interconnect) to exploit both intra- and interapplication temperature variations. Whether we can gain greater advantages by combining the Srinivasan et al. model and our techniques is open to future investigation.

Banerjee and Mehrotra[5] showed that temperature-induced reliability problems will tend to limit circuit performance in future technology generations. Therefore, in this article, we assume that the temperature threshold is set solely for reliability specification and that circuits can operate correctly above this threshold whenever banking opportunities allow. Although extreme high temperature can cause immediate thermal damage of IC circuits, we study a range of temperatures associated only with long-term reliability effects (that is, temperature-induced aging). We assume that high temperatures causing immediate damage are far above the temperature range studied here, and we assume a monitoring and feedback mechanism has been implemented in the circuits (but not by us) to assure that circuits operate far below the temperatures causing short-term damage.

DRM based on lifetime banking

In general, an increase in temperature means that the chip is consuming its lifetime more rapidly, and vice versa. Therefore, if temperature has been below the traditional DTM engagement threshold for an extended period, thus accumulating surplus lifetime, it is acceptable to consume this surplus, exceeding the threshold for a time while maintaining the required expected lifetime. In effect, we model lifetime as a resource "banked" during low-temperature periods, allowing future withdrawals to maintain performance during times of higher operating temperatures. The chip benefits from lifetime banking by avoiding unnecessary DTM engagements while meeting expected lifetime requirements.

Lifetime banking opportunities

Because of activity variation, the power consumption of on-chip components (caches, floating-point or integer units, branch predictors, and so forth) is not constant. Therefore, a chip's thermal characteristics include not only cross-chip spatial temperature gradients but also temporal temperature gradients for each component. Here, we focus on temporal temperature gradients. Although a

Dynamic thermal management

Worst-case power dissipation and environmental conditions are rare for general-purpose microprocessors, so designing a cooling solution for the worst case is wasteful. Instead, the cooling solution should target the worst expected case. If environmental or workload conditions exceed the cooling solution's capabilities and temperature rises to a dangerous level, on-chip temperature sensors can engage some form of dynamic thermal management (DTM).

Most commercial microprocessors now employ this design philosophy; the challenge is to minimize performance overhead. The Intel Pentium 4 throttles the clock, Transmeta's Long Run scales back voltage and frequency, and the PowerPC G3 throttles instruction throughput. These were among the first processors announced as having DTM, although previous processors had forms of DTM to protect against major failures, such as heat-sink detachment. Other known DTM techniques include scaling only frequency and raising threshold voltage. Voltage-based techniques are attractive because they provide superlinear reductions in heat dissipation relative to their performance loss. But they usually entail discrete steps and stall time to safely switch voltages. This makes hybrid techniques attractive—for example, using a lightweight technique such as fetch throttling when a voltage step is too expensive and adjusting voltage only in response to more severe thermal stress.[1]

All the preceding techniques spread excess heat in time. An alternative is to spread heat in space. This is accomplished through load balancing across duplicated resources such as register banks, ALUs, pipeline clusters, or even entire CPUs.[2] Depending on how much area is available for extra resources and how much extra latency is required to access them, space-based techniques impose less performance overhead than time-based techniques.[1]

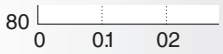
References

K. Skadron et al., "Temperature-Aware Microarchitecture: Modeling and Implementation," *ACM Trans. Architecture and Code Optimization*, vol. 1, no. 1, Mar. 2004, pp. 94-125.

S. Heo et al., "Reducing Power Density through Activity Migration," *Proc. Int'l Symp. Low Power Electronics and Design* (ISLPED 03), IEEE Press, August 2003, pp. 217-222.

Dynamic electromigration reliability model

Electromigration is an aging process in metal interconnects caused by the exchange of momentum between electrons and atoms. In this process, atoms are aggregated in specific locations in the metal lines and devoid in other locations, finally causing open- or short-circuit failures. Black's equation[1] is widely used to predict mean time to failure caused by electromigration:



(b)

where T_f is time to failure, A is a constant based on interconnect geometry and material, j is current density, Q is activation energy (for example, 1.0 eV for copper interconnect), and kT is thermal energy. Current exponent n is between 1 and 2, according to the actual failure mechanism. However, Black's equation is suitable only for interconnects subject to constant temperature and current density.

We derived a model to predict interconnect lifetime because of electromigration under simultaneous dynamic thermal and current stresses.[1] According to this model, we express mean time to failure as

$$\int_0^{T_f} \frac{\exp\left(-Q / kT(t)\right)}{kT(t)} dt = D \qquad (1)$$

where D is a constant determined by the interconnect structure, independent of the reliability management scheme. Here we assume void-growth-dominated failures, which are common in copper interconnect technology.[2]

Equation 1 models interconnect time to failure as a resource consumed by the system over time. Specifically, we can regard function $r(t)$

$$r(t) = \left[j(t) \left(\frac{\exp\left(-Q / kT(t)\right)}{kT(t)} \right) \right]$$

as the consumption rate, and it represents the void growth rate in the interconnect. This model captures the effect of transient behaviors (temperature and current) on electromigration lifetime.

In our chip-level reliability model, for simplicity, we use the maximum temperature measured across the chip to calculate the consumption rate, and we use the worst-case current density specified at design time for our calculations. Future refinement of our model will track the variability of current density across the chip, using the activity factors provided by the architectural simulator.

Dynamic voltage and frequency scaling (DVS) is a widely used effective technique for active cooling. When DVS is applied, the worst-case current density in the IC interconnects should be scaled accordingly by $j \propto = CVf$, where C is the effective capacitance.[3] Therefore, when we switch the chip to a new voltage and frequency setting, we scale the corresponding worst-case current density by the product of the new voltage and frequency, and thus track current density dynamically.

References

1. Z. Lu et al., "Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design," *Proc. Int'l Conf. Computer Aided Design* (ICCAD 04), IEEE Press, Nov. 2004, pp. 327-334.
2. S.V. Nitta et al., "Copper BEOL Interconnects for Silicon CMOS Logic Technology," *Interconnect Technology and Design for Gigascale Integration*, J.A. Davis and J.D. Meindl, eds., Kluwer, 2003, Chap. 2, pp. 48-55.
3. D.T. Blaauw et al., "Static Electromigration Analysis for On-Chip Signal Interconnects," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 1, Jan. 2003, pp. 39-48.

more accurate reliability model would incorporate all on-chip components, for simplicity in this study, we safely use the temperature of the chip's hottest unit.

Figure 1 depicts the temperature profiles of two different workloads common in general-purpose computing. Figure 1a represents a single-program workload, and Figure 1b a multiprogram workload with context switching. In the single-program workload, temperature changes over time because of the executed program's phased behavior. In the multiprogram workload, not only execution variations in each program but also interprogram thermal differences affect overall thermal behavior. The workload in Figure 1b consists of one cold program (applu) and one hot program (gcc). This shows that context-switching changes the temperature variation to exhibit more noise and usually lower amplitude.

Our dynamic reliability model reveals that the DC component (average value) in the temperature profile approximately equals the reliability-equivalent temperature (that is, the lifetime at that constant temperature equals the lifetime projection in the temperature profile), as shown in Figure 1a. When the actual temperature is lower than the reliability-equivalent temperature, the chip consumes its lifetime at a slower speed, allowing subsequent execution above the reliability-equivalent temperature.

Reliability-aware runtime management

Chip designers usually specify an expected chip lifetime (such as 10 years) under particular operating conditions (temperature, cur-

rent density, and so forth). We use $r_{nominal}$ to denote the lifetime consumption rate under nominal conditions (for example, the reliability-constrained temperature threshold). During runtime, we monitor the actual operating conditions regularly, calculate the actual lifetime consumption rate, $r(t)$, at each monitored time instance. We then compare this actual rate with $r_{nominal}$ by calculating $\int [r_{nominal} - r(t)]\,dt$, which we call the lifetime banking deposit. When $r(t) < r_{nominal}$, the chip is consuming its lifetime at a slower than nominal rate. Thus, the chip's lifetime deposit increases. When $r(t) > r_{nominal}$, the chip is consuming its lifetime faster than nominal, and the lifetime banking deposit decreases. As long as the lifetime deposit is positive, the expected lifetime will not be shorter than that under $r_{nominal}$, which is specified at design time.

Figure 2 illustrates this technique, which we call simple dynamic reliability management (SDRM). For example, in interval [$t0$, $t1$], the chip's reliability is banked, while in [$t1$, $t2$], the banking deposit is consumed. At time $t2$, the banking derosit becomes less than some threshold, and the system must engage a cooling mechanism to quickly pull down the lifetime consumption rate to the nominal rate, just as in conventional DTM techniques. Therefore, our SDRM technique adopts DTM as a bottom-line guarding mechanism.

DTM never allows the lifetime consumption rate to be higher than the nominal. In SDRM, before we engage thermal management mechanisms, we first check whether the chip currently has a positive lifetime balance. If enough lifetime has been banked, the system can afford to run with a lifetime consumption rate higher than the nominal rate. Otherwise, we apply a DTM mechanism to lower the consumption rate, thus preventing a negative lifetime balance. In our study, we use dynamic voltage or frequency scaling as the major DTM mechanism. Because SDRM must only monitor the actual lifetime consumption rate and update the lifetime banking deposit, its computation overhead is negligible compared with that of DTM.

Experiments and analysis

We evaluated the SDRM technique for general-purpose computing workloads, including single- and multiprogram workloads.

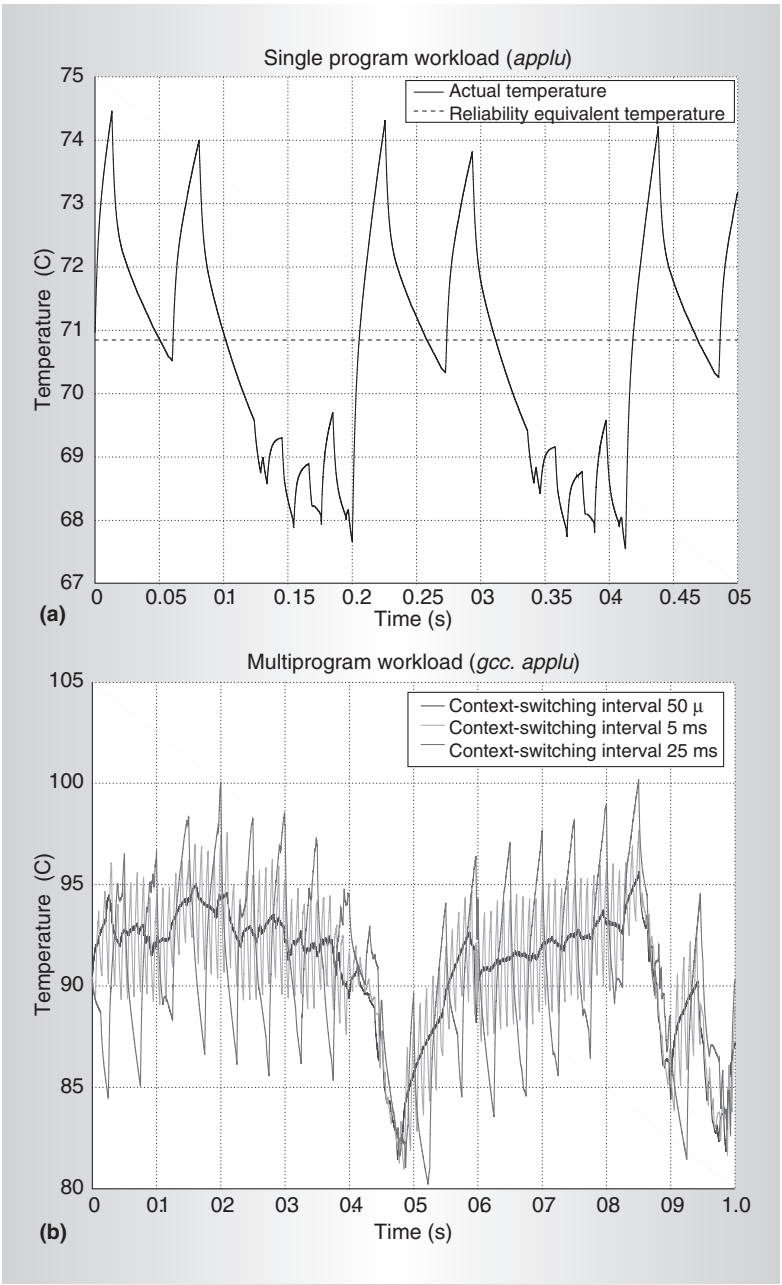


Figure 1. Temporal temperature variation: single-program (applu) workload (a) and two-program (applu and gcc) workload with context switching (b).

Experimental setup

We ran a set of programs from the SPEC2000 benchmark suite on the SimpleScalar processor simulator[6] with characteristics similar to those of a 0.13-micron Alpha 21364. We simulated each program for a length of 5 billion instructions and obtained both dynamic and static (leakage) power traces, which we fed as inputs to Hotspot (a chip-level,

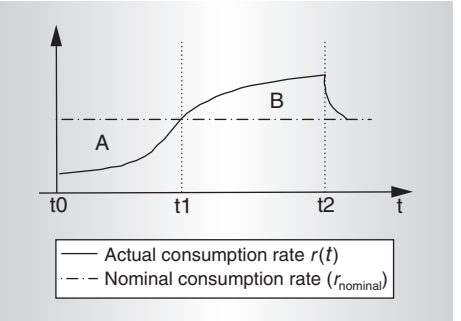


Figure 2. Simple dynamic reliability management (SDRM).

compact, thermal model) for trace-driven simulation.[2] In such simulations, we included the idle penalty of frequency or voltage switching, which is about 10 µs in many real systems,[2] and assumed that program execution time is proportional to frequency, a reasonable assumption for computation-intensive workloads. Furthermore, we scaled leakage power dynamically according to the actual temperature obtained during simulation, using a voltage- and temperature-aware leakage model.

Because the Hotspot model is highly parameterized, we could easily run experiments on a simulated processor with different thermal-package settings. For each new setting, we obtained initial temperatures by repeating the trace-driven simulation until the chip's steady temperatures converged.[2] The other parameters in our experimental setup were similar to those used by Skadron et al.[2]

We implemented DTM and SDRM in the Hotspot model and set 110°C as the temperature threshold for both techniques. Both use a feedback-controlled dynamic voltage- and frequency-scaling mechanism to guard program execution. For example, in DTM, when the actual temperature is above a certain threshold, a controller scales down the frequency or voltage as necessary, ensuring that the program never runs at a temperature higher than 110°C. The SDRM scheme uses 110°C as the nominal temperature for the lifetime consumption rate. Both techniques slow down execution when engaged, and we use this performance penalty as a metric to compare these two techniques.

Single-program workload

Figure 3a shows the performance penalty for DTM and SDRM with the same thermal configuration. The figure shows only the benchmarks subject to performance penalties from runtime management. Clearly, the performance penalty of the SDRM scheme is always less than that of the DTM scheme, when the thermal configuration is the same. On average, SDRM performance penalty is about 40 percent less that of DTM (a change of slowdown from 7 to 4 percent).

Figure 3a also shows the performance of DTM with a more expensive thermal package whose convective thermal resistance is only one-third of the other's. As you might expect, a more expensive thermal package can reduce the performance penalty. Figure 3a shows that, on average, SDRM with a higher thermal resistance achieves a performance very close to that of DTM with a lower thermal resistance. These results imply that, if we set a fixed tolerable performance loss, SDRM lets us use a much cheaper thermal package than that required by conventional DTM.

In addition, with the SDRM technique, we can explicitly trade reliability with performance by targeting different lifetime budgets. For example, we can increase the nominal lifetime consumption rate when the system allows a reduction of the lifetime target. Figure 3b plots SDRM performance averaging over all benchmarks at different lifetime budgets, with shorter expected lifetimes enabling faster execution. However, reducing lifetime 10 percent increases performance only about 1 percent.

Multiprogram workload

Another interesting program execution scenario is a workload of multiple programs with context switching between them. When a hot and a cold benchmark are executed together, the average operating temperature should be between the two benchmarks' operating temperatures. For example, gcc's operating temperature is around 115°C, and applu's is around 70°C. We showed in Figure 1b the temperature profile of a hybrid workload composed of gcc and applu, with different context-switching time intervals.

As you would expect, the smaller the context-switching interval, the less temperature fluctuation, with the chip's thermal package working as a low-pass filter. When the context-switching interval increases, individual benchmarks

can show their hot or cold properties, and the workload's temperature variation becomes obvious. To investigate how multiprogram workloads affect DTM and SDRM performance, we reduced the temperature threshold of the targeted lifetime from 110°C to 90°C. Figure 3c shows the DTM and SDRM performance penalties for the multiprogram workload with different context-switching intervals. We observe a trend similar to what we saw for the single-program workload (Figure 3a). SDRM outperforms DTM with the same thermal-package configurations. As the context-switching interval increases, SDRM performance becomes closer to that of DTM with a three-fold smaller convective thermal resistance.

DRM for server workloads

Applying the SDRM technique to server workloads is straightforward, just as for the context-switched multiprogram workload. However, our simulation results revealed that SDRM is not efficient for server workloads in terms of spending the lifetime banking to maximize performance (see Lu et al. for a detailed analysis[7]). Thus, we introduce profile-based DRM (PDRM), a natural extension of SDRM, with awareness of the optimal-balance consumption in the hot phase.

Profile-based DRM

Because the pertinent thermal characteristics in server workloads are on a very large time scale (see the "Thermal behaviors in server workloads" sidebar), simulating such workloads on a cycle-accurate simulator is difficult. Instead, we constructed a hybrid workload similar to the multiprogram workload but with a much longer context-switching interval. This synthetic workload consists of a cool phase and a hot phase, running SPEC2000 benchmarks applu and gcc. Figure 4 shows the synthetic workload's temperature profile.

Although the total simulated time is short (about one second) compared with the time of a real server workload (usually tens of hours), Figure 4 shows that each phase's time interval is long enough to reach the individual program's steady-state operating temperature. The temperature variations within each program mimic the workload variations in both the cool and hot phases of a real server workload. Therefore, we can interpret the time units in Figure 4 as scaled down from a much longer time interval (several hours). One disadvantage of our synthetic workload is that it doesn't model power peaks caused by individual requests in the cool phase. However, those intermittent power peaks' effect on reli-

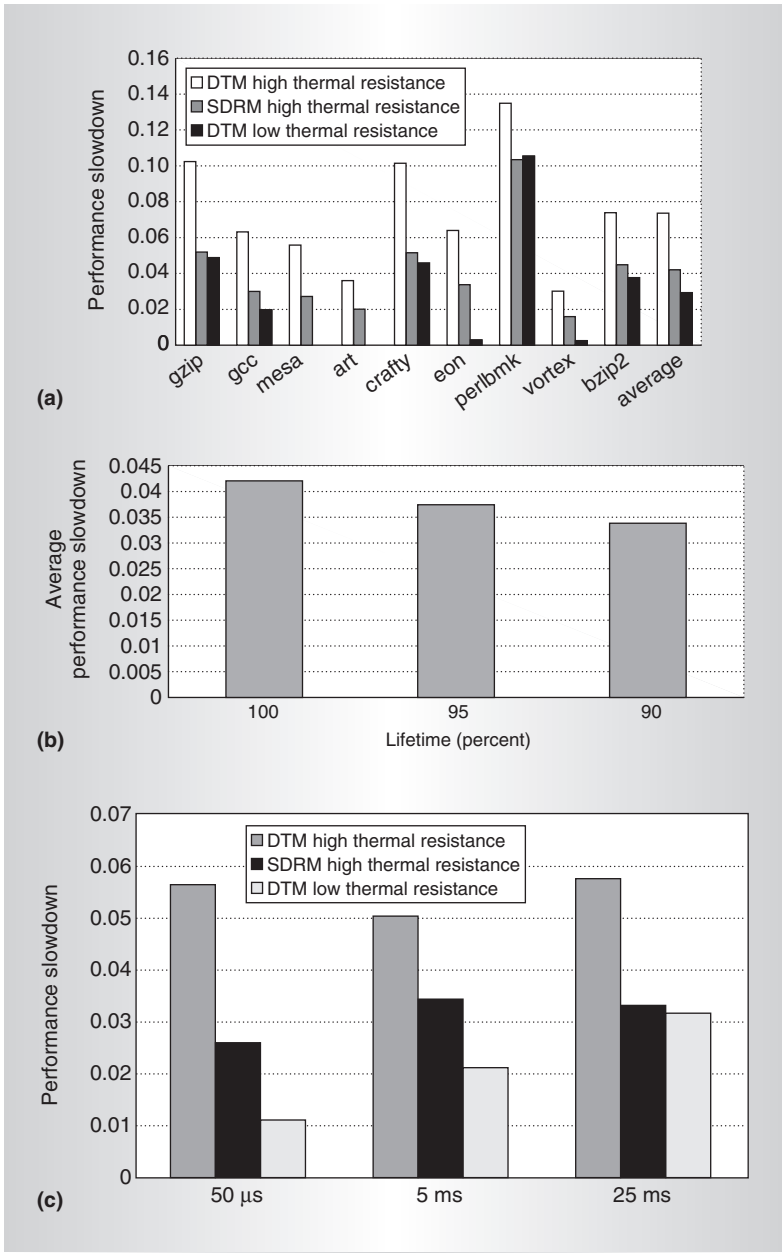


Figure 3. Performance comparison of DTM and SDRM on single-program workloads. (SDRM results are with a high-convection thermal-resistance configuration; DTM results are with two different thermal configurations.) (a) We also plot SDRM performance at different targeted lifetimes (b) and the performance comparison of DTM and SDRM on a multiprogram workload with different context-switching intervals (50 µs, 5 ms and 25 ms) (c).

Thermal behaviors in server workloads

In general-purpose computing, workload temperature variations occur largely because of inherent phased behaviors (phased activities or context switches). These variations usually occur in a very small-scale time interval, comparable to the thermal constant of the chip's thermal package. Server workloads, in contrast, depend on user requests, which vary over a much larger time scale, some tens of hours.[1]

Server workloads have several distinct characteristics. First, the workload distribution usually consists of distinct cool phases (lower request rate) and hot phases (higher request rate). For example, in a workload trace from a Web server for the 1998 Winter Olympic Games, the request rate increases from around 50 per second in the cool phase to more than 150 per second in the hot phase, a threefold difference.[1] Second, because of variations in workload and associated processor utilization, the processor's power consumption varies greatly (as much as a twofold difference in the Olympic Games servers[1]), which implies a large temperature variation. Third, each phase sustains itself for a very long time interval. Thus, each phase reaches its steady-state temperature and stays at that temperature for most of the phase interval. This is quite different from general-purpose computing, in which the interval for each thermal phase is very short, and the current phase seldom reaches the steady-state temperature before the next phase arrives. These distinct thermal characteristics make our lifetime-banking-based reliability management promising for server workloads.

In the hot phase, conventional thermal-threshold-based DTM clamps the maximum temperature to a predefined threshold by slowing the processor, possibly exacerbating the situation. In contrast, banking-based runtime management can exploit the long cool phase's banking effects, and delay or reduce the performance loss caused by engaging a cooling mechanism. From an average user's point of view, the server's quality of service depends largely on its performance in the hot phase, when most requests are made. The studies in this article therefore evaluate performance for hot-phase server workloads.

Reference

1. P. Bohrer et al., "The Case for Power Management in Web Servers," *Power Aware Computing*, R. Graybill and R. Melhem, eds., Kluwer Academic, 2002, Chap. 1, pp. 7-15.

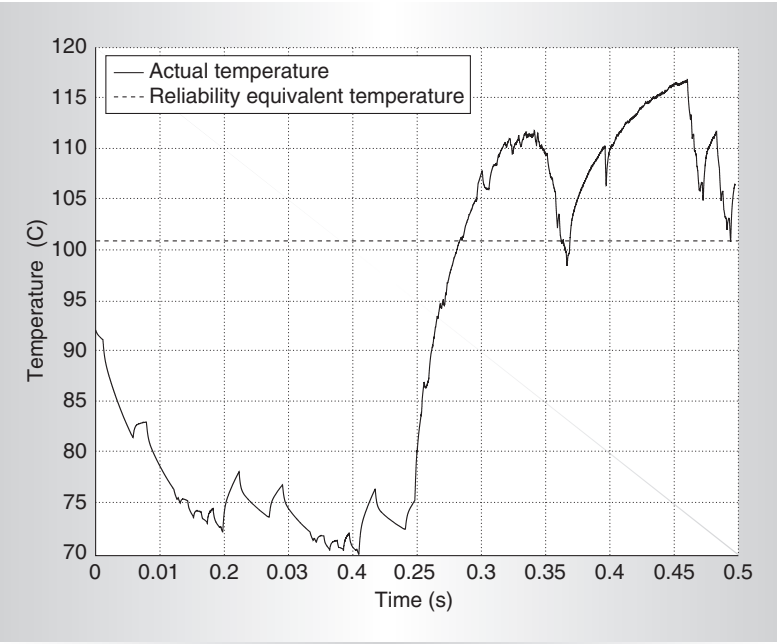


Figure 4. A synthetic workload constructed to mimic the thermal behavior of server workloads.

ability banking is insignificant because of the thermal package's filtering effect.

When the server is running in the cool phase, PDRM works the same way as SDRM with the banked lifetime balance. When the server enters the hot phase, PDRM calculates a new nominal lifetime consumption rate based on the lifetime balance already deposited and the estimated duration of the hot phase (obtained through profiling). Then, PDRM acts the same as SDRM, with the new nominal consumption rate.

In some cases, we might not be able to obtain accurate workload profiles, but the profile inaccuracy affects only performance optimality. It doesn't violate the lifetime budget because our technique always tracks the actual reliability consumption rate. Finally, our reliability banking scheme is effective irrespective of the workload's cool-hot phase order because lifetime banked in the cool phase is always ready for withdrawal in a future hot phase.

Simulation results for synthetic workloads

We simulated the synthetic workload in Figure 4, using three different cool-phase duty cycles. A *duty cycle* is the portion of time the cool phase occupies in the entire workload length. We compared the DTM, SDRM, and PDRM performance in the hot phase; Figure 5 presents the results. Both DRM techniques outperformed DTM, and PDRM performed the best. When the cool phase occupies 60 percent of the total time, PDRM can reduce the performance penalty from 16 percent (for DTM) to only 6 percent (or equivalently, PDRM increases the

hot phase's execution speed by about 9.5 percent over DTM). Interestingly, when the cool phase occupies 75 percent of total time, neither DRM technique incurs a performance slowdown because the reliability-equivalent temperature for that workload is less than the reliability-nominal temperature. In other words, the lifetime balance banked in the cool phase is enough to support full-speed execution in the hot phase. In contrast, DTM clamps the hot-phase temperature to the reliability-nominal temperature, resulting in about a 13 percent performance penalty in the hot phase.

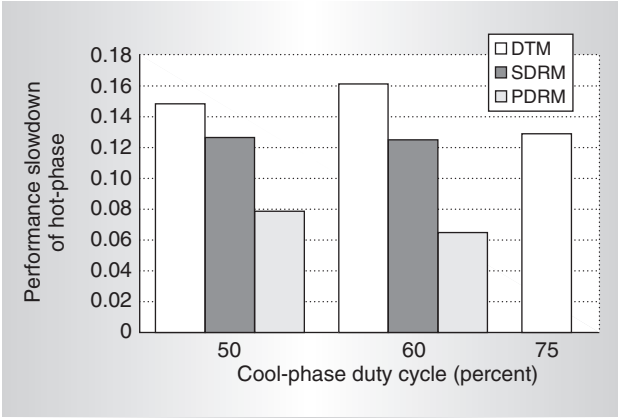


Figure 5. Performance comparison of runtime management techniques on the synthetic workload with different cool-phase duty cycles.

Analytical model

To fully understand PDRM's potential benefits on server workloads with large thermal variations, we devised a first-order analytical model. As Figure 6 shows, this model approximates server workloads using square waveforms. The dotted line represents the DTM temperature (which also implies performance) profile. In the cool phase, the PDRM temperature profile overlaps that of DTM. We assume that PDRM allows operation at a clock frequency higher than that clamped by the reliability temperature (a thermal threshold specified at design time, dashed line in Figure 6) in the hot phase, as represented by the solid line in the figure.

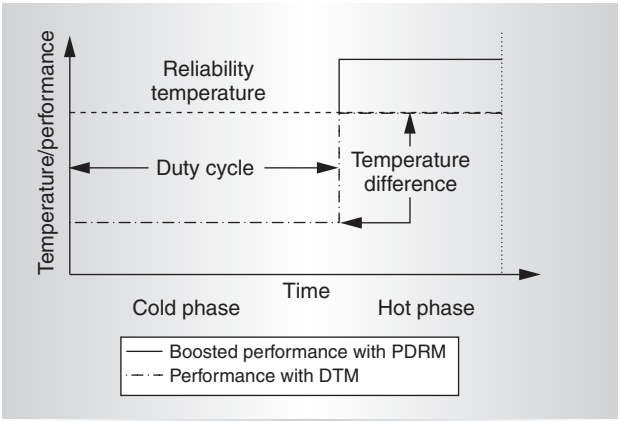


Figure 6. Modeling thermal behaviors of server workloads using square waveforms.

Figure 6 suggests that two factors affect PDRM's potential performance boost:

- the difference between the steady-state temperatures in the hot and cool phases and
- the cool-phase duty cycle.

To maintain the a specified lifetime budget throughout a workload with PDRM, the lifetime consumption rate profile must satisfy the following equation:

$$[r_n(T_n) - r_1(T_n - \Delta T)]\alpha = [r_2(f_2,\ T(f_2)) - r_n(T_n)](1 - \alpha)$$

where r_n is the nominal reliability consumption rate at the specified reliability temperature T_n; r_1 is the actual consumption rate in the cool phase; r_2 is the allowed consumption rate in the hot phase with clock frequency f_2 and temperature $T(f_2)$; and α is the cool-phase duty cycle. The left side of this equation represents the reliability balance banked during the cool phase, and the right side represents the banking deposits to be consumed in the hot phase.

With this analytical model, we can calculate PDRM's performance speedup (f_2/f_n in the hot phase) as a function of ΔT and the cool-phase duty cycle (Lu et al. present detailed calculations[7]). Figure 7 shows the results, which indicate that performance speedup is highly dependent on the cool-phase duty cycle. With a fixed cool-phase duty cycle, the increased temperature difference increases speedup. However, after some value (such as 20°C), the temperature difference has a minor effect on speedup because of the reliability consumption rate's exponential dependence on temperature. Figure 7 suggests that the sweet spot for PDRM performance speedup lies at a cool-

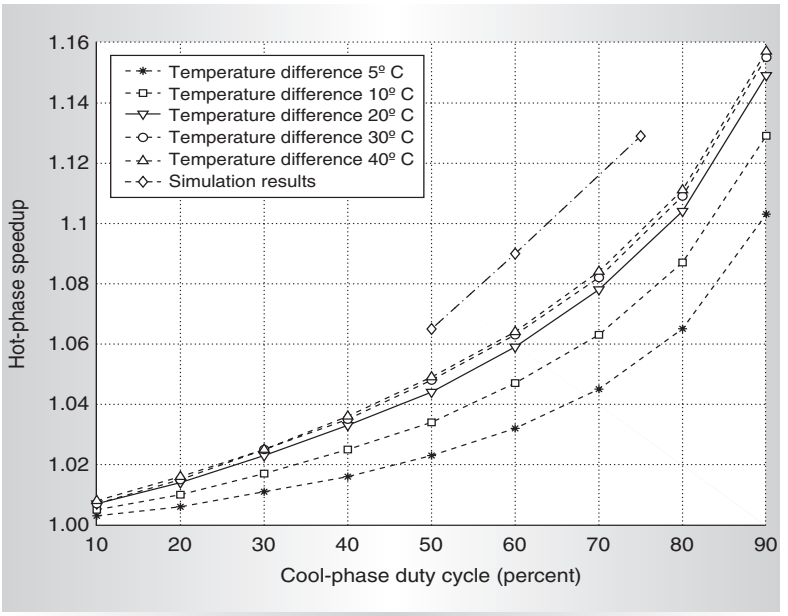


Figure 7. Performance speedup from lifetime banking with different workload characteristics.

phase duty cycle of more than 50 percent and a temperature difference around 20°C, where we can expect a performance speedup of more than 5 percent. Figure 7 also re-plots the simulated speedup of PDRM over DTM on our synthetic workload, which show a trend similar to that predicted by the simple analytical model, although we did not calibrate the analytical model against specific simulation data.

Incorporating other failure effects

Recent research on material and device reliability has shown that temperature- and voltage-dependent dynamic processes also govern other failure mechanisms, such as negative-bias temperature instability and gate oxide breakdown.[8,9] Therefore, we could derive consumption-rate-based dynamic reliability models for these failure mechanisms just as for electromigration. A simple yet conservative way to incorporate multiple failure mechanisms in the reliability-banking framework is to apply lifetime banking for each individual failure mechanism and obtain each mechanism's allowable operating point with the techniques presented in this article. We would then choose the safest one of the allowable operating points (that is, the lowest operating frequency) for circuit operation. Thus, no reliability budget violation will occur for individual failure mechanisms; however, performance gain will be minimal. A more complicated approach could trade off reliability budgets among different failure mechanisms without compromising system reliability.

The banking techniques presented here focus on the worst-case component (the hottest interconnect metal in the chip). Because our techniques guarantee that the worst-case component will satisfy its reliability constraint, we can safely claim that they don't violate system reliability. Nevertheless, this approach is conservative because temperature distribution and current density are uneven across the chip, and we can model the chip as a system consisting of serial and parallel components. A more sophisticated approach would trade off reliability among different components without violating system reliability. Such an approach would require a complex reliability model such as the one presented by Srinivasan et al.[4]

Our future work will include incorporating thermal-related failure mechanisms such as gate oxide and dielectric breakdown into the DRM framework and implementing our DRM techniques in real testbed systems. MICRO

Acknowledgments

This work is supported in part by NSF grants CCR-0133634 (Career), CCR-0306404, CCF-0105626, CCF-0429765, and CNS-0410526; Army Research Office grant W911NF-04-1-0288; a University of Virginia FEST Award; the Woodrow W. Everett Jr., SCEEE Development Fund in cooperation with the Southeastern Association of Electrical Engineering Department Heads; two research grants from Intel MRL; and an IBM Faculty Partnership Award. We thank Karthik Sankaranarayanan for his help with power trace extraction, and the reviewers 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-7), IEEE Press, 2001, pp. 171-182.
2. K. Skadron et al., "Temperature-Aware Computer Systems: Opportunities and Challenges," *IEEE Micro*, vol. 23, no. 6,

Nov.-Dec. 2003, pp. 52-61.
3. J. Srinivasan et al., "The Case for Lifetime Reliability-Aware Microprocessors," *Proc. 31st Ann. Int'l Symp. Computer Architecture* (ISCA 04), IEEE CS Press, 2004, pp. 276-287.
4. J. Srinivasan et al., "Exploiting Structural Duplication for Lifetime Reliability Enhancement," *Proc. 32nd Int'l Symp. Computer Architecture* (ISCA 05), IEEE CS Press, 2005, pp. 520-531.
5. K. Banerjee and A. Mehrotra, "Global (Interconnect) Warning," *IEEE Circuits and Devices,* vol. 17, no. 5, Sept. 2001, p. 16.
6. D. Burger and T.M. Austin, "The SimpleScalar Tool Set, Version 2.0," *Computer Architecture News*, vol. 25, no. 3, June 1997, pp. 13-25.
7. Z. Lu et al., *Temperature-Aware Modeling and Banking of IC Lifetime*, tech. report CS-2005-10, Dept. of Computer Science, Univ. of Virginia, July 2005.
8. E. Wu et al., "Interplay of Voltage and Temperature Acceleration of Oxide Breakdown for Ultra-Thin Gate Oxides," *Solid-State Electronics*, vol. 46, no. 11, Nov. 2002, pp. 1787-1798.
9. S. Zafar et al., "A Model for Negative Bias Temperature Instability (NBTI) in Oxide and High-k pFETs," *Proc. 2004 Symp. VLSI Technology and Circuits*, IEEE Press, June 2004, pp. 208-209.

Zhijian Lu is a PhD candidate in the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia. His research interests include low-power, high-performance computing systems and system-level design automation. Lu has a BE in automation engineering from Tsinghua University, China. He is a student member of the IEEE and the ACM.

John Lach is an assistant professor in the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia. His research interests include dynamically adaptable and real-time embedded systems, fault- and defect-tolerant hardware design, and CAD techniques for VLSI. Lach has a BS from Stanford University and an MS and a PhD, both in electrical engineering, from UCLA.

Mircea R. Stan is an associate professor in the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia. His research interests include high-performance, low-power VLSI; temperature-aware circuits and architecture; embedded systems; and nanoelectronics. Stan has a diploma in electronics and communications from Politechnica University in Bucharest, Romania; and MS and PhD degrees in electrical and computer engineering from the University of Massachusetts at Amherst. He is a senior member of the IEEE, and a member of the ACM, Usenix, Eta Kappa Nu, Phi Kappa Phi, and Sigma Xi.

Kevin Skadron is an associate professor in the Department of Computer Science at the University of Virginia. His research interests include how multi-threaded and multi-core organizations in the context of power, temperature, and reliability constraints are changing processor architecture. Skadron has a PhD in computer science from Princeton University and a BS in computer science from Rice University. He is a member of Eta Kappa Nu, Omicron Delta Epsilon, ACM, and a senior member of IEEE and the IEEE Computer Society and Circuits and Systems Society.

Direct questions and comments about this article to Zhijian Lu, Dept. of Electrical and Computer Engineering, University of Virginia, 351 McCormick Road, Charlottesville, VA 22904; zl4j@virginia.edu.