Temperature-Aware Modeling and Banking of IC Lifetime Reliability

UNIV. OF VIRGINIA DEPT. OF COMPUTER SCIENCE TECH. REPORT CS-2005-10
July 2005

Zhijian Lu, John Lach, Mircea Stan, Kevin Skadron‡
Dept. of Electrical and Computer Engineering, ‡Dept. of Computer Science
University of Virginia
Charlottesville, VA 22904
{zl4j, jlach, mircea}@virginia.edu, skadron@cs.virginia.edu

Abstract

Most existing integrated circuit (IC) reliability models assume a uniform, typically worst-case, operating temperature, but temporal and spatial temperature variations affect expected device lifetime. As a result, design decisions and dynamic thermal management (DTM) techniques using worst-case models are pessimistic and result in excessive design margins and unnecessary runtime engagement of cooling mechanisms (and associated performance penalties). By leveraging a reliability model that accounts for temperature gradients (dramatically improving interconnect lifetime prediction accuracy) and modeling expected lifetime as a resource that is consumed over time at a temperature- and voltage-dependent rate, substantial design margin can be reclaimed and runtime penalties avoided while meeting expected lifetime requirements. In this paper, we evaluate the potential benefits and implementations of this technique by tracking the expected lifetime of a system under different workloads while accounting for the impact of dynamic voltage and temperature variations. Simulation results show that our dynamic reliability management (DRM) techniques provide a 40% performance penalty reduction over that incurred by pessimistic DTM in general-purpose computing and a 10% increase in quality of service (QoS) for servers, all while preserving the expected IC lifetime reliability.

I. Introduction

The advance of technology scaling (and the resulting increases in power density) has made thermal-related reliability one of the major concerns in modern IC design. For example, in the deep sub-micron (DSM) region, electromigration, which is temperature dependent, is widely regarded as one of the dominant failure mechanisms. Designers must therefore rely on temperature-dependent reliability models to derive the expected lifetime of their circuits, increasing design margin (e.g., wire width) as necessary to meet lifetime requirements. Traditionally, a worst-case temperature is used to evaluate the reliability of the system, often resulting in excessive design margins.

Many post-design solutions are applied to address thermal reliability issues as well, and they can be generally divided into two categories: 1. passive cooling mechanisms, and 2. active cooling mechanisms. In the first category, designers pay an extra price for more efficient cooling packages, for example, with smaller convection thermal resistance or thinner interface spreading material. In the second category, people sacrifice a certain amount of performance to maintain reliability by reducing circuit speed (resulting in temperature reduction) whenever necessary. Recently developed dynamic thermal management (DTM) techniques [15], [5], [16] belong to this category. However, these techniques rely on worst-case assumptions, typically using a fixed temperature threshold to engage active cooling mechanisms, such as frequency/voltage scaling and throttling at the expense of degraded performance.

Under such pessimistic assumptions, DTM cooling mechanisms may often be engaged (and performance penalties incurred) unnecessarily. As a matter of fact, many programs/workloads exhibit temperature fluctuations during their executions due to inherently phased behaviors. In the paper, we show that the effect of hot (high temperature) phases on reliability can be compensated by that of cool (low temperature) phases [12]. Existing DTM techniques do not consider the effects of temperature fluctuations on lifetime and may unnecessarily impose performance penalties for hot phases. The disadvantages of these techniques become more obvious in server systems such as web servers, in which hot phases usually imply an increased number of service requests. The engagement of active cooling mechanisms then exacerbate the QoS provided by the server.

In our previous work [11], we developed a reliability model for IC interconnects under dynamic thermal and electrical current stresses. In this paper, using electromigration as the targeted failure mechanism, we extend this model and propose a dynamic reliability management (DRM) technique to dynamically track the "consumption" of chip lifetime during operation. In general, when temperature increases, lifetime is being consumed more rapidly, and vice versa. Therefore, if temperature is below the traditional DTM engagement threshold for an extended period, it may be acceptable to let the threshold be exceeded for a time while still maintaining the required expected lifetime. In effect, lifetime is modeled as a resource that is being "banked" during periods of low temperature, allowing for future withdrawals to maintain performance during times of higher operating temperatures. Using electromigration as an example, we show the benefits of lifetime banking by avoiding unnecessary DTM engagements while meeting expected lifetime requirements. The aging process due to many other temperature enhanced failure mechanisms such as gate oxide/dielectric breakdown is similarly determined by a temperature/voltage dependent rate [20]. Therefore, the results and approach discussed in this paper can be generalized to incorporate other failure mechanisms. Recently, Srinivasan *et al.* [17], [19] proposed a chip level reliability model and showed the potential benefits by trading off reliability with performance for individual application. They assumed an oracular algorithm for runtime management in their study. In this work, we

focus on practical runtime management techniques for the worst-case on-chip component (i.e. hottest interconnect) to exploit both intra- and inter- application temperature variations. The combination of their model and our techniques is expected to bring more advantages and is open for future investigation.

High temperature limits the circuit performance directly by increasing interconnect resistance and reducing carrier mobility. However, it has been shown that ([15]) using DTM to compensate the temperature dependency of clock frequency induces very mild performance penalty. On the other hand, Banerjee *et al.* [1] showed that temperature induced reliability issue tends to limit the circuit performance in future technology generations. Therefore, in this paper, we assume that the temperature threshold is set solely for reliability specification, and circuits can operate correctly above this threshold whenever allowed by the "banking" opportunities. Although extreme high temperature may cause immediate thermal damage for IC circuits, in this paper, we study a range of temperatures only with long-term reliability impacts (i.e. temperature induced aging). We assume that those high temperatures causing immediate damages are far above the range of temperatures studied here, and a monitoring and feedback mechanism is implemented to assure that circuits are operated far below those temperatures with short-term damages.

This paper is structured as follows. Section II presents an electromigration model subject to dynamic stress. Using this model, we propose a simple lifetime-banking-based DRM method (S_DRM) in Section III. We implemented this method in a compact architecture-level thermal model, *Hotspot* [15], running the Spec2000 benchmarks, with the results shown in Section IV. In Section V, we extend our study to server workloads, where temperature variation in large granularities is commonly seen, and propose a profile-based DRM technique (P_DRM). We present the results for server workloads and describe an analytical model to analyze P_DRM in Section VI. In Section VII, we discuss how other failure mechanisms can be possibly incorporated into the reliability banking framework. Finally, we conclude the paper in Section VIII.

II. Dynamic Electromigration Model

In this section, we briefly describe a dynamic electromigration (EM) model and explain how expected IC lifetime can be modeled as a resource that is consumed over time. The key to the model is to update the projected lifetime (or reliability consumption rate) according to the actual dynamic temperature information observed during program execution. Applying this model to DTM, instead of using a fixed temperature threshold, the cooling mechanisms are engaged only when the projected lifetime falls below the required lifetime.

Black's equation [3] is widely used to predict mean time to failure (MTF) due to electromigration.

$$T_f = \frac{A(kT)}{j^n} exp\left(\frac{Q}{kT}\right) \tag{1}$$

where T_f is the time to failure, A is a constant based on the interconnect geometry and material, j is the current density, Q is the activation energy (e.g., $1.0eV$ for copper interconnect), and kT is the thermal energy. The current exponent, n, has different values according to the actual failure mechanism. It is assumed that $n = 2$ for void nucleation limited failure and $n = 1$ for void growth limited failure [13]. However, Black's equation is suitable only for interconnects subject to constant temperature and current density.

We derived a model to predict interconnect lifetime due to electromigration under simultaneous dynamic thermal and current stresses [11]. In this dynamic model, Black's equation is still valid, but one should use reliability-equivalent temperature $T_{equivalent}$ and current density $j_{equivalent}$ as defined in the following:

$$j_{equivalent} = \frac{E\left[j(t)\left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right)\right]}{E\left[\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right]} \tag{2}$$

$$\frac{exp\left(\frac{-Q}{kT_{equivalent}}\right)}{kT_{equivalent}} = E\left[\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right] \tag{3}$$

where $E[\,]$ is the expected-value function, $j(t)$ and $T(t)$ are time-dependent current density and temperature functions, respectively. Substituting into Black's equation (1) the above two expressions for reliability-equivalent current density/temperature and using $n = 1$ for dual-damascene copper interconnect (widely used in modern chip manufacturing) [7], the MTF under time-varying current density and temperature stresses can be derived as:

$$\begin{aligned} T_f &= \frac{A(kT_{equivalent})}{j_{equivalent}} exp\left(\frac{Q}{kT_{equivalent}}\right) \\ &= \frac{A}{E\left[j(t)\left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)}\right)\right]} \end{aligned}$$

Or equivalently, by eliminating the expected-value function, One can express the MTF in an integral form:

$$\int_0^{T_f} j(t) \left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)} \right) dt = D \tag{4}$$

where D is a constant determined by the structure of the interconnect.

Equation (4) models interconnect time to failure (i.e., interconnect lifetime) as a resource consumed by the system over time with a temperature/current dependent rate. Function $r(t) = \left[j(t) \left(\frac{exp(\frac{-Q}{kT(t)})}{kT(t)} \right) \right]$ can be regarded as the consumption rate. In DSM copper technology, void growth failure (e.g. at vias) is the major EM induced failure mechanism [7], and $r(t)$ represents the void growth rate in this case. Due to the temporal behaviors of system workloads, interconnect current density and temperature are time-dependent. Equation (4) provides a model to capture the effect of transient behaviors on system lifetime.

In our chip-level reliability model, for simplicity, we use the maximum temperature measured across the chip to calculate the consumption rate. However, the variability of current density across the chip makes it much harder to track in real-time. Thus, we use the worst-case current density specified at design time in our calculations.

Many commercial processor products (i.e., [9], [10]) use dynamic voltage/frequency scaling (DVS) as an effective technique for active cooling. In this paper, we adopt DVS as the major guarding mechanism for preventing lifetime reliability violation. When DVS is applied, the worst-case current density in the IC interconnects should be scaled according to the voltage/frequency setting used. The relationship between current density, supply voltage and clock frequency can be modeled by transferred charges per clock cycle [2]:

$$j \propto \frac{CV}{T} = CVf$$

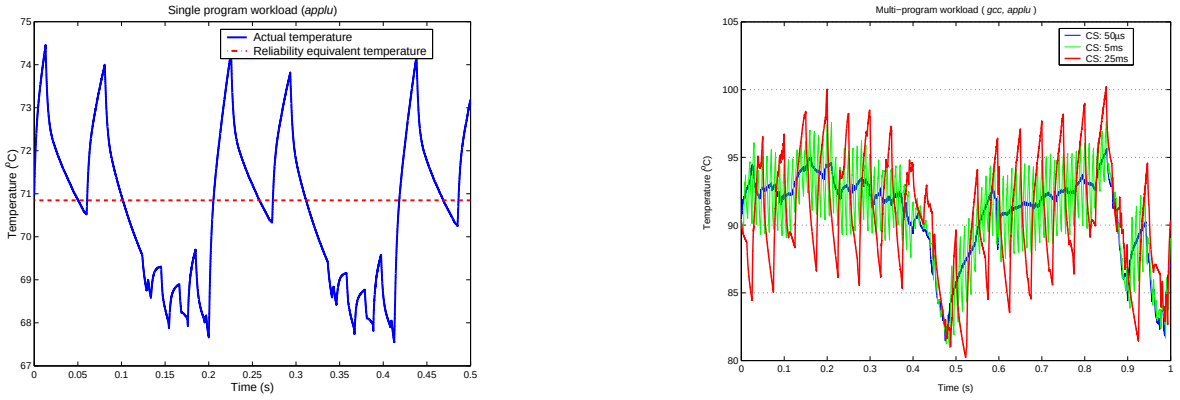
where C is the effective capacitance. Therefore, when the chip is switched to a new voltage/frequency setting, the corresponding worst-case current density is scaled by the product of the new voltage and frequency, and we can track the current density dynamically.

III. Dynamic Reliability Management Based on Lifetime Banking

Recently, many DTM techniques [15], [5], [16] have been proposed to ensure that a chip will never operate above some temperature threshold. However, these techniques do not explicitly study the effects of transient behaviors on system reliability, and instead implement a temperature upper-bound at the expense of degraded performance. By modeling lifetime as a resource to be consumed over time, we can manipulate chip lifetime directly at runtime. In this section, we present a simple dynamic reliability management (S_DRM) scheme built on conventional DTM techniques.

A. Lifetime banking opportunities

Due to activity variations, the power consumptions of on-chip components (i.e. caches, FP/INT units, branch predictor, etc.) are not constant. Therefore, there exists not only chip-wise spatial temperature gradients but also temporal temperature gradients for each component. In this paper, we focus on temporal temperature gradients. Though a more accurate reliability model should incorporate all on-chip components, in this study, for simplicity, we use the temperature of the hottest unit across the chip.



(a) (b)

Fig. 1. Temporal temperature variation. (a) Single program workload. (b) Two-program workload with context switching

Figure 1 depicts the temperature profiles for two different workloads that are commonly seen in general purpose computing. Figure 1(a) represents a single program workload and Figure 1(b) represents a multi-program workload with context switching. In the single program workload, temperature changes over time due to the phased behavior in the executed program. In the multi-program workload, besides the execution variations within each program, inter-program thermal differences also affect the overall thermal behavior of the workload. For example, in Figure 1(b), the workload is composed of one cold program (*applu*) and one hot program (*gcc*). Thus the temperature fluctuation in Figure 1(b) is quite different with various context switching intervals. Though there are different thermal behaviors for different workloads, one can still find some common characteristics as compared with server workloads, which we will discuss in Section V. In Figure 1, temperature variations occur in a manner with small granularity in both magnitude and time interval. More formally, the temperature profile can be decomposed into a constant temperature component (steady state temperature) and a high frequency component.

Calculations [11] reveal that the constant temperature component in the temperature profile is approximately equal to the reliability equivalent temperature (i.e., the lifetime at that constant temperature is equal to the lifetime projection under the temperature profile), as shown in Figure 1(a). It is the high frequency component that provides opportunities for lifetime banking. When the actual temperature is under the reliability equivalent temperature, the lifetime is consumed with a slower speed, which allows subsequent execution above the reliability equivalent temperature.

B. Reliability-aware runtime management

When a chip is designed, usually an expected lifetime (e.g., 10 years) is specified under some operating conditions (e.g., temperature, current density, etc.). We use $r_{nominal}$ to denote the lifetime consumption rate under the nominal conditions (e.g. reliability constrained temperature threshold). During runtime, we monitor the actual operating conditions regularly, calculate the actual lifetime consumption rate $r(t)$ at that time instance, and compare the actual rate with the nominal rate $r_{nominal}$ 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 slower than the nominal rate. Thus, the chip's lifetime deposit is increased. When $r(t) > r_{nominal}$, the chip is consuming its lifetime faster than the nominal, and the lifetime banking deposit will be reduced. According to Equation (4), as long as the lifetime deposit is positive, the expected lifetime will not be shorter than that under the nominal consumption rate $r_{nominal}$. Figure 2 illustrates this S_DRM technique. For example, in the interval [$t0$, $t1$], the reliability of the chip is banked, while in [$t1$, $t2$], the banking deposit is consumed. At time instance $t2$, the banking deposit becomes less than some threshold, and a cooling mechanism has to be engaged to quickly pull down the lifetime consumption rate to the nominal rate, just as is done in conventional DTM techniques. In other words, our S_DRM technique adopts DTM as a bottom-line guarding mechanism.

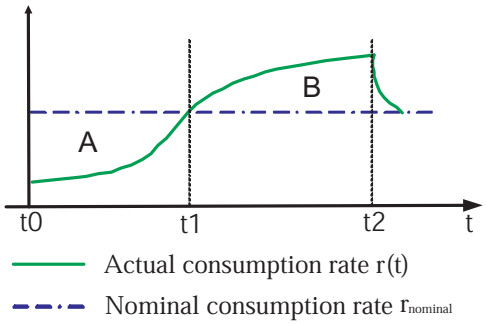


Fig. 2. Simple dynamic reliability management (S_DRM).

Therefore, the difference between conventional DTM and our S_DRM lies in the case where the chip's instantaneous consumption rate is larger than its nominal rate. In DTM, the lifetime consumption rate is never allowed to be larger than the nominal. In S_DRM, before we engage thermal management mechanisms we first check to see if 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 larger than the nominal rate. Otherwise, we apply some DTM mechanism to lower the consumption rate, thus preventing a negative lifetime balance. In this study, we use dynamic voltage/frequency scaling as the major DTM mechanism. Since S_DRM only needs to monitor the actual lifetime consumption rate and to update the lifetime banking deposit, the computation overhead is negligible compared to that of DTM.

IV. Experiments and Analysis for General-Purpose Computing Workloads

In this section, we present our simulation results for both single- and multi-program workloads using the S_DRM technique explained in the previous section. We compare these results with those obtained using conventional thermal threshold-based DTM techniques.

A. Experimental set-up

We run a set of programs from the Spec2000 benchmark suite on a processor simulator (SimpleScalar [6]) with the characteristics similar to a 0.13μm Alpha 21364. We simulate each program for a length of 5 billion instructions, and obtain both dynamic and static (leakage) power traces, which are fed as inputs to a chip-level compact thermal model *Hotspot* [15] for trace-driven simulation. In our trace-driven simulations, we include the idle penalty due to frequency/voltage switching, which is about 10us in many real systems [15]. Furthermore, since leakage power is strongly dependent on temperature, we scale the leakage power trace input dynamically according to the actual temperature obtained during runtime, using a voltage/temperature-aware leakage model [22]. Since the *Hotspot* model is highly parameterized, one can easily run experiments on a simulated processor with different thermal package settings. In order to obtain meaningful results, one should carefully choose the initial temperature setting for the *Hotspot* model. For each new thermal package setting, we obtain its initial temperatures by repeating the trace-driven simulations until the steady temperatures of the chip are converged, as suggested in [15].

We implement both DTM and S_DRM in the *Hotspot* model and set 110°C as the temperature threshold for both runtime management techniques. Both schemes use a feedback controlled dynamic voltage/frequency scaling mechanism to guard the program execution. For example, in DTM, when the actual temperature is above a certain temperature threshold, a controller is used to scale down the frequency/voltage, ensuring the program will never run at a temperature higher than 110°C. Our S_DRM scheme uses 110°C as the nominal temperature for the lifetime consumption rate. If the program never runs at a temperature less than that of the nominal (i.e., without banking opportunity), our S_DRM scheme will perform the same as thermal threshold-based DTM as the DTM policy is always engaged. On the other hand, if the program never exceeds the nominal temperature with full CPU speed, neither mechanism is engaged. Finally, we record the simulated execution times for fixed length power traces as the system performances under the two runtime management techniques, and use "performance slow-down" , defined as

$$\frac{\begin{array}{c}(\text{simulated time w/ runtime management}-\\ \text{simulated time w/o runtime management})\end{array}}{\text{simulated time w/o runtime management}}$$

as the metric to compare both techniques.

B. Single-program workload

Figure 3 shows the performance penalty for both DTM and S_DRM with the same thermal configuration. Only those benchmarks subject to performance penalties due to runtime management are shown here. As clearly indicated in the figure, performance penalty with the S_DRM scheme is always less than that with DTM scheme, when the thermal configuration is the same. On average, the S_DRM technique reduces the performance penalty by about 40% of that due to DTM (from 7% to 4%). Also shown in the figure is the performance of DTM with a more expensive thermal package whose convection thermal resistance is only one third of the others. As one can expect, a more expensive thermal package can reduce the performance penalty. Figure 3 shows that, on average, S_DRM with a higher thermal resistance can achieve a performance very close to that of DTM with a lower thermal resistance. These results imply that, if the tolerable performance lost is fixed, the application of S_DRM allows the usage of a much cheaper thermal package than that required by the conventional DTM technique.

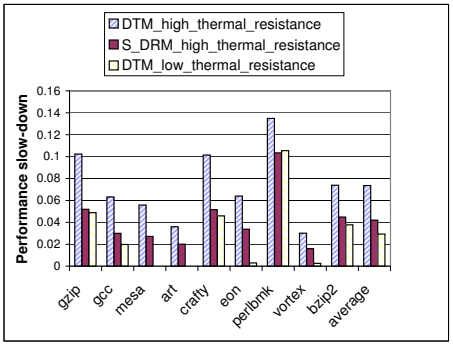


Fig. 3. Performance comparison of DTM and the proposed S_DRM. The results for S_DRM are based on high convection thermal resistance configuration. The results for DTM include two different thermal configurations.

In addition, using the S_DRM technique, one can explicitly trade-off reliability with performance by targeting different lifetime budgets. That is one can increase the nominal lifetime consumption rate when lifetime target is allowed to be reduced. Figure 4 plots the performance of S_DRM averaging over all benchmarks at different lifetime budgets, with

shorter expected lifetimes enabling faster execution. However, reducing lifetime by 10% only increases the performance by about 1%.

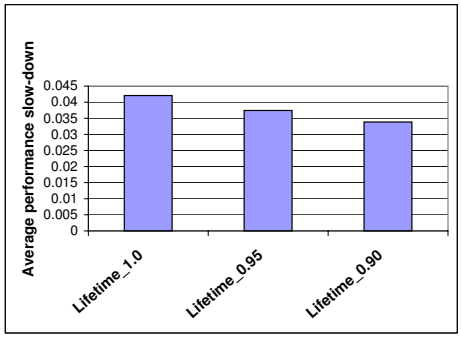


Fig. 4. S_DRM performance at different targeted lifetimes.

When compared with the conventional thermal threshold-based DTM, a distinct feature of S_DRM is its ability to "remember" the effects of previous behaviors. If the lifetime balance is high due to previous deposits, S_DRM will be more tolerant of higher operating temperatures for longer time intervals, thus reducing the performance penalties due to conventional DTM slow-down mechanisms. In summary, the advantage of S_DRM over DTM is largely dependent on the inherent variations in the temperature profile of the workload.

C. Multi-program workload

Another interesting program execution scenario is a workload of multiple programs with context-switching between them. When a hot benchmark and a cold benchmark are executed together, the average operating temperature should be between the individual benchmarks operating temperatures. For example, *gcc*'s own operating temperature is around 115^{o}C and *applu*'s is around 70^{o}C. Figure 1(b) plots the temperature profile of a hybrid workload composed of *gcc* and *applu*, with different context-switch time intervals. Note that, in our simulation, the multi-program workload is constructed using the power trace of the individual program, and the overhead of context-switching is not modeled and simulated. Since we are only interested in the relative performance of different runtime management technique, such simplification should not affect the final conclusions.

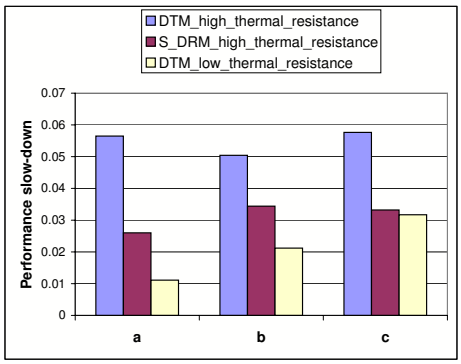


Fig. 5. Average performance comparison of DTM and DRM on a multi-program workload with different context-switch intervals ((a) 50μs, (b) 5ms, and (c) 25ms).

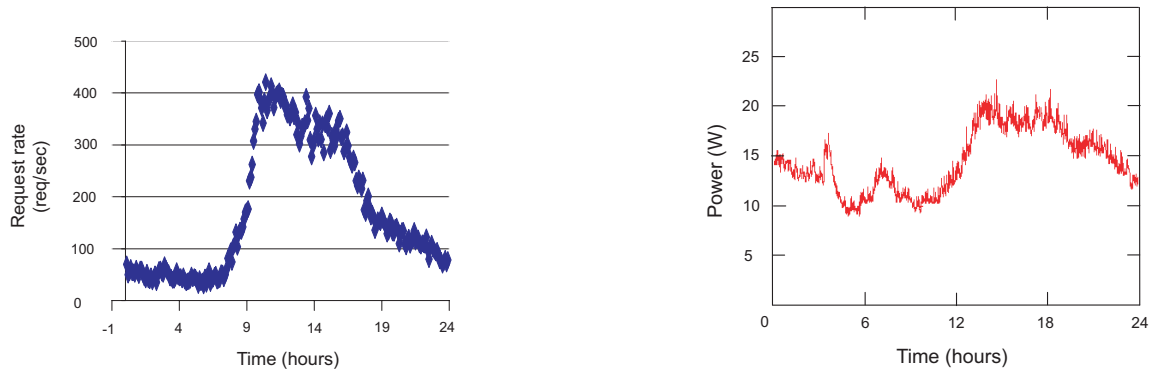
As one expects, the smaller the context-switch interval, the less temperature fluctuation, with the thermal package of the chip working as if a low-pass filter. When the context-switch interval is increased, individual benchmarks can show their hot/cold properties, and the temperature variation in the workload becomes obvious. In order to investigate how multi-program workloads affect the performance of DTM and DRM, we reduced the temperature threshold of the targeted lifetime from 110°C to 90°C. Figure 5 shows the performance penalties of DTM and S_DRM for this multi-program workload with different context-switch intervals. We observe a similar trend shown in the single-program workload. S_DRM outperforms DTM with the same thermal package configurations. As the context-switch interval increases, the performance of S_DRM becomes closer to that of DTM with a much smaller convection thermal resistance (three-fold smaller).

V. Dynamic Reliability Management for Server Workloads

In Sections III and IV, we investigated the application of DRM in workloads for general-purpose computing. In this section, we discuss some distinct characteristics of server workloads in terms of both thermal behaviors and performance requirements. We propose a profile-based dynamic reliability management (P_DRM) technique that can extract more benefits from lifetime banking for those server workloads.

A. Characteristics of server workloads

In general-purpose computing, the temperature variations of workloads are largely due to the inherent phased behaviors (i.e. phased activities or context-switches). These variations usually occur in a very small-scale time interval, which is comparable to the thermal constant of chip thermal package. Server workloads, in contrast, are dependent on user requests, which vary with a much larger time scale. Figure 6 presents some examples extracted from [8], [4]. Figure 6(a) shows user requests distribution over time for the web servers for the 1998 Winter Olympic Games, and Figure 6(b) presents a real power measurement of the processor for a single node web server processing requests similar to those in Figure 6(a).



(a) (b)

Fig. 6. Web server workload variation. (a) User requests distribution (Extracted from [8]). (b) Processor power consumption variation in a web server (Extracted from [4]).

There are several observations which can be drawn from Figure 6. First, there is clearly a cool phase and a hot phase in the workload distribution, and the cool phase occupies most of the time. The difference in computation requirements between the cool phase and the hot phase is very large. For example, the request rate increases from around 50 (req./s) in the cool phase to above 400 (req./s) in the hot phase, an eight-fold difference. Second, as a consequence of the workload and associated processor utilization variation, the power consumption of the processor varies greatly (a two-fold difference), which implies a large variation in temperature. Third, each phase sustains for a very long time interval. Thus one can expect that each phase reaches its steady-state temperature and stays at that temperature for most time of the phase interval. This is quite different from general-purpose computing, where the interval for each thermal phase is very short and the steady-state temperature is seldom reached before the next phase arrives. We believe these distinct thermal characteristics make our lifetime-banking-based reliability management more suitable for server workloads. Note that the power number shown in Figure 6 represents the averaged power consumption over time intervals and each individual request will still cause power consumption peak momentarily. However, the negative effects on lifetime banking by these power peaks in the cool phase will be diluted because of two reasons: 1. The corresponding temperature peaks are small due to the filtering effect of the large thermal constant of the processor package. 2. Our analysis [11] reveals that reliability banking is mostly determined by the average temperature.

As one can see from the workload distribution in Figure 6, the performance bottle-neck of a server exists in the hot phase, which is associated with high temperature. Conventional thermal threshold-based DTM clamps the maximum temperature to a predefined threshold by slowing down the processor, thus possibly exacerbating the situation. In contrast, banking-based runtime management can exploit the banking effects of the long cool phase and delay or reduce the performance loss due to engagement of an active cooling mechanism. From an average user's point of view, the QoS provided by the server is largely dependent on its performance in the hot phase, as most requests are made during that time. Therefore, in the following study, *we use the performance of the hot phase as our performance metric for comparison.*

B. Dynamic reliability management for server workloads

In order to evaluate our runtime management technique on server workloads, we construct a hybrid workload in a way similar to that of the multi-program workload discussed in Section IV, but with a much longer context-switch interval. This synthetic workload is composed of a cool phase and a hot phase, running Spec2000 benchmarks *applu* and *gcc* respectively. Figure 7 shows the temperature profile of the synthetic workload we use to mimic the thermal behavior of server workloads. From various experiments, we find that the thermal time constants of our simulated system are in the range of tens of milliseconds. Therefore, by simulating workloads in a time scale of several seconds, we can ensure that the portion of time in the profile spent on the transient behaviors from one phase to another is minimized, just like one may see in a temperature profile for server workloads. Although the total simulated time is short (i.e., about one second) compared to a real server workload, Figure 7 indicates that the time interval for each phase is long enough to reach the steady-state operating temperature of the individual program. The temperature variations within each program also mimic the workload variations in both the cool and hot phases of a real server workload. Therefore, the time units shown in Figure 7 could be interpreted as scaled down from a much longer time interval (e.g. several hours). One disadvantage of our synthetic workload is that power peaks due to individual requests in the cool phase are not modeled. However, as discussed previously, the effect of those intermittent power peaks on reliability banking is not significant.

In our synthetic workloads, the cool phase is followed by the hot phase, and lifetime will be banked first and then withdrawn. In other workloads where the hot phase is followed by the cool phase, DTM can be applied in the hot phase if there is no previous lifetime banking, and lifetime will be banked in the following cool phase and prepared for withdrawal in the future hot phase. Thus, our lifetime banking based approach is effective in spite of the detail of workloads (i.e. the order of cool and hot phases). We define the *duty cycle* of the cool phase as the portion of time the cool phase occupies in the whole length of simulation. For example, in Figure 7, the duty cycle of the cool phase is equal to 0.5. In our experiments, we also construct workloads with different duty cycles of the cool phase (e.g., 0.6 and 0.75), and in all of these workloads, individual programs reach their own steady-state temperatures.

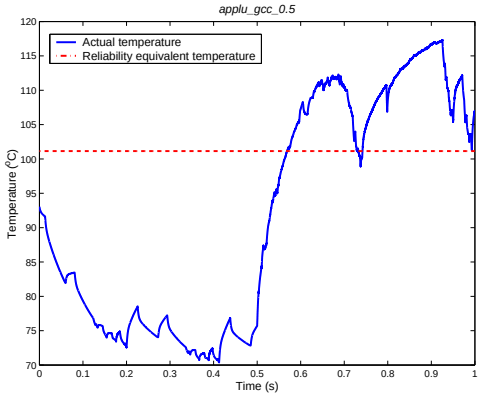


Fig. 7. A constructed workload used to mimic the thermal behavior of server workloads.

The application of the S_DRM technique to server workloads is straightforward, just like in the context-switched multi-program workload studied above. However, our simulation results reveal that S_DRM is not the best choice for server workloads. In the workloads of general-purpose computing, since each phase is very short, the lifetime balance deposited in the previous cool phases can support the subsequent over-consumption of lifetime for an interval comparable to that of the hot phase. S_DRM can minimize the impacts on the phase within these workloads. However, in the server workloads similar to that shown in Figure 6, the interval of the hot phase is much longer, and temperature rises steadily towards the hot phase steady-state temperature. At the same time, due to the exponential dependence of lifetime consumption rate on temperature, the lifetime balance is consumed more and more rapidly, despite a previous long cool phase. Figure 8 demonstrates such a process in the time interval $[0.6s, 0.68s]$. After 0.68s, the lifetime balance becomes *zero*. S_DRM performs during the rest of the hot phase just as it behaves in the single program workload. Therefore, only a small portion of the execution in the hot phase benefits from the lifetime banking by the cool phase.

Due to the above reason, one should find a more strategic way to spend the lifetime balance in order to maximize the performance in the hot phase. Since in steady state, temperature can be modeled as a function of the operating frequency, one can find the relationship between lifetime consumption rate and operating frequency. Let $f(t)$ denotes the operating frequency curve in the hot phase, and $r(f(t))$ be the corresponding lifetime consumption rate. The problem to find the maximum performance operating scheduling while satisfying the reliability constraint can be formulated as

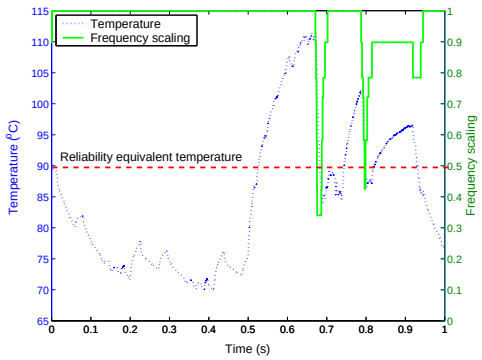


Fig. 8. S_DRM (simple dynamic reliability management) on the synthetic workload shown in Figure 7.

a constrained optimization problem as follows.

$$Max(E[f(t)]), \text{ subject to } E[r(f(t)] = R,\ t \in \text{hot phase}$$

where $E[\,]$ is the expected-value function, and R is a constant in the hot phase that is determined by the lifetime balance deposited during the cool phase as well as the nominal lifetime consumption rate. We assume that, in the hot phase, system performance is proportional to the clock speed.

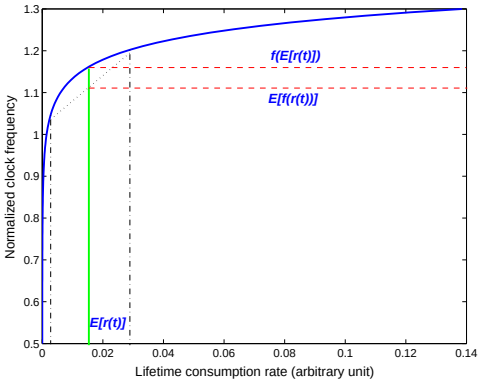


Fig. 9. Relationship between clock frequency and lifetime consumption rate.

Figure 9 plots clock frequency as a function of the lifetime consumption rate. It is obvious that the relationship between clock speed and lifetime consumption rate forms a convex curve. According to Jensen's inequality, it follows that (as shown in Figure 9) $f(E[r(t)]) \geq E[f(r(t))]$, which implies that, in order to obtain the best performance, one should operate with a constant consumption rate. In other words, one should distribute the lifetime balance evenly across the hot phase. In order to calculate the desired consumption rate in the hot phase, one has to know the duration of the hot phase. Currently we assume that this information can be obtained through profiling technique thanks to the high regularity of the workload distribution for servers.

With the optimal operating condition in mind, we introduce our (P_DRM, profile-based dynamic reliability management) technique, which is a natural extension of our S_DRM with the awareness of the optimal operating points in the hot phase. When the server is running in the cool phase, P_DRM works the same way as S_DRM with lifetime balance banked. When the server enters the hot phase, P_DRM calculates a new nominal lifetime consumption rate based on the lifetime balance and the duration of the hot phase (obtained through profiling). Then P_DRM acts just like S_DRM, with the new calculated nominal consumption rate, which can further exploit some banking opportunities due to temperature variations within the hot phase.

The profiling only provides a *prediction* that allows the CPU to jump to the best operating point during a hot phase. In some cases we might not be able to obtain accurate workload profiles. However, with our P_DRM technique, the inaccuracy of workload profiles only affects the the performance optimality, and does not result in violations to the lifetime budget. That is because our technique always tracks the actual reliability consumption rate and compares it with the nominal lifetime consumption.

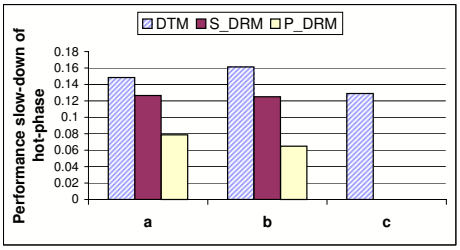


Fig. 10. Performance comparison of different runtime management techniques on the synthetic workload shown in Figure 7 with different duty cycles of the cool phase: (a) 0.5, (b) 0.6 and (c) 0.75.

VI. Experiments and Analysis for Server Workloads

A. Simulation results

We simulate the synthetic workload shown in Figure 7, which mimics the thermal behaviors of the real server workload, with different runtime management techniques. We change the program switching time so that we can test on 3 workloads with different duty cycles of the cool phase. We compare the performance slow-down in the hot phase and the results are presented in Figure 10. Both DRM techniques outperform DTM, and P_DRM performs the best. The performance of S_DRM is slightly better than that of DTM and much worse than P_DRM due to the reasons discussed in above. On the other hand, P_DRM can fully exploit the banking benefits of the cool phase. For example, when the cool phase occupies 60% of the total time (i.e. as indicated by (b) in Figure 10), P_DRM can reduce the performance penalty from 16%(DTM) to only 6% (or equivalently, the execution speed of the hot phase is increased by P_DRM by about 9.5% over DTM). Interestingly, for the case when the cool phase occupies 75% of the total time (i.e., (c) in Figure 10), no performance slow-down is incurred for both DRM techniques, because the reliability equivalent temperature for that workload is less than the reliability nominal temperature. Thus, in that case, the lifetime balance banked in the cool phase is enough to support the full speed execution in the hot phase, while DTM clamps the hot phase temperature to the reliability temperature, resulting in about a 13% performance penalty in the hot phase.

B. Analysis using an analytical model

In order to fully understand the potential benefits of p_DRM on server workloads, we present a first order analytical model, providing some insights of our proposed runtime techniques. In this model, we approximate server workloads using square waveforms as shown in Figure 11. The solid blue line represents the temperature/performance profile with DTM. The temperature profile with P_DRM in the cool phase overlaps with that of DTM. And P_DRM allows operating points above the reliability temperature in the hot phase, as presented by the dotted green line in the figure. We want to find out what is the allowable performance difference between the dotted green line and the solid blue line (i.e., the performance gain of P_DRM over DTM), subject to a fixed lifetime budget. Here we make an assumption that the processor can operate at a clock frequency higher than that clamped by the thermal threshold. There are two aspects to this. First, temperature excursions will require a reduction in frequency, thus reducing performance somewhat, but should still outperform a strict, temperature-limited form of DTM because the temperature dependence of frequency is mild [15]. Second, many ICs are actually under-clocked due to thermal limitations. In both cases, there exist possibilities that we can over-drive the processor in the hot phase to meet the QoS requirements without sacrificing reliability lifetime.

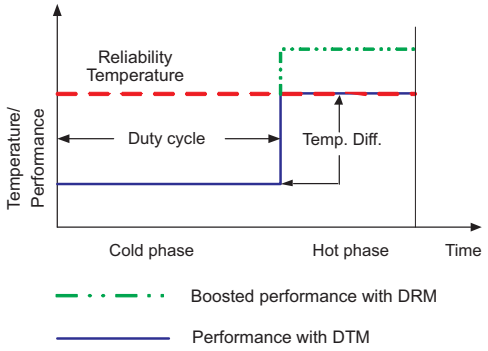


Fig. 11. Modeling thermal behaviors of server workloads using square waveforms.

As one can see from Figure 11, two factors might affect the potential performance boost by P_DRM over DTM: 1. the difference between the steady-state temperatures in both the hot phase and the cool phase, and 2. the duty cycle of the cool phase. Borrowing some data from the ITRS [14], we set the reliability temperature $T_n = 105^oC$, associated with the clock frequency $f_n = 3.0GHz$. This setting means that in the hot phase, the maximum performance achieved by DTM is to operate at $3.0GHz$. If we can assume that the dynamic power consumption of the processor is proportional to the cubic of clock frequency, the steady state temperature can be denoted by $T(f) = K_f f^3 + T_0$, where K_f is a constant and T_0 represents the ambient temperature of the thermal package. Accounting for the contribution of static power consumption to temperature, we set a higher ambient temperature $T_0 = 55^oC$, and obtain $K_f = 1.85K/GHz^3$. Let ΔT denote the temperature difference between the hot phase and the cool phase, f_2 the allowable operating clock frequency in the hot phase by P_DRM, and α the duty cycle of the cool phase. The following equation should be satisfied to retain the same lifetime budget with P_DRM:

$$[r_n(T_n) - r_1(T_n - \Delta T)]\alpha = [r_2(f_2, T(f_2)) - r_n(T_n)](1 - \alpha) \quad (5)$$

where r_n is the nominal reliability consumption rate at temperature T_n, r_1 is the consumption rate in the cool phase, and r_2 is the consumption rate in the hot phase with clock frequency f_2 and temperature $T(f_2)$. The lifetime consumption rate is discussed in Section II. The left hand side of the above equation represents the reliability balance banked during the cool phase and the right hand side represents the banking deposits to be consumed in the hot phase. Although the temperature dependence of static power is not taken into account in this model, we feel that it captures the key relationships between performance, operating temperature and reliability consumption rate, and is thus sufficient for our purposes.

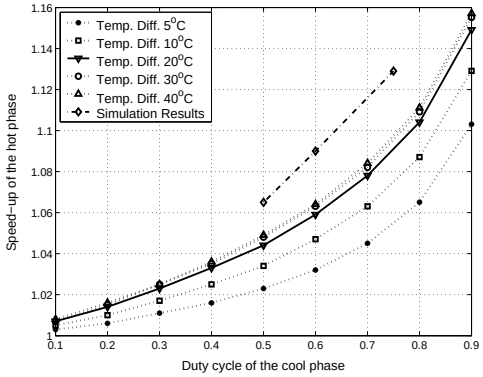


Fig. 12. Performance speed-up due to lifetime banking on different workload characteristics.

Using the above analytical model (i.e. Equation (5), we can calculate the performance speed-up by P_DRM (i.e. $\frac{f_2}{f_n}$ in the hot phase) as a function of ΔT and the duty cycle of the cool phase. The results are presented in Figure 12, which shows that the performance speed-up is highly dependent on the duty cycle of the cool phase. The increase of the speed-up vs. the duty cycle is more than linear. This is understandable, because larger cool phases produce more reliability balance, thus enabling higher execution points in the hot phase. When the duty cycle of the cool phase is fixed, the increase of temperature difference will also increase the speed-up. However, after some value (e.g. about 20^oC), the temperature difference has a minor effect on the speed-up, due to the exponential dependence of the reliability consumption rate on temperature. Because the extra reliability balance brought by further lowering the temperature in the cool phase is negligible when compared to the very high consumption rate in the hot phase. This figure suggests that the "sweet spot" for performance speed-up with P_DRM lies in the case when the duty cycle of the cool phase is more than 50% and the temperature difference is more than 20^oC, and we can expect more than 5% of performance speed-up. Fortunately, as shown before, many server workloads satisfy these requirements.

The simulation results of DTM and P_DRM from Figure 7 are re-plotted in Figure 12. The workloads for these data are similar to that shown in Figure 7, with the cool phase duty cycle equal to 0.5, 0.6 and 0.75 respectively. The reliability temperature is set to 90^oC, while the temperature of the cool phase in these workloads is about 70^oC. These simulation results show a similar trend to that predicted by our simple analytical model, though our analytical model is not calibrated against any specific simulation data. Therefore, these simulation results confirm the applicability of our analytical model. Compared with the simulation results, it seems that the analytical model underestimates the performance speed-up by P_DRM. Two major reasons might help explain the discrepancy. First, in our analytical model, we use a cubic relationship between power and operating frequency, which exaggerates the effect of clock frequency on the temperature, leading to a more conservative estimate of the performance speed-up. Second, in the simulations, we include the idle penalties for frequency/voltage transitions due to dynamic frequency/voltage scaling, while in the analytical model, we do not assume any extra performance penalty for DTM.

VII. Discussion

Recent researches on material/device reliability illustrated that other failure mechanisms, for example, Negative Bias Temperature Instability (NBTI) and gate oxide breakdown, are also governed by temperature/voltage dependent dynamic processes [21], [20]. Therefore, consumption rate-based dynamic reliability models for these failure mechanisms could be derived just as the case for electromigration. A simple and yet conservative way to incorporate multiple failure mechanisms in the reliability banking framework will be like this: lifetime banking is applied for each individual failure mechanism, and the allowable operating point for each mechanism is obtained using the techniques presented in this paper. The safest one among those allowable operating points (e.g. lowest operating frequency) is then chosen for circuit operation. In this way, no violation in reliability budget will occur for each individual failure mechanism, however, performance gain will be minimized. A more complicated approach should be able to trade off the reliability budgets among different failure mechanisms while the system reliability is not compromised.

The banking techniques presented in this paper focus on the worst-case component (i.e. the hottest interconnect metal in the chip). Since, using our techniques, we guarantee that the worst-case component will satisfy its reliability constraint, we can safely claim that the system reliability is not violated. Nevertheless this approach is conservative, since the temperature distribution is not even across the chip, and the chip can be modeled as a system constituted of serial and parallel components. A more sophisticated approach would trade off the reliability between different components while the system reliability is not violated. In order to do so, a complex reliability model such as the one in [18] has to be used.

VIII. Conclusion

Variations in operating temperature have a major impact on the expected lifetime of an IC. By taking such variations into account, we can model lifetime as a resource that is consumed over time at a temperature- and voltage-dependent rate.

In this paper, we detailed the use of the temperature variability and lifetime resource models to develop novel DRM techniques that reduce the performance penalties associated with existing DTM techniques while maintaining the required expected IC reliability lifetime. When the operating temperature is below a nominal temperature (i.e., the threshold temperature used in DTM techniques), lifetime is being consumed at a slower than nominal rate, effectively banking lifetime for future consumption. A positive lifetime balance allows the nominal temperature to be exceeded for some time (thus consuming lifetime at a faster than nominal rate) instead of automatically engaging DTM and unnecessarily suffering the associated performance penalties.

We discussed the applications of DRM in both general-purpose computing and server workloads, using interconnect electromigration as the temperature-dependent failure mechanism. For general-purpose computing, simulation results revealed that S_DRM provides performance improvements over traditional threshold-based DTM without sacrificing expected lifetime, or allows the usage of cheaper thermal package without sacrificing performance. In addition, we showed the relationship between performance and expected lifetime, revealing how one can be traded off for the other, thus providing another design optimization and runtime management dimension. For server workloads, simulations on synthetic workloads demonstrate the possibility to increase server QoS by using P_DRM when service requests are aggregated. A conservative analytical model further identifies the "sweet spots" of server workloads that benefit from our P_DRM. Future work will include incorporating other thermal related failure mechanisms such as gate oxide/dielectric breakdown into the DRM framework, and implementing our DRM techniques in practical systems.

Acknowledgments

This work is supported in part by the National Science Foundation under grant nos. CCR-0105626, CCR-0133634 and CCF-0429765, two grants from Intel MRL, and a grant from the University of Virginia Fund for Excellence in Science and Technology. The authors would also like to thank Karthik Sankaranarayanan for his help with the power trace extraction.

References

[1] K. Banerjee and A. Mehrotra. Global (interconnect) warning. *IEEE Circuits and Device Magazine*, page 16, September 2001.
[2] D. T. Blaauw, C. Oh, V. Zolotov, and A. Dasgupta. Static electromigration analysis for on-chip signal interconnects. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):39–48, January 2003.
[3] J. R. Black. Mass transport of aluminum by momentum exchange with conducting electrons. In *IEEE Int. Rel. Phys. Symp.*, pages 148–159, 1967.
[4] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, and R. Rajamony. The case for power management in web servers. In R. Graybill and R. Melhem, editors, *Power Aware Computing*. Kluwer Academic Publications, 2002.
[5] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. of the 7th International Symposium on High-Performance Computer Architecture (HPCA-7)*, January 2001.
[6] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.
[7] J. A. Davis and J. D. Meindl, editors. *Interconnect Technology and Design for Gigascale Integration*. Kluwer, 2003.
[8] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

[9] M. Fleischmann. Reducing x86 operating power through LongRun. In *IEEE 12th Hot Chips Symposium*, August 2000.
[10] D. Genossar and N. Shamir. Intel Pentium M processor power estimation, budgeting, optimization, and validation. *Intel Technology Journal*, 7(2), May 2003.
[11] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron. Interconnect lifetime prediction under dynamic stress for reliability-aware design. In *Proc. of International Conference on Computer Aided Design*, pages 327–334, November 2004.
[12] Z. Lu, J. Lach, M. Stan, and K. Skadron. Banking chip lifetime: Opportunities and implementation. In *Workshop on High Performance Computing Reliability Issues*, February 2005. in conjunction with HPCA 2005.
[13] Y.-J. Park, V. K. Andleigh, and C. V. Thompson. Simulations of stress evolution and the current density scaling of electromigration-induced failure times in pure and alloyed interconnects. *J. Appl. Phys.*, 85(7):3546–3555, April 1999.
[14] SIA. International technology roadmap for semiconductors, 2001.
[15] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankanarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. of the 30th International Symposium on Computer Architecture*, pages 2–13, June 2003.
[16] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *Proc. of the 17th Annual ACM International Conference on Supercomputing*, pages 109–120, June 2003.
[17] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proc. of the 31st International Symposium on Computer Architecture*, pages 276–287, June 2004.
[18] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. of the 32st International Symposium on Computer Architecture*, June 2005.
[19] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Lifetime reliability: Toward an architectural solution. *IEEE Micro*, 25(3), 2005.
[20] E. Wu, J. Sune, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, and D. Harmon. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. *Solid-State Electronics*, 46(11):1787–1798, November 2002.
[21] S. Zafar, B. Lee, J. Stathis, A. Callegar, and T. Ning. A model for negative bias temperature instability (nbti) in oxide and high-k pfets. In *2004 Symposium on VLSI Technology and Circuits*, June 2004.
[22] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical Report CS-2003-05, U.Va. Department of Computer Science, March 2003.