Managing Performance-Reliability Tradeoffs in Multicore Processors

William J. Song, Saibal Mukhopadhyay, *Senior Member, IEEE*, and Sudhakar Yalamanchili, *Fellow, IEEE*

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
wjhsong@gatech.edu, saibal@ece.gatech.edu, sudha@gatech.edu

Abstract— **There is a fundamental tradeoff between processor performance and lifetime reliability. High throughput operations increase power and heat dissipations that have adverse impacts on lifetime reliability. On the contrary, lifetime reliability favors low utilization to reduce stresses and avoid failures. A key challenge of understanding this tradeoff is in connecting application characteristics to device-level degradation behaviors. Using a full-system microarchitecture and physics simulation, the performance-reliability tradeoff in a multicore processor is analyzed by introducing a metric, *throughput-lifetime product* (TLP). A finding reveals that reducing the variance of degradation distribution on the multicore die leads to effectively enhancing processor lifetime with minimal impact on performance. This concept is referred to as *dynamic reliability variance management* (DRVM). We discuss three possible microarchitectural techniques that perform DRVM and improve the TLP; i) phase-aware thread migration, ii) dynamic voltage scaling, and iii) turbo-mode execution combined with DRVM. The simulation results with selected PARSEC and SPLASH-2 benchmarks show that DRVM techniques improve processor lifetime up to 15% or enhance the throughput-lifetime tradeoff by 12% without adding extra design margins or spare components on the multicore die.**

I. Introduction

Highly compact integration of transistors in a chip raises lifetime reliability concerns for future processors. Traditionally processors have been designed with large design margins and guard bands to guarantee the worst-case operations. However, in practice applications rarely operate at the limits, and advance of microarchitectural management techniques (e.g., thread scheduling, voltage and frequency scaling, etc.) enabled the processor to adapt operations and avoid extreme conditions. In physically constrained processors, adding large design margins is a costly solution and prohibits the performance growth. Therefore, microarchitectural approaches such as dynamic reliability management (DRM) [3], [7], [11], [17] have gained favor as cost-efficient methods to enhance the lifetime reliability of processors.

In this paper, we present an approach to address these challenges by bridging the gap between the physics of device operations and application behaviors. In a multicore processor, cores experience different levels of stresses depending on application characteristics, power states, thermal behaviors, etc. As a result, the processor produces uneven degradation across cores, and the cores that experience stronger stresses become subject to earlier failures. Processor-level lifetime and throughput are eventually limited by these early failing components. In addition, there exists a fundamental tradeoff between performance and lifetime reliability. High performance means more switching activities at microarchitectural units accompanied by higher power and heat dissipations that accelerate device degradation. On the other hand, enhancing lifetime reliability is biased to lower utilization to reduce stresses and chance of failures. Therefore, DRM is not merely about enhancing lifetime but must balance the tradeoff between performance and reliability. The challenges are i) characterizing how the executions of parallel applications create device-level degradation in a multicore processor, ii) understanding how device-level degradation behaviors affect processor-level lifetime reliability and performance, iii) quantifying the reliability and performance tradeoff, and iv) using these understandings to develop techniques to manage the tradeoff between processor-level lifetime reliability and performance.

The remainder of the paper is organized as follows. We first review related efforts regarding microarchitectural approaches to characterize and manage processor lifetime reliability. Our experiment using a full-system simulation framework is described including cycle-level microarchitecture timing simulator interacting with various physical models. Lifetime reliability characterization and observation are made using this framework. We introduce a metric to evaluate the tradeoff between processor performance and lifetime reliability. Finally, we present microarchitectural techniques to improve this performance-reliability tradeoff and discuss the results.

II. Related Works

It is anticipated that significant reduction of failure rates will be required to sustain lifetime reliability for future processors [17]. Adding large design margins or extra components (e.g., structural duplication or redundancy [6], [18]) on the die significantly increases development and manufacturing costs. It is better for the processor to adapt to avoid harmful operating conditions and resulting failures. Several researchers have proposed such microarchitectural adaptation techniques. Lu et al. [11], Coskun et al. [3], and Karl et al. [7] treated lifetime reliability as a resource and presented methods to save lifetime reliability during low performance or idle states and use it during high performance operations. Mercati et al. [12] studied regulating voltage and thermal histories to meet the reliability target. Feng et al. [5] showed a workload scheduling method for wear leveling based on degradation monitoring. Coskun

TABLE I. Microarchitecture-Physics Simulation Setup [15]

Models	Description
Front-end	Qsim (QEMU) functional emulation [8]
Benchmarks	Multi-threaded PARSEC & SPLASH-2 [2]
Cores	32 out-of-order cores (timing model) with 128-entry ROB, 6-issue width, 80-entry LSQ
Caches	32KB coherent L1 & 32MB shared L2 [1]
On-Chip Network	6x6 torus network
Memory System	8 MCs, 1 channel, 2 ranks, 8 banks, t_{RAS}=30, t_{CAS}=10, t_{RCD}=10, t_{RP}=10
Power Model	Enhanced McPAT [9] to support DVFS and leakage feedback, modeled at 16nm
Thermal Model	3D-ICE [16] configured to a 2D package
Reliability Model	Wear models in Table II adjusted to meet 5 years of processor-level MTTF at the baseline conditions (defined at T=65°C, V=0.8V).
Microarchitecture & Physics Interaction	KitFox framework [14] that orchestrates the interactions between power, temperature, reliability, and other runtime conditions (e.g, voltage, clock frequency) with a microarchitecture timing model

et al. [4] presented several thread scheduling and execution control techniques that led to lifetime reliability improvement.

The previous works commonly approached the lifetime reliability problem via heuristic methods using thread scheduling, power or thermal regulation, power gating, etc. These techniques from experimental results give deductive explanations about how they worked but do not provide fundamental reasoning why those techniques were needed to improve reliability. With the presence of the gap between device physics and application behaviors, DRM has to rely on devising plausible heuristics that can be implemented in various ways.

III. Experimental Methodology

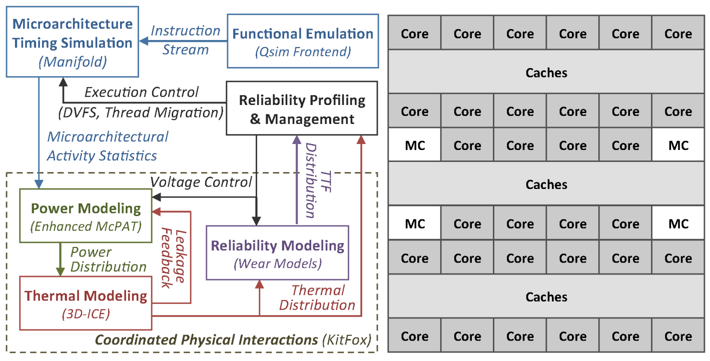
Our analysis is based on a full-system cycle-level simulation framework comprised of an application functional emulator, microarchitectural timing simulator, and coordinated interactive physics modeling library, as illustrated in Figure 1-(a). We present an overview of this framework and describe a method of microarchitecture-level lifetime reliability modeling.

A. Full-System Cycle-Level Simulation Framework

The Manifold microarchitectural timing simulator [19] is configured to model a 32-core homogeneous processor as shown in Figure 1-(b). Cores with coherent cache hierarchy [1] are connected via a 6x6 torus network. The Qsim (QEMU-based) front-end emulator [8] boots a Linux kernel and executes x86 parallel application binaries to drive the microarchitecture cycle-level timing models. PARSEC and SPLASH-2 benchmarks [2] are used in the experiment. In each simulation, a benchmark is fast-forwarded to the region of interest to skip initialization phases, and then timing simulation is performed until the benchmark finishes. Benchmarks are executed in multi-threaded mode using all 32 cores of the processor. A summary of the simulation setup is listed in Table I.

TABLE II. Failure Models and Parameters [15]

Failure Types	Models & Description
hot carrier injection (HCI) [20]	Electrons with sufficient kinetic energy overcome the barrier to gate oxide and cause degradation. $\lambda_{\text{HCI}} = \alpha \times {V_{ds}}^n \times e^{-E_a/kT}$ α= tech-dependent, $n = 3, E_a = -0.1$ k= Boltzmann's const., T= temperature
electro-migration (EM) [17]	Directional transports of electrons in interconnect causes degradation. $\lambda_{\text{EM}} = \alpha \times J^n \times e^{-E_a/kT}$ J= current density, $n = 2, E_a = 0.9$
negative bias temperature instability (NBTI) [20]	Gradual degradation causes threshold voltage shift and timing errors. $\lambda_{\text{NBTI}} = \alpha \times {V_{gs}}^n \times e^{-E_a/kT}$ $n = 5, E_a = 0.4, V_{dd}$= supply voltage
stress migration (SM) [17]	Differences in the expansion rates of metals cause mechanical stress. $\lambda_{\text{SM}} = \alpha \times (T_0 - T)^n \times e^{-E_a/kT}$ $T_0 = 500, n = 2.5, E_a = 0.9$
time-dependent dielectric breakdown (TDDB) [17]	Wearout of gate oxide leads to short between gate and substrate. $\lambda_{\text{TDDB}} = \alpha \times {V_{gs}}^{c(a+bT)} \times e^{\frac{x+y/T+zT}{kT}}$ $a = 78, b = -0.081, c = 0.1,$ $x = -0.759, y = 66.8, z = 8.37e^{-4}$



Microarchitecture Timing Simulation (Manifold)
Instruction Stream
Functional Emulation (Qsim Frontend)
Execution Control (DVFS, Thread Migration)
Reliability Profiling & Management
Microarchitectural Activity Statistics
Voltage Control
TTF Distribution
Power Modeling (Enhanced McPAT)
Leakage Feedback
Reliability Modeling (Wear Models)
Power Distribution
Thermal Modeling (3D-ICE)
Thermal Distribution
Coordinated Physical Interactions (KitFox)
Core Core Core Core Core Core
Caches
Core Core Core Core Core Core
MC Core Core Core Core MC
Caches
MC Core Core Core Core MC
Core Core Core Core Core Core
Caches
Core Core Core Core Core Core

(a) (b)

Figure 1. (a) Full-system cycle-level microarchitecture simulation framework with coordinated interactive physics modeling [14], [15]. (b) 32-core homogeneous processor floor-planning.

The KitFox framework [14] is used to coordinate the interactions among multiple physical models including McPAT [9] and 3D-ICE [16]. In particular, McPAT is substantially enhanced to support transient power modeling with dynamic voltage and frequency scaling (DVFS) and leakage-temperature feedback. 3D-ICE is configured to simulate a 2D package with a conventional air cooling model. At the device level, we use the wear models listed in Table II [15]. These models are adjusted to meet 5 years of processor-level mean-time-to-failure (MTTF) at the baseline condition (defined at T=65°C and V=0.8V), and this is referred to as baseline time-to-failure (TTF) in the experiment discussions.

In the full-system microarchitecture and physics simulation framework, multiple physical models are simultaneously simulated with a microarchitectural timing simulator, as illustrated in Figure 1-(a). Execution of application binaries through the front-end functional emulator feeds microarchitectural timing simulator with instructions to simulate. The microarchitecture simulator collects access counters of functional components and timing information that are used to estimate the power dissipations of modeled components. Power results are mapped onto the floor-plan blocks in Figure 1-(b), and the thermal field is calculated at the package level. Changes in temperature incur feedback interactions between temperature and leakage power. Cumulative failure rates are calculated at the floor-plan blocks with respect to time-varying operating conditions (i.e., voltage and thermal states). The chain of these physical interactions creates a loop and is repeated during the transient simulation. Based on collected reliability profiles, dynamic management controls are applied, such as thread migration, DVFS, etc. This infrastructure and approach enable us to explore how the executions of parallel applications create device-level degradation variation across cores in the processor and assess how this variation manifests itself as processor lifetime reliability and performance.

B. *Microarchitecture-Level Lifetime Reliability Modeling*

With billions of transistors in a chip today and continuously increasing device density at every technology node, processor-level reliability modeling becomes a statistical analysis. Since these failure mechanisms reflect long-term behaviors, they are impractical to simulate across the processor at the cycle level. Therefore, higher level abstractions are generally employed in lifetime reliability studies such as using representative thermal profiles (e.g., average temperature) to estimate lifetime reliability or Monte Carlo simulations that generate random samples to mimic unknown microarchitectural behaviors. However, such an approach via high-level abstraction fundamentally prohibits connecting application behaviors to device-level physics. In contrast, we utilize the detailed modeling of interacting physical behaviors to calculate failure rates and predict lifetime. Transient failure rates are calculated with respect to time-varying stress conditions including voltage and thermal states that are induced by workload dynamics and microarchitectural operations. We use common exponential models to express the failure rate λ of different failure mechanisms listed in Table II. Although Weibull or lognormal distributions are known to better represent long-term degradation behaviors (e.g., measured in years), they can be simplified to a constant failure rate modeling in cycle-level microarchitecture simulations that typically span over seconds in real time as shown in Figure 2. Relative changes in failure rate and its projection to lifetime are used in our experiment to assess reliability criticality of different applications or dynamic execution controls (e.g., turbo-mode executions).

$$\lambda = \sum_{r \in R} p_r \lambda_r \quad \text{and} \quad \sum_{r \in R} p_r = 1 \tag{1}$$

With R different failure models, the failure rate of a component is expressed as Eq. (1) that is a weighted average for all $\lambda_{r \in R}$. Since the relative criticality of different failure mechanisms is not known, we assume these failures are

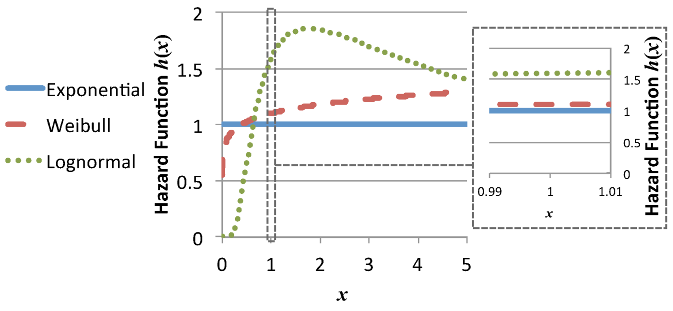


Figure 2. Effect of selecting reliability distribution functions in cycle-level microarchitecture simulations.

equally likely at the baseline condition (defined at T=65°C and V=0.8V in the experiment) [3], [6], [15], [17], [18].

$$\lambda_{(t=t_n)} = \sum_{i=1}^{n} \left\{ \sum_{r \in R} p_r \lambda_{r,(t=t_i - t_{i-1})} \times \frac{(t_i - t_{i-1})}{t_n} \right\} \tag{2}$$

In the microarchitecture and physics simulation, the failure rate λ_r of each failure model changes over time with temperature and voltage variations. Total failure rate at $t = t_n$ with t_i time steps ($i = 1, 2, ..., n$) is expressed as Eq. (2). $\lambda_{\text{TOTAL}(t=t_n)}$ in this equation denotes the failure rate based on the λ trends up to $t = t_n$. TTF due to such trends is calculated as TTF $= 1/\lambda_{(t=t_n)}$, where the failure rate reflects the operation history between (t_0, t_n]. We notice from a preliminary study that using representative thermal profiles (e.g., average temperature) may overestimate the TTF especially for highly variant applications, since it neglects high temperature phases that accelerate aging effects.

IV. Lifetime Reliability Characterization

Using the microarchitecture and physics simulation, this section presents a characterization of workload-induced degradation patterns on the multicore die and discusses about how these patterns affect processor lifetime reliability.

A. *Characterizing Spatial Distribution of Degradation*

Non-uniform degradation in a multicore processor leads to variations in core-level lifetime. Processor TTF depends on how many failed components would be tolerated until the processor is regarded as inoperable. To determine the failure of the processor, *operability threshold* [15] is defined as Eq. (3). The processor is regarded as failed if there are smaller number of operable cores than the operability threshold. Therefore, processor TTF is determined by the operability threshold.

$$\textit{Operability Threshold} = \frac{\textit{Minimum \# of operable cores}}{\textit{Total \# of cores}} \tag{3}$$

Based on the experimental results of the full-system microarchitecture and physics simulation, core-level TTF distribution on the multicore die is characterized. At the end of simulation of each benchmark, mean (μ) and variance (σ^2) are calculated from the samples of core TTF in the 32-core processor shown in Figure 1-(b). For an unidentified distribution, we assumed a normal-like distribution and generated random

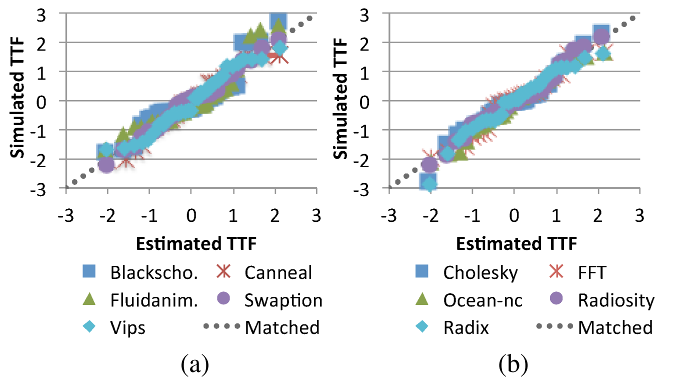


Figure 3. Comparison between the simulated TTF and estimated TTF based on a normal distribution model for (a) PARSEC and (b) SPLASH-2 benchmarks. TTF distributions are normalized to the standard normal.

TABLE III. Reliability Characterization: Normal Distribution Models of PARSEC and SPLASH-2 Benchmarks

Parsec	$N(\mu, \sigma)$	Splash-2	$N(\mu, \sigma)$
Blackscholes	$N(1.285, 0.097)$	Cholesky	$N(1.499, 0.112)$
Canneal	$N(2.264, 0.019)$	FFT	$N(2.224, 0.019)$
Fluidanimate	$N(1.398, 0.121)$	Ocean-nc	$N(2.309, 0.014)$
Swaptions	$N(1.824, 0.060)$	Radiosity	$N(1.682, 0.098)$
Vips	$N(2.056, 0.032)$	Radix	$N(2.178, 0.026)$

samples with the same mean and variance calculated from the simulation results. Figure 3 shows that the generated normal distribution samples closely match the simulation results. This reveals that core TTF distribution on the multicore die due to parallel execution of applications can be characterized by using a normal distribution model. Table III lists the reliability characteristics of PARSEC and SPLASH-2 benchmarks based on the normal distribution model; mean and standard deviation are normalized to the baseline TTF, and $\mu = 1.0$ means the baseline TTF. This observation brings a new insight of characterizing the lifetime reliability distribution in a multicore processor. In the subsequent sections, we utilize this observation and use the mean (μ) and variance (σ^2) of core TTF distribution to analyze application-induced reliability behaviors in the multicore processor. Notably, mean and variance are general characteristics of any random distributions, and the proposed approach in this paper can be applied to other distribution models [15].

B. *Effect of Reliability Variance on Processor Lifetime*

Based on the observation of normally distributed degradation on the multicore die, we analyze how the variance of the distribution affects processor lifetime reliability. Figure 4-(a) shows the TTF changes of two different cases; i) high TTF average with high variance and ii) low TTF mean with low variance. The first case represents the situation that the processor has low average degradation with large non-uniformity in degradation distribution on the multicore core, whereas the second case is that the processor has more degradation on average but with relatively even degradation pattern across cores. We assume that the most practical use case is within 10% failing (or above 90% operability threshold) [13], denoted by *practical region of lifetime* in the figure. In this region, avoiding early failures effectively leads to processor lifetime improvement. Thus, reducing the variance of the degradation distribution is a key to enhancing processor lifetime. This concept is referred to as *dynamic reliability variance management* (DRVM) [15] and illustrated in Figure 4-(b). This is distinct from wear leveling that typically refers to evening out activity in cores using various heuristics.

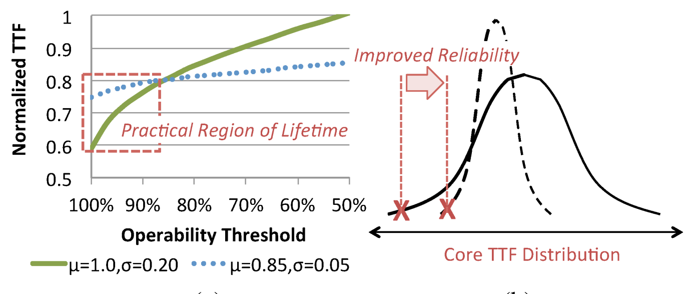


Figure 4. (a) In the practical region of lifetime, the distribution with lower variance provides better reliability even with lower mean [15]. (b) Reshaping the TTF distribution by reducing the variance improves processor lifetime.

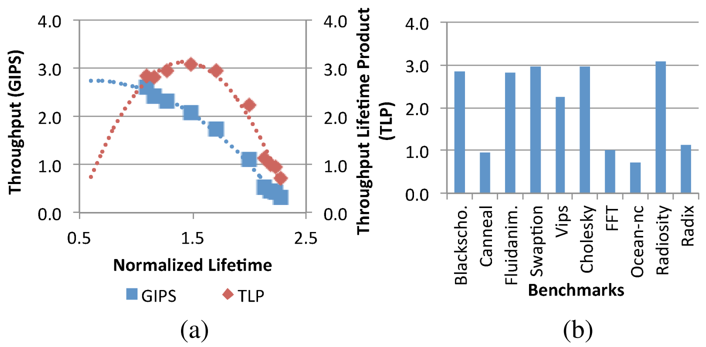


Figure 5. (a) Inverse relation between performance and lifetime reliability. (b) throughput-lifetime product evaluation of simulated benchmarks.

V. Performance and Reliability Tradeoff

There exists a fundamental tradeoff between performance and lifetime reliability. High performance operations generate more switching activities of microarchitecture components accompanied by increased power and heat dissipations that accelerate device degradation processes. On the other hand, lifetime reliability favors lower utilization to reduce stresses and resulting degradation. Therefore, DRM techniques cannot simply work to improve lifetime but must balance the tradeoff between performance and reliability.

Throughput and lifetime are inversely related as shown in Figure 5-(a). Each square mark in the graph is plotted based on the simulation results in Table III, and a trend line is shown. In each simulation of a benchmark, instruction counts and execution time were measured, and average Giga instructions per second (GIPS) was calculated to represent processor throughput. The x-axis is processor lifetime calculated as $(\mu - 2\sigma)$, where μ and σ are obtained from the simulation results (see Table III). With the normal distribution model, $(\mu - 2\sigma)$ corresponds to approximately 97% operability threshold such that 32-core processor would tolerate one core failure. When the processor has high throughput (e.g., execution of compute-bound applications), there are more switching activities at microarchitectural components, which increase power and heat dissipations. Consequently, the executions of such workloads have adverse impact on lifetime reliability. Reliability may be naively improved by regulating processor operation and power consumption, but it is traded with performance loss. To evaluate this tradeoff, we introduce a new metric, *throughput-lifetime product* (TLP), expressed as Eq. (4). In this equation, gross throughput of the processor is represented with GIPS of all cores, and lifetime is expressed by $(\mu - 2\sigma)$.

$$\mathrm{TLP} = \mathrm{GIPS} \times (\mu - 2\sigma) \tag{4}$$

In Figure 5-(a) TLP shows a parabolic trend as a function of lifetime. Due to inverse relation between performance and lifetime reliability, high throughput operations decrease lifetime and produce low TLP. At the other end of the curve, low performance has longer lifetime, but it also exhibits poor TLP. For example, execution of the Ocean benchmark generates minimal stresses on the processor and results in the greatest lifetime as shown in Table III. However, when evaluated with TLP, it is the worst benchmark in that it produces less throughput for the amount of degradation occurred during the execution. The use of TLP provides us with a quantified method to evaluate performance and reliability tradeoff. We study microarchitectural approaches to manage the performance-reliability tradeoff in the multicore processor.

VI. Microarchitectural Approaches to Manage Performance-Reliability Tradeoff

In this section, we present DRVM-based microarchitectural techniques to improve the TLP. Presented DRVM techniques are i) phase-aware thread migration (PATM), ii) dynamic voltage scaling (DVS) for reliability variance management, and iii) turbo-mode execution (TME) combined with DVS-based reliability variance management. Although thread migration and DVS have appeared in various implementations for lifetime reliability management [3], [4], [7], [11], [12], [17], the contribution is that these techniques are grounded in fundamental behaviors of degradation variation that lead to manage the processor performance and lifetime reliability tradeoff. The implementations of these techniques are first described, and then the results are discussed.

A. Phase-Aware Thread Migration for DRVM

Thread migration is a dynamic thermal management (DTM) technique that are used to spreads out thermal hotspots and also suggested for DRM [4]. However, DRM fundamentally differs from DTM in that management decisions based on instantaneous properties such as instructions per second (IPC) or temperature readings [3], [4] do not necessarily reflect "cumulative" degradation behaviors. For similar reasons, relying only on degradation monitoring causes inefficiency in reliability management. For instance, if a thread of a hot core enters a low performance (e.g., memory-bound) or idle state, it is better to keep the thread in the same core rather than swapping the threads with other cores that may have transitioned to high power state and thus will exacerbate the problem when moved to the hot core. Therefore, thread migration have to utilize both instantaneous performance metrics and degradation monitoring for efficient reliability management.

In phase-aware thread migration, the cumulative failure rates of cores are measured at every monitoring interval, considering time-varying thermal histories. We assume that degradation monitoring is available on the die, such as using aging sensors. The TTF of cores are predicted from the failure rates, and μ and σ of the core TTF distribution are calculated. Since DRVM is to reduce the degradation variance to enhance processor lifetime, thread migration is triggered only when the variance of core TTF exceeds a threshold set as $\sigma_{\text{th}} = 0.05$ in the experiment. It also monitors the IPC of threads being executed on different cores. Low IPC operations generate less switching activities of microarchitectural components and thus decrease power and heat dissipations. Such threads are relocated to weak cores to reduce degradation. Thus, when the phase-aware thread migration is invoked ($\sigma > \sigma_{\text{th}}$), it swaps threads between the cores that have the lowest IPC and TTF. Similarly, thread swapping is made between the cores that have the highest IPC and TTF to avoid μ of the distribution being biased to high TTF cores. Such "coordinated" thread swapping is applied to the next set of cores if their expected TTF deviate more than σ_{th} from the mean of the distribution. After migrating the threads, these cores are protected from being invoked for another thread migration to avoid threads being continuously tossed around cores and also to allow the cores to recover from deviated degradation status. A weakness of using the thread migration technique for DRVM is that degradation adjustment is not precisely controllable but strongly depends on the behaviors of migrating threads.

B. Dynamic Voltage Scaling for DRVM

DVS exploits the voltage impact on reliability to adjust the degradation levels of cores for DRVM. DVS techniques were widely studied for power and thermal controls as well as reliability management [4], [7], [11], [12]. However, voltage scaling for DRVM differs from previous works in that it attempts to reshape the core TTF distribution by reducing variance (σ^2) but does not control the mean (μ) such that the impact on application performance is minimized.

The difficulty is in determining the degree of voltage scaling required to adjust the degradation by a desirable amount, since failure rates are the functions of both voltage and thermal stresses and they have different impacts on different failure models. To simplify the problem, we created a voltage scaling table with predicted TTF changes at the baseline condition, shown in Table IV. The cumulative failure rates of cores

TABLE IV. Voltage Scaling for DRVM

$\mu_{\text{core}} - \mu$	Voltage	$\mu_{\text{core}} - \mu$	Voltage
-0.10	0.774V	+0.10	0.828V
-0.15	0.762V	+0.15	0.843V
-0.20	0.750V	+0.20	0.860V
-0.30	0.729V	+0.30	0.894V

are measured at every sampling interval, and μ and σ of core TTF distribution are calculated. Distance to the mean of TTF distribution is measured for each core, expressed as ($\mu_{\text{core}} - \mu$) in the voltage scaling table. Then, necessary voltage adjustment is applied. Voltage scaling in general has better controllability for DRVM than thread migration, but it may have a negative impact on performance due to frequency changes accompanied by voltage scaling.

C. Turbo-Mode Execution with DRVM

When a core is boosted such as turbo-mode execution, the failure rate gradually rises due to increased voltage and thermal stresses. However, if the turbo mode is applied for a relatively short duration compared to overall execution time, the changes in failure rate can be kept small while improving performance. We combine the turbo-mode execution with DRVM using voltage scaling to achieve performance improvement with minor impact on or even improved processor lifetime reliability.

In Eq. (4), turbo-mode execution attempts to increase the throughput term (i.e., GIPS) but instead sacrifices μ due to accelerated degradation caused by increased voltage and thermal stresses. In our preliminary study, we find that turbo-mode execution also amplifies σ, which exacerbates the reliability problem by increasing non-uniformity in degradation distribution. We learn that the reliability penalty is greater than performance benefit when turbo-mode execution is engaged. Therefore, turbo-mode execution can only be effective if it returns a good performance improvement, or otherwise decrease in lifetime reliability (i.e., $\mu - 2\sigma$) due to accelerated degradation dominates the throughput increase and thus diminishes the TLP. Contentions for shared resources prohibit turbo-mode execution from increasing throughput [10]. In the 2-level cache hierarchy of the 32-core processor model in the experiment, we use L1 cache miss rate and IPC to decide if the turbo-mode execution can be triggered. When the turbo-mode execution is not employed, this technique performs DRVM via voltage scaling as the base operation to mitigate reliability penalty due to turbo-mode execution.

D. DRVM and TLP Evaluation

Since the key to DRVM is to reduce the degradation variance to enhance processor lifetime, we select four benchmarks that have large non-uniformity in their core TTF distributions (i.e., $\sigma \gg \sigma_{\text{th}} = 0.05$) from the application characterization results in Table III; Blackscholes, Fluidanimate, Cholesky, and Radiosity. The presented DRVM techniques are applied to these benchmarks, and the results are discussed. Although we observe that the DRVM also works for other benchmarks, there are marginal improvements since these applications have

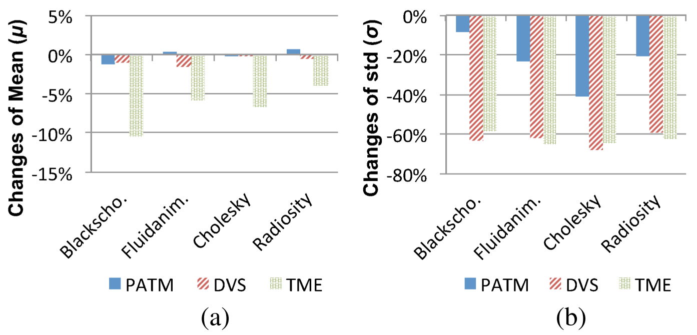


Figure 6. Changes of (a) mean μ and (b) standard deviation σ of core TTF distribution by applying DRVM with phase-aware thread migration (PATM), dynamic voltage scaling (DVS), and turbo-mode execution (TME).

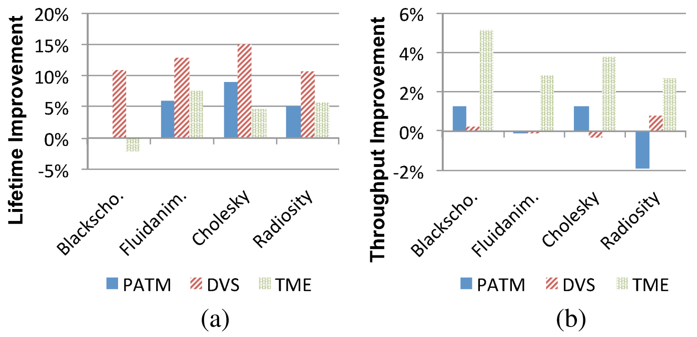


Figure 7. (a) Lifetime and (b) throughput improvement by DRVM techniques.

relatively even degradation across cores. In addition, these benchmarks do not have much performance increase when the turbo-mode is applied due to memory-bound behaviors. Therefore, in this section we discuss the results of DRVM with selected benchmarks and evaluate their performance impact.

Both phase-aware thread migration and voltage scaling for DRVM attempt to reshape the core TTF distribution by reducing the variance (σ^2) but does not control the mean (μ). Figure 6 shows the changes in mean and standard deviation of the curves by applying DRVM techniques, compared to uncontrolled executions. Thread migration or voltage scaling has minor changes to the mean but greatly reduces the standard deviation, which leads to the improvement of processor lifetime reliability, as shown in Figure 7-(a). For instance, voltage scaling applied to the Fluidanimate benchmark decreases the mean of core TTF distribution by 2% (normalized to baseline TTF), but 62% reduction in standard deviation compensates for the shift of the mean, resulting in 13% improvement in lifetime. The results show that voltage scaling produces greater lifetime improvement than thread migration. The effectiveness of thread migration is limited by the behaviors of migrated threads. The PATM works for applications with distinct power states across threads, but it does not perform well with those with similar power dissipations across cores, such as Blackscholes. Migrating the threads of Blackscholes does not effectively re-distribute thermal stresses and therefore shows no lifetime improvement over uncontrolled execution. Collec-

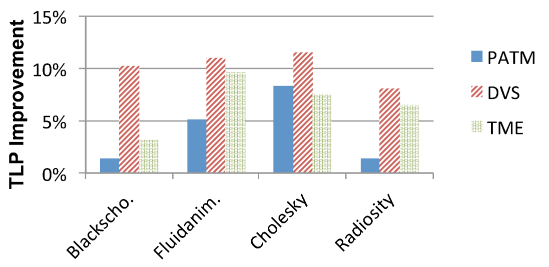


Figure 8. Improvement of throughput-lifetime product by DRVM techniques.

tively, both phase-aware thread migration and voltage scaling for DRVM improve the performance and reliability tradeoff, by enhancing processor lifetime but with little effect on the throughput. Therefore, Figure 8 shows that TLP improvements by these two techniques have similar trends as the changes in lifetime shown in Figure 7-(a).

Turbo-mode execution combined with DRVM via voltage scaling attempts to increase performance by sacrificing the mean, where decrease in lifetime reliability is mitigated by DRVM. As plotted in Figure 6-(a), turbo-mode execution significantly decreases the mean of core TTF distribution, which indicates accelerated degradation. Since DRVM with voltage scaling has little impact on the mean, the changes are primarily caused by the turbo-mode execution. Instead, DRVM reduces the variance of core TTF distribution, as shown in Figure 6-(b). As a result, turbo-mode execution with DRVM does not reduce reliability, but rather it may improve lifetime depending on workload characteristics. Compute-bound workloads such as Blackscholes operate longer in turbo mode since they can benefit from accelerated execution. Those benchmarks therefore experience more degradation and reduce processor lifetime that is traded with throughput improvement. Consequently, turbo-mode execution with DRVM improves the performance and reliability tradeoff, measured by the TLP. Importantly, we find that the reliability penalty of turbo-mode execution is greater than the throughput benefit. Therefore, the TLP improvement of this operation is less than that of voltage scaling-based DRVM without turbo-mode executions.

VII. Conclusion

Microarchitectural approaches such as dynamic reliability management have gained favor as cost-efficient methods to enhance processor lifetime reliability. However, without understanding the basic physics between device-level operations and application behaviors, dynamic reliability management has to rely only on heuristic approaches. In this paper, we have characterized how the parallel executions of applications created degradation distribution on the multicore die and affected processor lifetime reliability. A finding revealed that reducing the variance of degradation distribution effectively led to processor lifetime improvement with minimal impact on performance. Most importantly, we claim that dynamic reliability management cannot simply work to improve lifetime but must balance the tradeoff between processor performance and reliability. We foresee that reliability will be an important determinant of microarchitectural operations in future processors in conjunction with performance or energy efficiency metrics.

Acknowledgement

This research was supported by the Semiconductor Research Corporation (SRC) under task #2318.001, IBM/SRC Graduate Fellowship, and Sandia National Laboratories.

References

[1] J. Beu, M. Rosier, and T. Conte, "Manager-Client Pairing: A framework for implementing coherence hierarchies," *IEEE/ACM Int. Symp. Microarchit.*, Dec. 2011.

[2] C. Bienia, S. Kumar, and K. Li, "PARSEC vs SPLASH-2: Quantitative comparison of two multithreaded benchmark suites on processors," *IEEE Int'l Symp. Workload Charact.*, Sep. 2008.

[3] A. Coskun, T. Rosing, K. Mihic, G. Micheli, and Y. Leblebici, "Analysis and optimization of MPSoC reliability," *J. Low Power Elect.*, Jan. 2006.

[4] A. Coskun, R. Strong, D. Tullsen, and T. Strong, "Evaluating the impact of job scheduling and power management on processor lifetime reliability for chip multiprocessors," *Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2009.

[5] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, "Maestro: Orchestrating lifetime reliability in chip multiprocessors," *Conf. High Perform. Embedded Archit. Compil.*, Jan. 2010.

[6] L. Huang and Q. Xu, "Characterizing the lifetime reliability of manycore processors with core-level redundancy," *Int. Conf. Comput.-Aided Des.*, Nov. 2010.

[7] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," *Des. Autom. Conf.*, Jul. 2006.

[8] C. Kersey, A. Rodrigues, and S. Yalamanchili, "A universal parallel frontend for execution driven microarchitecture simulation," *Workshop Rapid Simul. Perform. Eval.*, Jan. 2012.

[9] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: Integrated power, area, timing modeling framework for multicore architectures," *IEEE/ACM Int. Symp. Microarchit.*, Dec. 2009.

[10] D. Lo and C. Kozyrakis, "Dynamic management of TurboMode in modern multi-core chips," *IEEE Int. Symp. High Perform. Comput. Archit.*, Feb. 2014.

[11] Z. Lu, J. Lach, M. Stan, and K. Skadron, "Improved thermal management with reliability banking," *IEEE Micro*, Dec. 2005.

[12] P. Mercati, A. Bartolini, F. Paterna, T. Rosing, and L. Benini, "Workload and user experience-aware dynamic reliability management in multicore processors," *Des. Autom. Conf.*, Jun. 2013.

[13] V. Morozov, K. Kumaran, V. Vishwanath, J. Meng, and M. Papka, "Early experience on the BlueGene/Q supercomputing system," *IEEE Int. Parallel Distrib. Process. Symp.*, May 2013.

[14] W. Song, S. Mukhopadhyay, and S. Yalamanchili, "Energy Introspector: Parallel, composable framework for integrated power-reliability-thermal modeling for multicore architectures," *IEEE Int'l Symp. Perform. Anal. Syst. Softw.*, Mar. 2014.

[15] W. Song, S. Mukhopadhyay, and S. Yalamanchili, "Architectural Reliability: Lifetime reliability characterization and management of many-core processors," *Comput. Archit. Lett.*, Jul. 2014.

[16] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," *Int. Conf. Comput.-Aided Des.*, Nov. 2010.

[17] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Lifetime Reliability: Toward an architectural solution," *IEEE Micro*, May 2005.

[18] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," *Int. Symp. Comput. Archit.*, Jun. 2005.

[19] J. Wang, J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao, P. Xu, and S. Yalamanchili, "Manifold: A parallel simulation framework for multicore systems," *IEEE Int'l Symp. Perf. Anal. Syst. Softw.*, Mar. 2014.

[20] M. White and J. Bernstein, "Microelectronics Reliability: Physics-of-failure based modeling and lifetime evaluation," JPL Publication 08-5 2/08, NASA Jet Propulsion Laboratory, 2008.