LIFETIME RELIABILITY AWARE MICROPROCESSORS

BY

JAYANTH SRINIVASAN

B.Tech., Indian Institute of Technology, Madras, 2000
M.S., University of Illinois at Urbana-Champaign, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Abstract

Ensuring long-term, or "lifetime" reliability, as dictated by the hard error rate due to wear-out based failures, is a critical requirement for microprocessor manufacturers. At the same time, the steady increases in CMOS processor performance have been driven by aggressive device scaling. This continuous scaling coupled with increasing temperatures on chip are making lifetime reliability targets increasingly difficult to meet. This dissertation addresses lifetime reliability issues from a microarchitectural perspective. Our key contributions include (i) the first architecture-level methodology for evaluating lifetime reliability, as a function of application behavior, (ii) a quantification of the impact of device scaling on lifetime reliability, taking workload characteristics into consideration, and (iii) performance and cost-effective architectural solutions targeted at enhancing lifetime reliability.

The first part of this dissertation focuses on the design of tools and models to evaluate processor lifetime reliability. Using industrial strength models for lifetime reliability modes, we develop a methodology, called *RAMP*, to estimate lifetime reliability from an architectural and application perspective. We propose two implementations of RAMP, RAMP 1.0 and RAMP 2.0, which differ in their utility and accuracy.

This dissertation also extends the RAMP methodology by adding scaling models for different technology generations to its failure mechanisms. Our quantification of the impact of scaling on a contemporary superscalar processor shows that device scaling has a significant detrimental impact on processor hard failure rates.

The second part of this dissertation examines a range of microarchitectural techniques for lifetime reliability enhancement. In contrast to previous application-oblivious methods, these techniques allows processor designers to trade-off cost, performance, and reliability in an application-aware fashion. First, we propose *dynamic reliability management (DRM)* where the processor uses adaptive hardware to dynamically respond to changing application behavior to maintain its lifetime reliability target. Our results show that DRM enables the processor to extract significant performance benefit for a spectrum of reliability design costs.

Next, we study two techniques that leverage microarchitectural structural redundancy for lifetime reliability enhancement. Structural redundancy has the potential to be more cost and performance effective

than traditional processor redundancy. In *structural duplication*, redundant microarchitectural structures are added to the processor and designated as spares. Spare structures can be turned on when the original structure fails, increasing the processor's lifetime. *Graceful processor degradation* is a technique that exploits existing microarchitectural redundancy for reliability. Redundant structures that fail are shut down while still maintaining functionality, thereby increasing the processor's lifetime, but at a lower performance. Our evaluation shows significant reliability benefit from these techniques for a range of cost and performance budgets.

Overall, this dissertation lays the basic foundation for microarchitectural analysis of lifetime reliability and provides new tools and techniques to handle this critical emerging technology challenge.

To Appa and Amma,

and a grandfather's belief in his six year old grandson.

Acknowledgements

I cannot hope to enumerate, let alone repay, all those who helped me prosper in my five years at Illinois. Despite the unavoidable omissions, I still insist on mentioning a few in print.

I was extraordinarily fortunate that Sarita Adve took me on as her graduate student in 2000. I've grown and matured tremendously under her sage guidance, and for that I cannot thank her enough. Sarita's intellect, patience (which I stress-tested many a time), and commitment to both my professional and personal growth made her a pleasure to work with.

In an area like Computer Architecture, it is crucial that one balances academic exploration with industrial relevance. Your research is that much more powerful when you understand its near and long-term ramifications – in some small way, you are helping maintain the fantastic growth rate in computing power! In this respect, I was fortunate to have spent three summers at IBM Research and Intel Research. In particular, my stay at IBM helped me define an interesting and rewarding Ph.D. problem in lifetime reliability, and for this, I owe a debt of gratitude to my immensely knowledgeable mentors, Pradip Bose and Jude Rivers. I thank Pradip and Jude for giving me the opportunity to interact with IBM's world class researchers at the beautiful Yorktown Heights campus. The summer evenings I spent in Pradip's room at IBM sharing in his passion for computer architecture are some of the fondest memories I will take away from my Ph.D.

I would also like to thank the members of my Ph.D. committee, Marc Snir, Josep Torrellas, Sanjay Patel, and Craig Zilles for their time and valuable feedback. Josep was a constant source of support and mentorship during my years in the Computer Science department, and Craig was more a friend than a professor.

Working in a lab would be quite monotonous were it not for the constant interaction with my office mates and friends at the department. Thanks especially to the members of the RSIM team (or is it the 4P team now?), past and present – Alex, Chris, Jerry, Rohit, Ruchira, Pradeep, Praful and Vibhore. Chris was a wonderful guide during my first few years at Illinois, and Rohit became a very close personal friend. Pradeep's help was invaluable in my final days at the department, in everything from identifying errors in my thesis to the actual deposit process. And Vibhore, I was the one stealing your food.

I would also like to thank the able and incredibly friendly staff of the Computer Science department.

They've gone out of their way many times to help disorganized, deadline-missing graduate students like me, and I don't think anyone can escape the maze that is graduate school without their assistance. Thanks in particular to Sheila Clark who is the epitome of efficiency and professionalism.

My graduate career was supported by an IBM PhD fellowship in 2004-05. I would also like to acknowledge the following funding for different parts of my work: an equipment donation from AMD Corp., a gift from Motorola Inc., and grants from the National Science Foundation (Grants CCR-0096126, CCR-0209198, CCR-0205638, CCR-0313286, EIA-0103645, EIA-0224453).

There's more to life than work, and the wonderful friends I made at Illinois (and before) are a testament to that. I'm not even going to try to name everyone, but some people stand out - Thank you Sumant for all the coffee. Thank you Andy for cleaning the tub (and for everything else). Thank you Devatha for showing me that there are people in the world who do stupider things than me. Thank you Arun for all those crazy days (and nights!). Thank you Walter for all the insensitive jokes. Thank you Harshaw for always making time for me when I came home to Madras. Thank you Shankar for always doing better than me – I always have something to aspire to when you're around. And thank you Arvind for asking me to be your wingman.

And beyond friends, there's family. Appa, Amma, and Shylu have been there all along. They have always supported me and their sacrifices and love is the reason I have been able to get this far. They have inspired me both on a personal and professional level throughout my life. Appa and Amma, even if I don't always show it, know that I owe all my successes to you. As a tiny step towards showing my gratitude, I dedicate this thesis to you. And Shylu, someone as kind and compassionate as you will become a far better doctor than I could ever hope to be.

I also need to thank the family I have inherited through marriage. They, along with the Chicago Bunts, have taken me in as one of their own, and for that I am thankful.

I found something in Champaign far more valuable than a graduate education. I found Smitha. I found someone who brings joy into my life everyday. I found someone who supports me in times of doubt, and has enriched my life beyond measure. I found a best friend.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Ensuring long-term, or "lifetime" reliability, as dictated by the hard error rate due to wear-out based failures, is a critical requirement for all microprocessor manufacturers. However, relentless technology scaling coupled with increasing processor power densities are threatening the nearly unlimited lifetime reliability standards that customers have come to expect. This has led the International Technology Roadmap for Semiconductors (ITRS) to predict the onset of significant lifetime reliability problems, and at a pace that has not been seen in the past. It is expected that in the future, product cost and performance requirements will be substantially affected, and in many cases, superseded by constraints brought on by processor wear-out and dwindling lifetime reliability [1].

Traditionally, microarchitects have treated the issue of processor lifetime reliability as a manufacturing problem, best left to be handled by device and process engineers. We observe that the microarchitecture's ability to track application behavior can potentially be leveraged to the benefit of reliability qualification, enabling reduced reliability design costs, increased processor yield, and/or increased performance. The goal of this dissertation is to design tools and techniques for microarchitectural lifetime reliability awareness and enhancement.

1.1.1 Classification of Processor Errors

Processor errors can be broadly classified into two categories: soft and hard errors.

Soft errors, also called transient faults or single-event upsets (SEUs), are errors in processor execution due to electrical noise or external radiation, rather than design or manufacturing related defects. Extensive research is being performed by the architecture community to make processors resilient to soft errors (for e.g., [2, 3]). Although soft errors can cause errors in computation and data corruption, they do not fundamentally damage the microprocessor and are not viewed as a lifetime reliability concern. As a result, this dissertation does not address soft error reliability.

Hard errors are caused by defects in the silicon or metalization of the processor package, and are usually permanent once they manifest. Hard errors, or hard failures can be further divided into extrinsic failures and intrinsic failures [4].

Extrinsic failures are caused by process and manufacturing defects and occur with a decreasing rate over time. For example, contaminants on the crystalline silicon surface and surface roughness can cause gate oxide breakdown [5]. Other examples include short-circuits and open-circuits in the interconnects due to incorrect metalization during fabrication. Extrinsic failures are mainly a function of the manufacturing process – the underlying microarchitecture has very little impact on the extrinsic failure rate. After manufacturing, using a technique called burn-in [6, 7], the processors are tested at elevated operating temperatures and voltages in order to accelerate the manifestation of extrinsic failures. Since most of the extrinsic failures are weeded out during burn-in, shipped chips have a very low extrinsic failure rate. Semiconductor manufacturers and chip companies continue to extensively research methods for improving burn-in efficiency and reducing extrinsic failure rates [6, 7].

Intrinsic failures are those related to processor wear-out and are caused over time due to operation within the specified conditions. These failures are intrinsic to, and depend on, the materials used to make the processor and are related to process parameters, wafer packaging, and processor design. If the manufacturing process was perfect and no errors were made during design and fabrication, all hard processor failures would be due to intrinsic failures. Intrinsic failures occur with an increasing rate over time. It is essential that these failures do not occur during the intended lifetime of the device when it is used under specified operating conditions [8]. Examples of intrinsic failures include time dependent dielectric breakdown (TDDB) and negative bias temperature instability (NBTI) in the gate oxides, electromigration and stress migration in the interconnects, and thermal cycling and cracking.

It should be noted that many extrinsic failures are caused by intrinsic failure mechanisms. However, due to manufacturing defects, these intrinsic failures manifest themselves very early in the processor's lifetime. As discussed, burn-in attempts to filter out all processors which manifest early-life or extrinsic failures. As a result, processor lifetime reliability is mainly dependent on wear-out failures or intrinsic hard failures. Very little microarchitectural research has been done on modeling and analyzing intrinsic failures, and these are the focus of this dissertation.

1.1.2 Main Lifetime Reliability Challenges

Although providing significant benefits in microprocessor performance, advances in technology are accelerating the onset of intrinsic failures, causing a reduction in processor lifetimes. Specifically, the three main

reasons are:

- **Processor scaling and increasing power densities.** Device miniaturization due to scaling is increasing processor power densities and eating away at process and design margins [9]. Scaling decreases lifetime reliability by shrinking the thickness of gate and inter-layer dielectrics, increasing current density in interconnects, and by raising processor temperature which exponentially accelerates wear-out failures. Scaled down transistors in deep sub-micron CMOS technologies also have significantly higher leakage power which has an exponential dependence on temperature leading to even higher processor temperatures. Finally, supply voltage and threshold voltage are not scaling appropriately with technology because of performance and leakage power concerns creating further reliability problems.

- **Increasing transistor count.** Increasing functionality, facilitated by increasing transistor densities, causes the transistor count of processors to grow rapidly. More transistors result in more failures which results in lower processor lifetimes. Hence, not only is the reliability of individual transistors decreasing, the number of transistors that can fail is also increasing.

- **The advent of on-chip power management techniques like gating and adaptive processing.** In order to cope with escalating power, most modern processor designs employ some form of gating, usually of the clock. Other forms of dynamic, workload-driven adaptation of processor resources and bandwidth are also becoming part of on-chip power management [10, 11]. These techniques are promising from the point of view of reducing average power and temperature; however, they introduce new effects on chip like thermal cycling which may have a negative impact on reliability.

In this dissertation, we focus on the the impact of processor scaling and increasing power densities.

1.1.3 Microarchitectural Awareness of Lifetime Reliability

This dissertation makes the case that lifetime reliability must be treated as a first-class design constraint at the microarchitectural design stage. This is true for all market segments ranging from server class processors where lifetime reliability is an implicit requirement, to commodity processors where screening for higher reliability impacts the number of processors shipped (yield) and resultant profit.

Extensive research has gone into techniques that can improve energy and thermal efficiency by exploiting microarchitectural features and adaptation capabilities. A similar approach can be used for lifetime reliability – the microarchitecture's unique knowledge of application run-time behavior can be leveraged to increase processor reliability. Such an approach to reliability is fundamentally different from existing methodologies where processor reliability is qualified during device design, circuit layout, manufacture, and chip test.

1.2 Contributions

This dissertation makes three key contributions.

1. RAMP, the first microarchitectural modeling methodology for lifetime reliability that accounts for application behavior [12, 13].

2. Quantification of the impact of technology scaling on the lifetime reliability of a contemporary superscalar processor [14].

3. Performance and cost-effective microarchitectural solutions for lifetime reliability enhancement [12, 15].

We now discuss each of the contributions in more detail.

1.2.1 The RAMP Model

Evaluating the potential benefit of microarchitectural lifetime reliability-awareness requires tools and models that can be used at early processor design stages. In this dissertation, we introduce RAMP (**R**eliability **A**ware **M**icro**p**rocessor), the first application-aware architecture-level methodology for evaluating processor lifetime reliability. RAMP uses state-of-the-art analytical models for five important intrinsic failure mechanisms: time dependent dielectric breakdown (TDDB) and negative-bias temperature instability (NBTI) in the transistors, electromigration (EM) and stress migration (SM) in the interconnects, and thermal cycling (TC) in the package.

We present two versions of RAMP; RAMP 1.0 is a simpler version that can be implemented both in real hardware and in a simulator. A key feature of RAMP 1.0 is its ability to be used for run-time reliability management. RAMP 2.0 is a more complex version which improves on the accuracy of RAMP 1.0. However, its additional complexity makes its use as a run-time tool difficult, restricting its use to design time evaluations.

1.2.2 Impact of Scaling on Lifetime Reliability

The second contribution of this dissertation is a quantitative evaluation of the impact of device scaling on the hard error rates and lifetime reliability of processors. We enhance the RAMP methodology by adding scaling specific parameters to enable lifetime reliability evaluation at different technology generations. In particular, our evaluation and analysis attempt to model the scaling effects of taking one chip design, and gradually scaling that chip down from 180nm to 65nm, without any substantial modifications to the microarchitectural pipeline.

Our results show that scaling has a significant and increasing impact on processor hard failure rates. The increase in processor temperature is one of the key reasons for this trend. In our experiments, on average, the maximum temperature reached by a 65nm processor is 15 degrees Kelvin higher than that reached by a 180nm processor. On average, the lifetime for a 65nm processor is 79% lower than the lifetime at 180nm, with similar reliability qualification. More importantly, the rate of decrease of lifetime increases as we scale to smaller technologies. Our results clearly demonstrate that hard failures will present a significant and increasing challenge in future technology generations. An important practical consequence is that, in contrast to current practice, leveraging a single design for multiple remaps across a few technology generations (with only minor design tweaks) will become increasingly difficult.

1.2.3 Architectural Adaptation for Reliability Enhancement

The third contribution of this dissertation is to propose architectural techniques for processor lifetime reliability enhancement.

First, we propose *dynamic reliability management (DRM)* – a technique where the processor can respond to changing application behavior to maintain its lifetime reliability target. Currently, processor reliability design done in an application oblivious fashion, and is instead based on worst-case processor operating conditions. This results in processors that are over-designed from a reliability perspective imposing an unnecessary performance and/or cost overhead. In contrast to current worst-case behavior based reliability qualification methodologies, DRM allows processors to be qualified for reliability at lower (but more likely) operating points than the worst case. This saves cost, but possibly at a loss of some performance in the unexpected case. Conversely, DRM allows processors that are over-designed for reliability for some applications to respond by improving performance for these applications. Our results show that DRM can be used to improve performance and/or lower cost while achieving reliability targets, providing the designer with a spectrum of reliability design points. This would be particularly useful in future systems where reliability cost could be too prohibitive to stay on the required performance curve.

We also explore techniques that exploit processor structural redundancy for lifetime reliability. Redundancy is a commonly used technique for lifetime reliability enhancement. However, most previous work for lifetime reliability focused on redundancy at the processor granularity (for e.g., [16, 17]). Due to the large area overheads involved in duplicating entire processors, such redundancy tends to be expensive both from a cost and performance perspective. By working at a finer granularity, structural redundancy addresses some of these shortcomings of processor redundancy by incurring less area and performance overheads.

We examine two techniques that exploit structural redundancy: *structural duplication (SD)* adds redun-

dant microarchitectural structures to the processor. These "spare" structures can be turned on during the processor's lifetime when the original structure fails, thereby extending the processor's lifetime. However, the spare structures add an area and cost overhead to the processor. *Graceful performance degradation (GPD)* is a technique that exploits existing microarchitectural redundancy for reliability. Modern processors have replicated structures that are used for increasing performance for some high ILP applications. However, the replicated structures are not required for functional correctness. If a replicated structure fails in the course of a processor's lifetime, the processor can extend its lifetime by shutting down the structure, albeit with a performance loss. Our analysis shows that structural redundancy can be exploited for significant reliability benefit. In addition, we show that different structural redundancy techniques are beneficial for different performance and cost overheads. We provide guidelines for intelligent reliability decisions by identifying the optimal design technique for a given performance or area trade-off.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 describes RAMP. Chapter 3 describes the scaling models added to the RAMP methodology and quantifies the impact of scaling on a contemporary superscalar processor, highlighting the detrimental impact of scaling on lifetime reliability. Chapter 4 discusses the improvements required to increase RAMP's validity and usability. Chapters 5 and 6 examine microarchitectural techniques for reliability enhancement. Chapter 5 proposes DRM, a technique by which adaptive hardware is used to dynamically respond to changing application behavior to maintain the lifetime reliability target. Chapter 6 examines the benefits of exploiting structural redundancy for lifetime reliability through two techniques, structural duplication and graceful processor degradation. Chapter 7 discusses related work and Chapter 8 describes the conclusions of this dissertation and possible avenues of future work.

Chapter 2

The RAMP Model

Based on extensive discussions with front-end, back-end, and reliability engineers at IBM, we determined that the critical intrinsic failure mechanisms for processors are: Electromigration (EM), stress migration (SM), gate-oxide breakdown or time dependent dielectric breakdown (TDDB), thermal cycling (TC), and negative bias temperature instability (NBTI) [18, 1, 19]. Based on these discussions, we also identified the state-of-the-art device-level analytic models for these failure mechanisms [18, 5]. These models assume steady state operation at specific (generally worst-case) temperature and utilization, and express reliability in terms of MTTF (mean time to failure or the expected lifetime of the processor). RAMP uses these models to calculate an "instantaneous" MTTF based on current temperature and utilization. Much like previous power and temperature models [20, 11], RAMP divides the processor into a few structures (e.g., ALUs, FPUs, register files, branch predictor, caches, load-store queue, instruction window) and applies the analytic models to each structure as an aggregate. In order to calculate the MTTF of the entire processor, structure-level instantaneous MTTFs obtained from RAMP must be combined. We propose two versions of RAMP, RAMP 1.0 and RAMP 2.0, which differ in their utility and accuracy.

As a simulation tool, RAMP should be used in conjunction with a timing simulator to determine workload behavior, and a power and thermal simulator for power and temperature profiles. A real hardware version of RAMP would require sensors and counters that provide information on processor operating conditions.

We discuss the individual failure models used in RAMP in Sections 2.1.1 to 2.1.5. Section 2.2 describes RAMP 1.0, and Section 2.2.2 describes how RAMP incorporates application-driven temporal variations of temperature and utilization, to calculate the processor MTTF for a given application. Section 2.3 describes the more accurate RAMP 2.0. Section 2.2.2 describes how RAMP 1.0 and RAMP 2.0 incorporate application-driven temporal variations of temperature and utilization, to calculate the processor MTTF for a given application. Section 2.4 compares some results between RAMP 1.0 and RAMP 2.0 and Section 2.5 discusses the process of reliability qualification and its incorporation in our models.

2.1 Failure Mechanisms Modeled in RAMP

2.1.1 Electromigration (EM)

Electromigration is one of the best studied and well understood failure mechanisms in semiconductor devices and occurs in interconnects. Extensive research has been performed by the material science and semiconductor community on modeling and understanding the effects of electromigration [18, 21, 22, 23, 24, 25, 26, 27].

Electromigration occurs in aluminum and copper interconnects due to the mass transport of conductor metal atoms in the interconnects. Conducting electrons transfer some of their momentum to the metal atoms of the interconnect – this "electron wind" driving force creates a net flow of metal atoms in the direction of electron flow. As the atoms migrate, there is depletion of metal atoms in one region and pile up in other regions. The depletion sites can see increased interconnect resistance or open circuits, and extrusions can occur at sites of metal atom pile up. Electromigration has an exponential dependence on temperature.

The currently accepted model for MTTF due to electromigration ($MTTF_{EM}$) is based on Black's equation [1] and is as follows [18]:

$$MTTF_{EM} \propto (J - J_{crit})^{-s} e^{\frac{E_{a_{EM}}}{kT}} \tag{2.1}$$

where J is the current density in the interconnect, J_{crit} is the critical current density required for electromigration, $E_{a_{EM}}$ is the activation energy for electromigration, k is Boltzmann's constant, and T is absolute temperature in Kelvin. s and $E_{a_{EM}}$ are constants that depend on the interconnect metal used (1.1 and 0.9 for copper interconnects as modeled in RAMP [18]). J tends to be much higher than J_{crit} in interconnects (nearly 2 orders of magnitude [18]). Hence, $(J - J_{crit}) \approx J$.

J for an interconnect can be related to the switching probability of the line, p, as [26]

$$J = \frac{CV_{dd}}{WH} \times f \times p \tag{2.2}$$

where C, W, and H are the capacitance, width, and thickness, respectively of the line and f is the clock frequency.

Equations 2.1 and 2.2 offer a convenient abstraction for computer architects to work with electromigration. Abstracting out only the architectural variables for a given process, the MTTF due to electromigration as modeled in RAMP is given by:

[1] Black's equation calculates reliability in terms of the *median* time to failure. In this thesis, we model electromigration with either exponential or lognormal failure distributions. In both cases, MTTF is directly proportional to the median.

$$MTTF_{EM} \propto \frac{e^{\frac{E_{a_{EM}}}{kT}}}{V^s f^s p^s} \tag{2.3}$$

Currently, RAMP assumes all interconnects in a structure to be similar, and does not differentiate interconnects on the basis of their C, W, and H (we currently fold these terms into the proportionality constant). The activity factor (or switching probability or utilization) of a structure, p, is obtained from the timing simulator.

2.1.2 Stress Migration (SM)

Much like electromigration, stress migration is a phenomenon where the metal atoms in the interconnects migrate. It is caused by mechanical stress due to differing thermal expansion rates of different materials in the device. This stress, σ, is proportional to the change in temperature, $T_0 - T$, where T is the operating temperature and T_0 is the stress free temperature (metal deposition temperature) of the metal. That is, when the metal was originally deposited on the device, there is no thermo-mechanical stress. At other temperatures, there are stresses due to differing expansion rates. The exact mechanisms behind stress migration are still not completely understood and research is ongoing on the subject [18].

The mean time to failure due to stress migration, $MTTF_{SM}$, as modeled in RAMP is given by [18]:

$$MTTF_{SM} \propto |T_0 - T|^{-s} e^{\frac{E_{a_{SM}}}{kT}} \tag{2.4}$$

where the temperatures are in Kelvin, and s and $E_{a_{SM}}$ are material dependent constants (2.5 and 0.9 for the copper interconnects modeled [18]). T_0 depends on whether vapor deposition or sputtering was used for depositing the metal - sputtering uses high temperatures to increase the stickiness of the deposited metal; on the other hand, vapor deposition happens near room temperature [28]. RAMP assumes that sputtering was used to deposit the interconnect metal and uses a value of 500K for T_0 [18].

The relationship between stress migration and temperature is governed by two opposing properties. The exponential temperature relationship accelerates wear-out with increases in temperature. However, since metal deposition temperatures tend to be higher than typical operating temperatures, higher operating temperatures decrease the value of $T_0 - T$, thus reducing the value of σ and increasing the MTTF. However, this increase in MTTF is typically much smaller than the exponential decrease due to temperature.

2.1.3 Time-Dependent Dielectric Breakdown (TDDB)

Time-dependent dielectric breakdown (TDDB), or gate oxide breakdown, is another well studied failure mechanism in semiconductor devices. The gate dielectric wears down with time and fails when a conductive path forms in the dielectric. When a conducting path forms between the gate and the substrate, it is no longer possible to control current flow between the drain and the source with a gate electric field, effectively rendering the transistor device useless [29, 5]. The advent of thin and ultra-thin gate oxides, coupled with non-ideal scaling of supply voltage is accelerating TDDB failure rates. The failure rate is also increasing due to the fact that the supply voltage is not scaling down appropriately with technology [1].

Various models have been proposed for TDDB degradation relating TDDB degradation to the electric field, the inverse of the electric field and the gate voltage. The TDDB model we use is based on recent experimental work done by Wu et al. [5] at IBM. Wu et al. collected experimental data over a wide range of oxide thicknesses, voltages, and temperatures to create a unified TDDB model for current and future ultra-thin gate oxides. The model shows that the lifetime due to TDDB for ultra-thin gate oxides is highly dependent on voltage and has a larger than exponential degradation due to temperature. Based on [5], the MTTF due to TDDB, $MTTF_{TDDB}$, at a temperature, T, and a voltage, V, is given by:

$$MTTF_{TDDB} \propto (\frac{1}{V})^{(a-bT)} e^{(\frac{x}{T}+\frac{y}{T^2}+z)} \tag{2.5}$$

In order to be compatible with the conventional Arrhenius temperature dependence with an activation energy, the non-Arrhenius relationship ($e^{(\frac{a}{T}+\frac{b}{T^2})}$) can be represented in the form, $e^{\frac{E_a}{kT}}$, where $E_a = A + \frac{B}{T} + CT$. This gives:

$$MTTF_{TDDB} \propto (\frac{1}{V})^{(a-bT)} e^{\frac{(X+\frac{Y}{T}+ZT)}{kT}} \tag{2.6}$$

where a, b, X, Y, and Z are fitting parameters. Based on data in [5], the values currently used in RAMP are $a = 78$, $b = -0.081$, $X = 0.759ev$, $Y = -66.8evK$, and $Z = -8.37E-4ev/K$.

2.1.4 Thermal Cycling (TC)

Fatigue failures can occur due to temperature cycling. Permanent damage accumulates every time there is a cycle in temperature, eventually leading to failure. Normal powering up and powering down also causes damage. Although all parts of the device experience fatigue due to thermal cycling, the effect is most pronounced in the package and die interface (for example, solder joints) [18].

The package goes through two types of thermal cycles – The first type are large thermal cycles that occur at a low frequency (a few times a day). These include powering up and down, or going into low power

or stand-by mode for mobile processors. The second type are small cycles which occur at a much higher frequency (a few times a second). These are due to changes in workload behavior and context switching. The effect of small thermal cycles at high frequencies has not been well studied by the packaging community, and validated models are not available. As a result, we do not discuss models for the reliability impact of small thermal cycles. Large thermal cycles are modeled using the Coffin-Manson equation [18]:

$$N_f = C_0(\Delta T)^{-q} \tag{2.7}$$

where N_f is the number of thermal cycles to failure, C_0 is an empirically determined material-dependent constant, ΔT is the temperature range experienced in the thermal cycle, and q is the Coffin-Manson exponent, an empirically determined constant.

Using Equation 2.7, we can see that the MTTF due to thermal cycling depends on the frequency of cycling, and on the magnitude of the cycles. Hence, the equation used to determine mean time to failure due to thermal cycles ($MTTF_{TC}$) is:

$$MTTF_{TC} \propto (\frac{1}{T - T_{ambient}})^q \tag{2.8}$$

where the proportionality constant also factors in the frequency of thermal cycling, which we assume stays constant. T is the average temperature of the structure and $T_{ambient}$ is the ambient temperature. Hence, $T - T_{ambient}$ represents the thermal cycle modeled. As mentioned, RAMP only models cycling fatigue in the package, since that is where the impact of cycling is most pronounced. For the package, the value of the Coffin-Manson exponent, q, is 2.35 [18].

2.1.5 Negative Bias Temperature Instability (NBTI)

NBTI is an electro-chemical reaction that takes place in PFETs when the gate is biased negative with respect to the source and drain. This typically occurs when the input to a gate is low while the output is high, resulting in an accumulation of positive charges in the gate oxide. This accumulation causes the threshold voltage of the transistor to increase. Higher threshold voltages result in gate overdrive (supply voltage - threshold voltage) decreasing, which slows down the performance of the gate. This will eventually lead to processor failure due to timing constraints [19].

NBTI has a strong positive temperature and field dependence. As a result, the higher temperatures seen on chip due to scaling exacerbate this problem. Similarly, thinning of the gate oxide due to scaling also increases NBTI reliability concerns [19].

The NBTI model we use is based on recent work by Zafar et al. at IBM, and is a physics-based model verified using new and published NBTI failure data [19]. The model shows that lifetime due to NBTI has a large dependence on temperature. The MTTF due to NBTI, $MTTF_{NBTI}$, at a temperature, T, is given by:

$$MTTF_{NBTI} \propto [(ln(\frac{A}{1+2e^{\frac{B}{kT}}}) - ln(\frac{A}{1+2e^{\frac{B}{kT}}} - C)) \times \frac{T}{e^{\frac{-D}{kT}}}]^{\frac{1}{\beta}} \tag{2.9}$$

and A, B, C, D, and β are fitting parameters, and k is Boltzmann's constant. Based on the data in [19], the values we use are $A = 1.6328$, $B = 0.07377$, $C = 0.01$, $D = -0.06852$, and $\beta = 0.3$.

2.1.6 Combining Structure-Level MTTFs

To obtain the total MTTF of the processor, we need to combine the effects of different failure mechanisms, and across structures. Since different failure mechanisms have different lifetime distributions, combining them directly is difficult. Also, we need a convenient way to add the failure contributions of individual structures.

In this dissertation, we propose two versions of RAMP, RAMP 1.0 and RAMP 2.0, which use different methods to combine individual failure mechanisms. RAMP 1.0 uses a simpler method which allows its use as a run-time tool both in real hardware and in simulators. RAMP 2.0 uses a more complex method to obtain processor MTTF. This increases the accuracy of the tool, but makes its implementation in real hardware difficult, restricting it to simulators.

2.2 RAMP 1.0

RAMP 1.0 uses the industry standard sum-of-failure-rates (SOFR) model to obtain processor MTTF.

2.2.1 Sum-of-Failure-Rates (SOFR) Model

In order to combine MTTFs, the SOFR model requires two simplifying assumptions to be made.

- The processor is a series failure system – in other words, the first instance of any structure failing due to any failure mechanism causes the entire processor to fail.
- Each individual failure mechanism has a constant failure rate (equivalently, every failure mechanism has an exponential lifetime distribution). The failure rate (also known as the hazard function), $h(t)$ at a time t, is defined as the conditional probability that a component will fail in the interval $(t + \delta t)$,

given that it has survived till time t. A constant failure rate implies that the value of $h(t)$ remains fixed, and does not vary with the component's age; i.e., $h(t) = \lambda$.

The above two assumptions of the SOFR model imply [30]: (1) the MTTF of the processor, $MTTF_p$, is the inverse of the total failure rate of the processor, λ_p; and (2) the failure rate of the processor is the sum of the failure rates of the individual structures due to individual failure mechanisms. Hence,

$$MTTF_p = \frac{1}{\lambda_p} = \frac{1}{\sum_{i=1}^{j}\sum_{l=1}^{k}\lambda_{il}} \tag{2.10}$$

where λ_{il} is the failure rate of the i^{th} structure due to the l^{th} failure mechanism.

2.2.2 Computing MTTF Values for Applications

The MTTF models used in RAMP (Equations 2.1, 2.4, 2.6, 2.8, 2.9) provide MTTF estimates for fixed operating parameters (in particular, fixed temperature (T), voltage (V), and frequency (f), and activity factor (p)). However, when an application runs, these parameters can all vary with time (V and f vary in processors with DVS). We assume that we can account for the impact of this variation by: (1) calculating an MTTF value (and consequently failure rate) based on instantaneous T, V, f, and p (measured over a reasonably small time granularity); and (2) using the inverse of the average over time of these failure rates to determine the actual MTTF value of every structure for every failure mechanism when running the application (this averaging over time is similar to the assumption used in the SOFR model which averages over space). For thermal cycling, we calculate the average temperature over the entire run, which is used in Equation 2.8 to determine the thermal cycling MTTF value. Sometimes we refer to this average MTTF value as the MTTF value of the application. To determine the MTTF value for a workload, we can use a weighted average of the MTTF values of the constituent applications.

2.3 RAMP 2.0

As discussed in the previous section, the SOFR model requires two simplifying assumptions to be made. Both these assumptions limit RAMP 1.0's applicability and introduce some inaccuracies. RAMP 1.0 assumes all processors are series failure systems; i.e., the first failure anywhere on chip will cause the entire processor to fail. This is not strictly accurate – some duplicated structures on chip can fail without causing the entire processor to fail. Also, RAMP 1.0 assumes all failure mechanisms have an exponential failure distribution, which implies the failure mechanisms have a constant failure rate. This is inaccurate – a typical wear-out

failure mechanism will have a low failure rate at the beginning of the component's lifetime and the value will grow as the component ages.

RAMP 2.0 addresses these inaccuracies in the SOFR model. Although there is still debate about the most accurate failure distribution for each failure mode, lognormal distributions have been shown to be applicable for a wide range of wear-out mechanisms [31, 32, 33]. Hence, RAMP 2.0 models failure mechanisms with lognormal distributions. RAMP 2.0 then uses Monte-Carlo simulation methods to calculate total processor MTTF for series-parallel failure systems. In Section 2.3.1, we describe lognormal distributions, and we explain our Monte-Carlo simulation methods in Section 2.3.2.

2.3.1 Lognormal Distributions

The lognormal distribution has been found to be a better model for failure degradation processes common to semiconductor failure mechanisms than the exponential distribution [31, 32, 33]. In most cases, this can be shown using the multiplicative degradation argument [31], briefly explained below. For a structure undergoing wear-out due to some failure mechanism, let $x_1, x_2, ...x_n$ be the amount of degradation seen at successive discrete time intervals. Let us assume that the amount of degradation seen in a time interval tends to depend on the total amount of degradation already present. This is known as multiplicative degradation [31]. In other words, the amount of degradation experienced in the n^{th} time interval, $(x_n - x_{n-1})$, will be some multiple of the total degradation already present at the end of the $(n-1)^{th}$ time interval, x_{n-1}. Hence,

$$x_n - x_{n-1} = \alpha_n x_{n-1} \Longrightarrow x_n = (1 + \alpha_n) x_{n-1} \tag{2.11}$$

where α_n is a small random value. Based on the above, we can express the total amount of degradation at the end of the n^{th} time interval, x_n, as:

$$x_n = [\prod_{i=1}^{n} (1 + \alpha_i)] x_0 \tag{2.12}$$

where x_0 is the degradation at time 0, and is a constant, and α_i are small random values. Taking the natural logarithm of both sides,

$$\ln x_n = \sum_{i=1}^{n} \ln(1 + \alpha_i) + \ln x_0 \approx \sum_{i=1}^{n} \alpha_i + \ln x_0 \tag{2.13}$$

since $\ln(1 + x) \approx x$ for small values of x. Since α_i are random, equidistant, independent values, the Central

Limit Theorem [30] implies that $\ln x_n$ has a normal distribution. Hence, x_n has a lognormal distribution for any n (or any time t). Since failure occurs when the amount of degradation reaches a critical point, time of failure will be modeled successfully by a lognormal for this type of process. The multiplicative degradation model has been shown to be a good fit for chemical reactions, diffusion of ions, and crack growth and propagation. Most semiconductor failure models are caused by one of these three degradation processes [31]. Hence, the lognormal distribution is a good fit for wear-out mechanisms.

The probability density function for the lognormal distribution is given by [34]:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{2.14}$$

μ and σ are the mean and standard deviation of the underlying norrmal distribution [34]. μ is related to the MTTF of the lognormal distribution, $MTTF$, as $MTTF = e^{\mu + \frac{\sigma^2}{2}}$ [34]. As suggested in [35], we use $\sigma = 0.5$ which has been found to model wear-out based failure mechanisms well.

2.3.2 Monte Carlo Simulation for Reliability

To obtain the lifetime distribution and MTTF for the processor as a whole, we need to combine the effects of the individual lognormal distributions across all the mechanisms and structures. Due to the complexity of the lognormal distribution, and the large cross-product of structures and mechanisms, calculating processor reliability analytically is exceedingly difficult. [2] To address this problem, we use a Monte Carlo simulation method to calculate total processor reliability. A Monte Carlo method is an algorithm that solves a problem by generating suitable random numbers and observing the fraction of the numbers that obey some property or properties. The method is useful for obtaining numerical solutions to problems that are too complicated to solve analytically [36].

Generating Lognormal Distributions

The Box-Muller transform can be used to generate a lognormal distribution from a uniform distribution [37]. As discussed previously, the mean of the underlying normal distribution, μ, is related to the MTTF of the lognormal distribution, $MTTF$, by

$$MTTF = e^{\mu + \frac{\sigma^2}{2}} \tag{2.15}$$

Hence,

$$\mu = ln(MTTF) - \frac{\sigma^2}{2} \tag{2.16}$$

[2] As explained, if the individual failure distributions were exponential, with the SOFR model, the total processor MTTF can be easily calculated as the inverse of the sum of the FIT rates individual structures and mechanisms.

Also, as described earlier, for a wear-out based failure mechanism, $\sigma = 0.5$.

If $rand1$ and $rand2$ are two independent uniformly distributed random numbers, a normally distributed random number, $rand_{normal}$, with mean 0 and standard deviation 1, is given by [37]

$$rand_{normal} = \sqrt{-2ln(rand1)} \times sin(2\pi rand2) \tag{2.17}$$

Next, the scaled normally distributed random number, $rand_{scaled-normal}$, with mean μ and σ, can be obtained from the normally distributed random number by

$$rand_{scaled-normal} = \mu + rand_{normal} \times \sigma \tag{2.18}$$

The scaled normal random number can be used to generate a random lognormal distribution, $rand_{lognormal}$, as

$$rand_{lognormal} = e^{rand_{scaled-normal}} \tag{2.19}$$

Substituting,

$$rand_{lognormal} = e^{ln(MTTF) - \frac{\sigma^2}{2} + \sigma(\sqrt{-2ln(rand1)}sin(2\pi rand2)} \tag{2.20}$$

Hence, with Equation 2.20, two random uniform variables, $rand1$ and $rand2$, can be used to generate a lognormal distribution with parameters $MTTF$ and σ.

Modeling Systems with the MIN-MAX Method

Next, we need a method to compute the MTTF of series-parallel failure systems. Unlike a series failure system where the processor will fail when its first structure fails, a series parallel system can survive structure failures when a parallel or redundant unit is available. We use a simple MIN-MAX analysis to determine the lifetime of such systems. Consider a single processor that consists of two structures, A, and B, with lifetimes, t_A, and t_B. It should be noted that t_A and t_B are not the MTTFs of A and B, but are the lifetimes of the structures for a *single* random processor. The average value of t_A and t_B across many processors would give the MTTFs of A and B.

If A and B are in series, failure would occur at $MIN(t_A, t_B)$ because the first structure to fail will cause the processor to fail. On the other hand, if A and B are in parallel, failure would occur at $MAX(t_A, t_B)$ because both structures have to fail for the processor to fail. If a structure, C, with lifetime, t_C, is added

in series to A and B in parallel, the new lifetime of the processor would be $MIN(MAX(t_A, t_B), t_C)$. This simple concept can be extended to any processor represented in a series or series-parallel fashion to obtain total MTTF.

Now, in any single iteration of the Monte-Carlo experiment, we use Equation 2.20 to generate a random lifetime for each failure mechanism and structure on chip. A MIN-MAX analysis of these lifetimes based on the processor's layout would give the lifetime of the entire processor for that iteration. The MTTF of the processor can now be calculated by repeating this process over many iterations and averaging the processor lifetimes obtained. As in any other Monte-Carlo experiment, the accuracy of the analysis increases with the number of iterations performed.

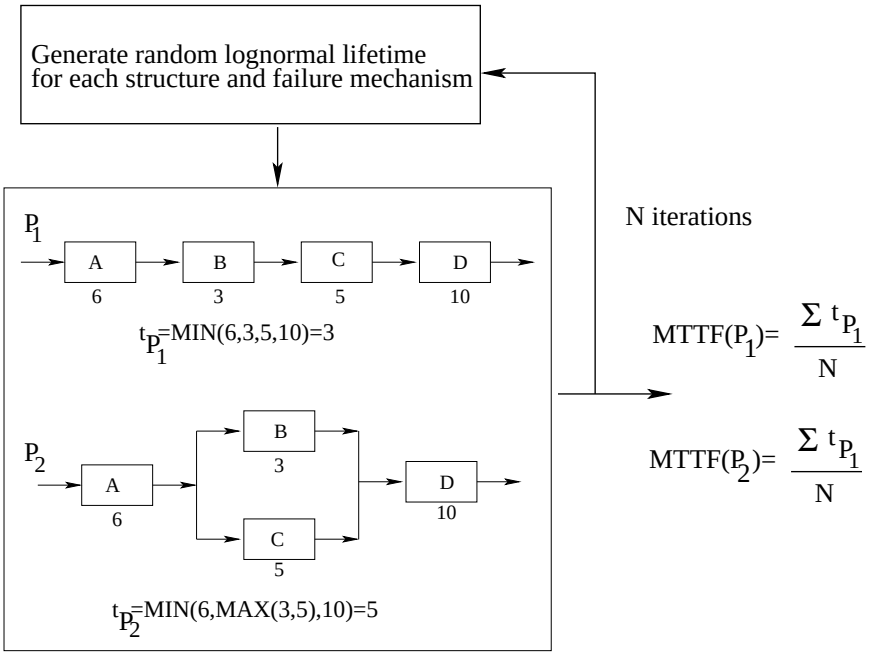


Figure 2.1: Monte Carlo simulation of MTTF of two systems, P_1 and P_2. The MIN-MAX method to determine system lifetime is illustrated for sample lifetime values for both systems. The higher the number of iterations, N, the higher the accuracy of the final MTTFs.

Figure 2.1 illustrates this method. Consider two systems, P_1 and P_2. Both systems have four structures, A, B, C, and D. As can be seen in Figure 2.1, P_1 is a series failure system while P_2 is a series-parallel failure system. For any single iteration of the Monte-Carlo algorithm, the lifetime of P_1, $t_{P_1} = MIN(t_A, t_B, t_C, t_D)$, while the lifetime of P_2 is $t_{P_2} = MIN(t_A, MAX(t_B, t_C), t_D)$, where t_A, t_B, t_C, and t_D are the random lifetimes of each structure. If N iterations are performed, the MTTF of system P_1 is $MTTF_{P_1} = \sum \frac{t_{P_1}}{N}$, and the MTTF of system P_2 is $MTTF_{P_2} = \sum \frac{t_{P_2}}{N}$. In addition, we also have the distribution of failure times for the N systems. In our experiments, we use a value of $N = 10^7$.

In order to compute application MTTFs, RAMP 2.0 uses time averaging like RAMP 1.0 (described in Section 2.2.2).

2.4 RAMP 1.0 vs RAMP 2.0

As explained earlier, RAMP 1.0 and RAMP 2.0 differ in their utility and accuracy. The relative simplicity of Equation 2.10 facilitates the use of RAMP 1.0 both as a simulation tool and in real hardware. As a simulation tool, RAMP 1.0 should be used in conjunction with a timing simulator to determine workload behavior, and a power and thermal simulator for power and temperature profiles. In real hardware, RAMP 1.0 would require sensors and counters that provide information on processor operating conditions.

Unlike RAMP 1.0, it is difficult to use RAMP 2.0 in real hardware in its current form, and instead, it can only be used as a simulation tool. This is due to the difficulty of implementing the Monte-Carlo methods in real hardware.

Due to its simplicity, the SOFR model used in RAMP 1.0 is commonly used in industry to combine failure rates. However, assuming exponential distributions for the failure mechanisms reduces the accuracy of RAMP 1.0. In Sections 3.4.4 and 6.3.7, we quantify this inaccuracy by comparing our experimental results on RAMP 1.0 and RAMP 2.0.

2.5 The Reliability Qualification Process

Based on the desired reliability for a product, an MTTF value, $MTTF_{target}$, is targeted during the process of reliability qualification. CMOS devices like microprocessors are commonly designed to have an MTTF in the range of 30 years [8]. [3]

Inherent in attaining this target MTTF value is a cost-performance tradeoff. The cost-performance tradeoff that is made determines the proportionality constants in the individual failure model equations. These constants are technology dependent and also depend on factors like design materials used and yield. High values for the proportionality constants imply more reliable processors, but higher cost.

For a specific set of proportionality constants, RAMP 1.0 and RAMP 2.0 can provide an absolute MTTF value for a given application. However, since we do not have the function that relates these constants to cost, we do the following: according to current worst-case based reliability qualification methodologies, the architectural parameters in the model equations (temperature (T), voltage (V), frequency (f), and activity

[3] Processor MTTFs tend to be much higher than the expected consumer use lifetime of the product. These values allow the product's consumer service life to fall far out in the tails of the lifetime distribution curve [8].

factor (p)) are worst-case operating parameters. For a given technology, the target MTTF value and the cost we are willing to pay for reliability qualification (i.e., the proportionality constants determined by materials, yield, etc.) determine these worst-case parameters. We therefore use these architecture level parameters as a proxy for the cost of reliability qualification. We call these proxy parameters as T_{qual}, V_{qual}, f_{qual}, and p_{qual} respectively. The higher the value of these parameters, the more expensive the reliability qualification. We then feed these parameters to RAMP 1.0 and RAMP 2.0 to determine the proportionality constants that will give us the target MTTF value for each failure mechanism for each structure. We use those constants to determine an absolute MTTF value according to the actual T, V, f, and p seen by the workload.

In our experiments, to bound the space explored, we only vary T_{qual} to represent different costs. We maintain a fixed value for f_{qual} and V_{qual} depending on the processor being simulated. We fixed p_{qual} to be the highest activity factor obtained across our application suite from our timing simulator. As described, the above methodology also requires setting a target MTTF value for each structure for each failure mechanism. We assumed that at qualification, the probability of failure is equal across all failure mechanisms and is proportional to the area of a structure.

We require the use of T_{qual}, V_{qual}, f_{qual}, and p_{qual} only when we need to calculate absolute values of MTTF (as in Section 5.5). In situations where we only compare MTTF ratios and not absolute values, we do not need to use the qualification proxies.

2.6 Summary

In this chapter, we present a new modeling methodology, called RAMP, for enabling power, performance and reliability tradeoffs. RAMP deals primarily with the the impact of temperature on wearout driven (un)reliability. There are, of course, other lifetime reliability degradations that we do not currently consider in RAMP. One example is the effect of inductive noise on the voltage rails (Ldi/dt) caused by current surges in various units.

We propose two implementations of RAMP, RAMP 1.0 and RAMP 2.0, which differ in the method used to combine MTTFs across different structures and failure mechanisms. RAMP 1.0 uses a simpler method which facilitates its use both in real hardware and as a simulation tool. RAMP 2.0 uses a more complex Monte-Carlo method to combine MTTFs. This increases its accuracy but makes its use as a dynamic tool in real hardware difficult.

Chapter 3

The Impact of Technology Scaling on Lifetime Reliability

Advances in CMOS semiconductor technology have been steadily improving processor performance. These advances have been driven by aggressive scaling of device feature sizes. However, CMOS scaling is accelerating the onset of problems due to long-term processor hardware failures or lifetime reliability.

In this chapter, we quantify the impact of technology scaling on the failure mechanisms discussed in Chapter 2. In particular, our evaluation and analysis attempt to model the scaling effects of taking a 180nm POWER4-like processor, and gradually scaling that chip from 180nm to 65nm, without any substantial modifications to the architectural pipeline. To facilitate this analysis, we enhance the RAMP models by adding scaling specific parameters to enable lifetime reliability evaluation at different technologies. Section 3.1 gives an overview of scaling theory and practice. The scaling models for the individual failure mechanisms are described in Section 3.2. Our scaling analysis is described in Section 3.3, and we discuss the results in Section 3.4.

3.1 Scaling Theory and Practice

Device scaling results in the reduction of feature sizes and voltage levels of transistors. Application of ideal scaling theory results in three main benefits in going from one generation to the next [9]: (a) reduction of gate delay by about 30%, resulting in an increase in operating frequency by about 43%; (b) doubling of transistor density; and (c) reduction of dynamic power per transistor by about 50% (this assumes constant electric field scaling, where the supply voltage scales down by 30% in each generation). Combining the beneficial effects of (b) and (c) implies that for the same die size, under ideal scaling considerations alone, the net chip dynamic power and power density would remain unchanged with scaling.

However, in practice, processors do not scale ideally. With real scaling in the deep sub-micron range, processor power density, and consequently temperature, have been increasing at an alarming rate, which directly affects processor lifetime reliability. The main reasons behind this increase are:

- **Supply voltages are not scaling ideally:** This prevents the dynamic power per transistor from

decreasing at the ideal rate. One of the reasons behind the slowing down of supply voltage scaling is the attempt to retain competitive frequency growth by tuning up the voltage to the maximum levels allowed in a given technology generation. Also, as the gap between the threshold voltage and the supply voltage diminishes to less than a volt, basic noise immunity issues (in logic) and cell state stability issues (in SRAM macros) makes it ever harder to scale down the supply voltage. Hence, although processor area scales down ideally, power does not, resulting in higher power density and consequently higher temperatures.

- **Total chip leakage power is increasing:** Scaling down device threshold voltages (ideally) by about 15% per generation causes sub-threshold leakage current per transistor to increase by 5 times. Since the total transistor count on the die increases by about 50% per generation, the total chip leakage power increases about 7.5 times. This increase is further compounded by the exponential dependence of leakage power on temperature.

3.1.1 Impact of Non-Ideal Scaling

The above non-ideal scaling coupled with the reduced feature sizes affects processor lifetime reliability in the following ways. First, all of the five failure mechanisms in RAMP are adversely affected by increases in temperature, with some of the mechanisms exhibiting an exponential or larger dependence on temperature. Second, the dielectric thickness of devices is fast decreasing to the point where it is approaching a few angstroms. This, coupled with the fact that there has been a general slowdown in supply voltage scaling is expected to increase the intrinsic failure rate due to gate oxide breakdown (TDDB). Third, the decreasing feature size of interconnects accelerates electromigration failure rates.

The detrimental impact of scaling on intrinsic reliability in general, and gate oxide reliability in particular, has been studied extensively [27, 29, 5]. However, most of these studies have been performed at the device level, and consider individual failure mechanisms in isolation. Additionally, they are performed at fixed worst case operating points without any knowledge of the target application suite of the processor. However, since the power consumed by the processor varies with the executing workload, the actual operating temperature and interconnect current densities also depend on the workload. Consequently, the failure rate of a component (or the processor as a whole) depends on the target workload. Thus, an application oblivious analysis of processor reliability would produce unrepresentative reliability data.

3.2 Impact of Scaling on Failure Mechanisms

3.2.1 Electromigration

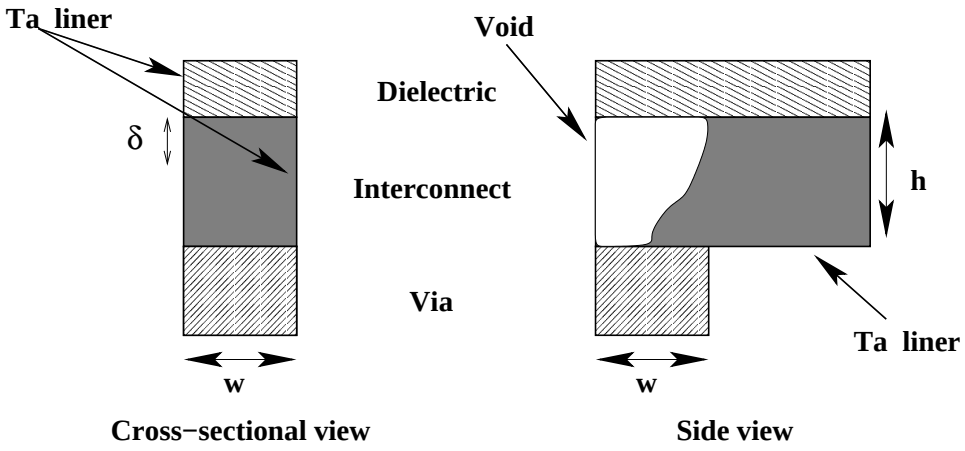


Figure 3.1: EM in copper interconnects

The detrimental impact of increasing temperatures on electromigration due to scaling is already modeled in RAMP. However, scaling also reduces interconnect dimensions which has a negative impact on electromigration.

For years, copper doped aluminum had been the semiconductor industry's interconnect metal of choice because of its ease of integration into the manufacturing process, low resistivity, and cheap availability. In the past few years, reliability problems due to electromigration and a need for even lower resisitivities prompted the industry to consider using only copper for interconnects. Copper has lower resistivity (and hence lower interconnect delay) than copper-doped aluminum and is much more resilient to electromigration [27]. However, there are problems with using copper - in particular, copper diffuses readily into silicon causing deep level defects. This problem is solved by adding a lining layer using tantalum (Ta) which separates the copper interconnects from the surrounding devices [38]. Copper interconnects have now moved from development to manufacturing and high-level server processors already use copper interconnects [39]. Some commodity processors have also started using all copper interconnects and the rest are expected to start using copper interconnects in the near future. The impact of scaling on electromigration reliability is different for copper and aluminum interconnects. This is because of the difference in the electromigration mechanism between copper and aluminum [27, 40]. Since it is expected that copper will be the dominant interconnect metal in the future, we only model the impact of scaling on copper interconnects.

Copper interconnects are typically fabricated using a damascene processing method. In these structures,

the top surface of the copper damascene line is covered with a dielectric film, while the bottom surface and two sidewalls are sealed with a tantalum (Ta) liner [27]. The tantalum liner prevents electromigration along the surfaces it covers. However, the top surface of the line cannot be covered with tantalum due to manufacturing constraints. As a result, electromigration in copper is dominant at the top interface layer between the interconnect and the dielectric [27]. This is illustrated in Figure 3.1.

If the effective thickness of the interface layer is δ, and the interconnect width is w, then the electromigration flux is constrained to an area δw. If the height of the interconnect is h, then the interconnect current flows through an area wh. The relative amount of atomic flux flowing through the interface region is proportional to the interface area to interconnect area ratio, $\frac{\delta w}{wh} = \frac{\delta}{h}$ [27].

Electromigration voids are found to occur most commonly at the interface between the interconnects and the metal vias [27]. Electromigration failure is considered to have occurred when the void formed grows larger than the width of the via, w (which is the same as the interconnect width). Hence, mean time to failure due to electromigration, $MTTF_{EM}$, is proportional to the width of the via, w, and inversely proportional to the relative amount of flux passing through the interface region, $\frac{\delta}{h}$ [27]. Thus, when a scaling factor of κ is applied, electromigration lifetime reduces by κ^2 due to w and h (both w and h scale by κ while δ remains constant).

Additionally, the value of J for a structure (in Equation 2.1) is equal to the product of the activity factor of the structure, p, and the maximum allowed interconnect current density for that technology generation. This maximum allowed current density changes with scaling.

3.2.2 Stress Migration

The main impact of scaling on stress migration is the dependence on temperature, which is already modeled in Equation 2.4. Temperature affects stress migration failure rate in two ways: there is an exponential dependence on temperature which is detrimental to reliability, and there is the $|T - T_0|^{-m}$ term from Equation 2.4 which has a positive effect on reliability. However, the exponential term usually overshadows the other term, resulting in a decrease in reliability with temperature. Scaling has no other direct impact on stress migration.

There are indirect scaling effects on stress migration due to the use of new low-k dielectrics which tend to be porous and brittle [41]. However, since our experiments assume that our scaled processors all use the same type of interconnect metal and dielectric material, we do not model these effects.

There are some indirect effects due to scaling due to the materials used in the inter-layer dielectrics (ILD). Scaling requires the increased use of new low-k dielectrics for the ILD layers. These materials tend

to be porous and brittle and some are known to have reliability problems [41]. The thermal expansion rates of these new low-k dielectrics also tend to be significantly different from the interconnect thermal expansion rate [41], creating stress migration problems. In this dissertation, we do not model these effects caused by different interconnect and dielectric materials.

3.2.3 Time-Dependent Dielectric Breakdown

Scaling has a profound effect on gate oxide reliability. Effects of scaling on TDDB already modeled in RAMP in Equation 2.6 are the detrimental effect of increasing temperatures and the beneficial effect of decreasing supply voltage. Gate oxide reliability depends on other scaling parameters as described below.

First, decreasing gate oxide thickness with scaling decreases reliability, due to increasing gate leakage and tunneling current, I_{leak}. The MTTF due to gate oxide breakdown is directly proportional to the value of I_{leak}. I_{leak} increases by one order of magnitude for every 0.22nm reduction in gate oxide thickness [29]. As a result, if gate oxide thickness reduces by Δt_{ox} with scaling, then $MTTF_{TDDB}$ reduces by $10^{\frac{\Delta t_{ox}}{0.22}}$, where the reduction in gate oxide thickness, Δt_{ox}, is expressed in nanometers.

Second, for current and future range gate oxide thicknesses, $MTTF_{TDDB}$ is inversely proportional to the total gate oxide surface area [5]. Third, decreasing supply voltage increases reliability [42].

Combining the scaling effect of voltage, gate oxide thickness, area, and temperature, if we scale down from process 1 to process 2, which have supply voltages, V_1 and V_2, gate oxide thicknesses, t_{ox1} and t_{ox2}, total gate oxide areas, A_1 and A_2, at temperatures, T_1 and T_2, the ratio of mean time to failures, $MTTF_1$ and $MTTF_2$ is given by:

$$\frac{MTTF_1}{MTTF_2} = 10^{\frac{(t_{ox1}-t_{ox2})}{0.22}} \times \frac{V_2^{(a-bT_2)}}{V_1^{(a-bT_1)}} \times \frac{A_1}{A_2} \times \frac{e^{\frac{(X+\frac{Y}{T_1}+ZT_1)}{kT_1}}}{e^{\frac{(X+\frac{Y}{T_2}+ZT_2)}{kT_2}}} \tag{3.1}$$

where X, Y, Z, a and b are empirically determined constants, described in Section 2.1.3.

3.2.4 Thermal Cycling

Like stress migration, the main impact of scaling on thermal cycling modeled in RAMP is the impact of temperature. Scaling has no other direct impact on thermal cycling. There are indirect scaling effects on thermal cycling due to the use of new low-k dielectrics which have inferior adhesive properties [41], increasing susceptibility to thermal cycling failure. However, since our experiments assume that our scaled processors all use the same type of interconnect metal and dielectric material, we do not model these effects.

Failure Mech.	Major temperature dependence	Voltage dependence	Feature size dependence
EM	$e^{\frac{E_{aEM}}{kT}}$		wh
SM	$\|T - T_0\|^{-m} e^{\frac{E_{aSM}}{kT}}$		
TDDB	$e^{\frac{(X + \frac{Y}{T} + ZT)}{kT}}$	$(\frac{1}{V})^{(a-bT)}$	$10^{\frac{\Delta t_{ox}}{0.22}}$
TC	$\frac{1}{T^q}$		
NBTI	$[(ln(\frac{A}{1+2e^{\frac{B}{kT}}}) - ln(\frac{A}{1+2e^{\frac{B}{kT}}} - C)) \times \frac{T}{e^{\frac{-D}{kT}}}]^{\frac{1}{\beta}}$	Not modeled	Not modeled

Table 3.1: Summary of impact of scaling on MTTF.

3.2.5 Negative Bias Temperature Instability (NBTI)

NBTI has a strong positive temperature and field dependence. As a result, the higher temperatures seen on chip due to scaling exacerbate this problem. As can be seen in Equation 2.9, RAMP already models the effect of temperature on NBTI.

Thinning of the gate oxide due to scaling also increases NBTI reliability concerns [19] due to an increase in the electric field across the gate. However, we do not currently have accurate models for the impact of electric field changes on NBTI MTTF. As a result, we do not factor this effect in our scaling analysis. As discussed in Section 4.2.2, this is am important area of future work.

3.2.6 Summary of Impact of Different Parameters

Table 3.1 summarizes the impact of different scaling related parameters on the intrinsic failure mechanisms. It shows that temperature has an exponential detrimental impact on EM, SM, and NBTI (despite the $|T - T_0|$ in SM), a more than exponential impact on TDDB, and a less than exponential impact on TC. Electromigration is also detrimentally impacted by smaller values of w and h, and TDDB is adversely affected by reducing t_{ox}. Finally, a positive effect of scaling is observed in TDDB due to lower supply voltages. Note that lower voltages also help with temperature, but not enough because of increasing power density. Also, we do not currently model the effect of voltage and feature size (which both dictate electric field) on NBTI.

3.3 Scaling Experimental Methodology

In this section, we describe the experimental methodology used in our scaling analysis.

Technology Parameters	
Process technology	180 nm
V_{dd}	1.3 V
Processor frequency	1.1 GHz
Processor core size (not including L2 cache)	$81mm^2$ ($9mm$ x $9\ mm$)
Leakage power density at $383K$	0.04 W/mm^2
Base Processor Parameters	
Fetch/finish rate	8 per cycle
Retirement rate	1 dispatch-group (=5, max) per cycle
Functional units	2 Int, 2 FP, 2 Load-Store (Agen), 1 Branch, 1 LCR
Integer FU latencies	1/7/35 add/multiply/divide (pipelined)
FP FU latencies	4 default, 12 div. (pipelined)
Reorder Buffer size	150
Register file size	120 integer, 96 FP
Memory queue size	32 entries
Base Memory Hierarchy Parameters	
L1 (Data)	32KB
L1 (Instr)	32KB
L2 (Unified)	2MB

Table 3.2: Base 180nm POWER4-like processor used in scaling analysis

3.3.1 Architecture Modeled and Performance Simulation Methodology

The base processor simulated is a 180nm out-of-order 8-way superscalar processor, conceptually similar to a single core 180nm POWER4-like processor [43]. Table 3.2 summarizes the base 180nm processor modeled. Note that some of the microarchitectural parameters like cache sizes, assumed in our base model, are different from real values in the POWER4 processor. Also, although we model the performance impact of the L2 cache, we do not model its reliability. This is because the temperature of the L2 cache is much lower than the processor core [43], resulting in very low L2 intrinsic failure rates. Hence, we concentrate on intrinsic failures in the core.

The processor is modeled using a trace-driven research simulator called Turandot [44], developed at IBM T.J. Watson Research Center. The modeled microarchitecture is conceptually similar to a single core 180nm POWER4-like processor [43]. As described in [44], Turandot was calibrated against a pre-RTL, detailed, latch-accurate processor model. Despite the trace-driven nature of Turandot, the extensive validation methodology provides high confidence in the results.

3.3.2 Power Simulation Methodology

To estimate power dissipation, we use the PowerTimer toolset developed at IBM T.J. Watson Research Center [45]. This toolset, in its default form, is built around the Turandot cycle-accurate performance simulator referred to in the previous section. The power models that are built into the Turandot-based

PowerTimer are based on circuit accurate power estimations from the 180nm POWER4 processor [43]. The power analysis has been performed at the macro level using CPAM, a circuit-level power analysis tool [46]. Multiple macros are combined to form microarchitectural structures. PowerTimer combines the circuit accurate power estimates from over 400 macros into 60 primary microarchitectural structures. PowerTimer uses microarchitectural activity information obtained from the performance simulator, Turandot, to provide per-cycle power estimates. For our simulations, we use realistic clock gating assumptions in PowerTimer, in tune with actual data available from current generation (post-POWER4) microprocessors.

Leakage Power

Leakage power is calculated based on modeled structure areas. For the base 180nm process modeled, a leakage power density of 0.04 W/mm^2 at 383K is used. This value is based on simulation-based estimates for processors like the POWER4 , and assumes standard leakage power control techniques like the use of high-threshold devices in non-critical logic paths and arrays.

We also model the impact of temperature on leakage power using the technique in [47]. At a temperature T, the leakage power, $P_{leakage(T)}$, is given by:

$$P_{leakage(T)} = P_{leakage(383K)} \times e^{\beta(T-383)} \tag{3.2}$$

where β is a curve fitting constant. The value of β we use (0.017) is taken from [47].

3.3.3 Temperature Simulation Methodology

We use the HotSpot tool [11] to derive temperature estimates from power. The chip floorplan fed to HotSpot resembles a single core of a 180nm POWER4-like processor, of size 81mm^2 (9mm x 9 mm). The microarchitectural structures modeled using Turandot and PowerTimer are combined into 7 structures. Based on each structure's area, HotSpot calculates thermal resistance and capacitance values. These thermal resistances and capacitances are combined into an RC network. HotSpot dynamically solves this RC network to produce temperature measurements at the granularity of 1μ second (using power information from PowerTimer).

Heat Sink Temperatures

As explained in [11], the RC time constant of the processor heat sink is significantly larger than the RC time constant of individual structures. Since the RC time constant of the heat sink is so large, there is not enough time for there to be significant changes in heat sink temperature during simulation runs. Hence, it is crucial that HotSpot be initialized with accurate heat sink temperatures.

As a result, all simulations are run twice - the first run is used to obtain average power consumption values for every structure on chip. These average power values are then used to calculate the initialization temperature of the heat sink. Once the heat sink is initialized, the second run produces accurate temperature results.

For the heat sink thermal resistance at 180nm, we use 0.8 W/K [11]. For the sake of comparing technology generations, we maintain a constant heat sink temperature for each application with scaling (different applications have different heat sink temperatures, which remain constant with scaling). In order to simulate this, we vary the thermal resistance of the heat sink with technology generation in HotSpot. It should be noted that this could result in potentially conservative failure rate estimates for advanced technologies. In an actual scaling scenario, maintaining constant heat sink temperature might not be feasible from a heat sink cost perspective, resulting in higher heat sink temperatures (and resultant higher peak temperatures) for advanced technology generations.

3.3.4 Reliability Calculation

For our reliability calculations, we use the more accurate RAMP 2.0 model. Based on temperature estimates obtained from HotSpot and power estimates obtained from PowerTimer, RAMP 2.0 calculates MTTF values, for every structure on chip due to each failure mechanism. With these structure and failure mechanism MTTFs, we use RAMP 2.0 to estimate total processor MTTF for different applications at different technology generations. The average MTTF across all applications for the 180nm base processor is normalized to 1.0. All other MTTFs are expressed with respect to the average 180nm base processor MTTF. As discussed in Section 2.5, since we only compare MTTF ratios in our scaling analysis, we do not require the use of T_{qual} and other qualification proxies.

In order to quantify the difference between using lognormal and exponential distributions for failure mechanisms, we repeat some of our scaling experiments with RAMP 2.0. These results are described in Section 3.4.4.

3.3.5 Workload Description

Our experimental results are based on an evaluation of SPEC2K benchmarks. We report experimental results based on PowerPC traces of 16 SPEC2K benchmarks (8 SpecInt + 8 SpecFP). The SPEC2K trace repository used in this study was generated using the Aria trace facility in the MET toolkit [48], and was generated using the full reference input set. Sampling was used to limit the trace length to 100 million instructions per program. The sampled traces have been validated with the original full traces for accuracy

Type	Application	IPC	180nm Power (W)	Max Temp (K)
Spec2K	ammp	1.06	26.08	334.5
Float	applu	1.17	26.94	336.4
	sixtrack	1.38	27.32	336.0
	mgrid	1.71	27.78	338.1
	mesa	1.75	29.21	339.3
	facerec	1.79	29.60	339.4
	wupwise	1.66	30.50	341.2
	apsi	1.64	30.65	341.0
SpecFP average		**1.52**	**28.51**	
Spec2K	vpr	1.38	26.93	335.5
Int	bzip2	2.31	27.71	336.6
	twolf	1.26	28.44	337.5
	gzip	1.85	28.69	337.1
	perlbmk	2.25	30.59	340.7
	gap	1.76	31.24	341.2
	gcc	1.24	31.73	340.9
	crafty	2.25	31.95	342.4
SpecInt average		**1.79**	**29.66**	

Table 3.3: Average IPC, power consumption, and the maximum temperature of the hottest structure on the 180nm base processor for our scaling workload, ordered by increasing power.

and correct representation [49].

Table 3.3 summarizes the benchmarks studied, including the IPC, average power consumption, and maximum temperature reached on the 180nm base processor. The power values include leakage power consumption. As can be seen, for our processor, SpecInt has a higher average IPC and marginally higher power consumption than SpecFP.

3.3.6 Scaling Methodology

We study the failure rate for our POWER4-like processor for five technology generations, ranging from $180nm$ to $65nm$. The scaling parameters used are listed in Table 3.4. All scaling is done with respect to $180nm$, as the performance and power simulator are calibrated for this technology point. A scaling factor of 0.7 is assumed from $180nm$ to $90nm$. For $90nm$ to $65nm$, a scaling factor of 0.8 is used, based on the assumption that a scaling factor of 0.7 will be difficult to maintain in technology generations after 90nm. Next, we discuss each column in Table 3.4.

Voltage and Frequency Scaling

With an ideal scaling factor of 0.7, [9] states the best-case device frequency scaling per generation would be about 43%. However, in the most recent era of technology remaps, ideal frequency scaling is constrained,

Technology Generation (nm)	V_{dd} **V**	**Frequency GHz**	**Relative Capacitance**	**Relative Area**	t_{ox} $\AA$
180	1.3	1.1	1.0	1.0	25
130	1.1	1.35	0.7	0.5	17
90	1.0	1.65	0.49	0.25	12
65 (0.9V)	0.9	2.0	0.4	0.16	9
65 (1.0V)	1.0	2.0	0.4	0.16	9
Technology Generation (nm)	**Current Density** $\frac{mA}{m^2}$	**Leakage Power Density** $\frac{W}{mm^2}$	**Total Power (Dyn+Leak)** (W)	**Relative Total Power Density**	
180	9.0	0.040	29.1	1.0	
130	6.0	0.10	19.0	1.31	
90	4.0	0.25	14.7	2.02	
65 (0.9V)	4.0	0.54	14.4	3.09	
65 (1.0V)	4.0	0.60	16.9	3.63	

Table 3.4: Scaling parameters used for different technology generations in our scaling analysis

primarily due to tighter design rules caused by increased wiring pressures. Also, in doing progressive scaling of the same microarchitecture over multiple technology generations, it is hard to achieve ideal frequency boosts without significant investment in re-tuning all the circuit delay paths in the machine. Hence, we assume conservative 22% frequency scaling per generation. (If we assume 43% scaling, power and temperature figures will be higher resulting in even higher failure rates.)

With ideal scaling, the supply voltage should scale by a factor of 0.7 every generation. However, since threshold voltage scaling is limited to a few mV per technology generation, as the supply voltage becomes smaller, the reduction obtained by scaling slows down, such that the device overdrive (supply voltage minus threshold voltage) does not drop by more than a factor of 0.7. Additionally, due to the increase in leakage power with threshold voltage scaling, further scaling limits are imposed on threshold voltage and supply voltage. The supply voltage values in Table 3.4 are carefully chosen to match up with the scaled frequencies, while adhering to threshold voltage scaling that would reflect the leakage power density assumptions shown.

Also, we simulate two 65nm processors. One processor assumes that the voltage scales down from 90nm to 65nm to a value of 0.9 V. However, as the supply voltage approaches the threshold voltage, scaling voltage appropriately is becoming increasingly difficult. Basic noise immunity issues (in logic) and cell state stability issues (in SRAM macros) make it difficult to operate reliably at voltages below 1.0 V. As a result, we also simulate a 65nm processor which runs at 1.0 V, which we believe is more realistic (no voltage scaling takes place between 90nm and 65nm). The two different technology points are represented as 65nm (0.9V) and 65nm (1.0V) in our results.

Capacitance Scaling

The capacitance value for each technology generation is proportional to the scaling factor used for that generation. The 180nm processor is assumed to have a relative capacitance value of 1.0.

Area Scaling

The area of the processor for each technology generation is proportional to the square of the scaling factor used for that generation. The 180nm processor is assumed to have a relative area of 1.0.

t_{ox} Scaling

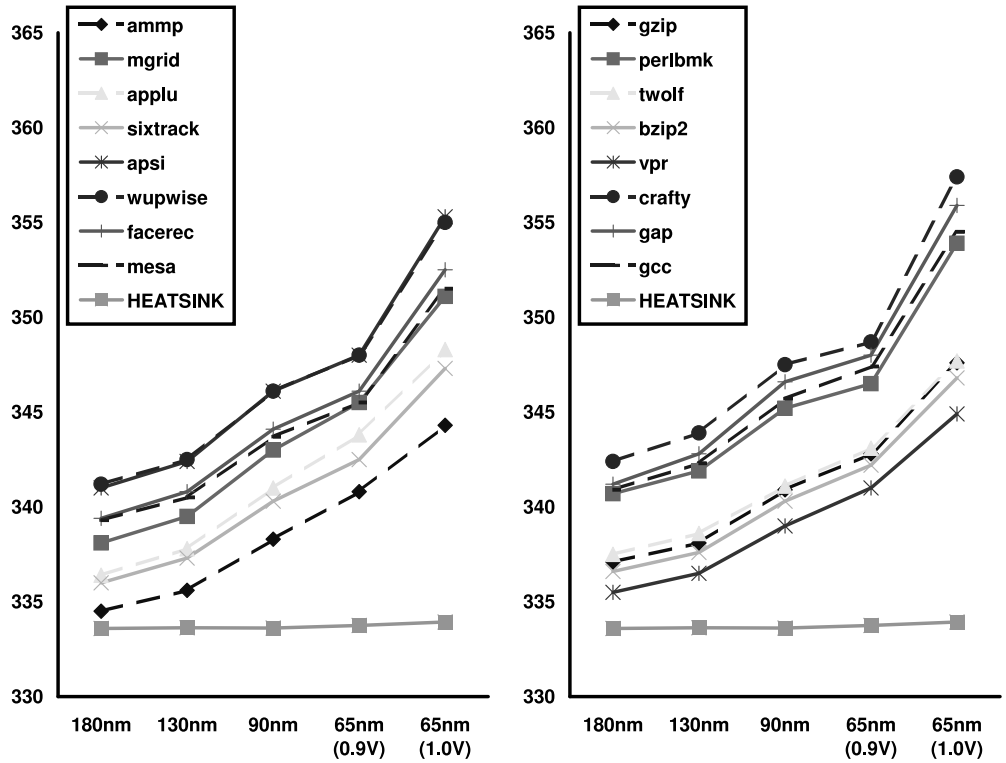
The values of t_{ox} used were obtained from the high performance logic parameters in the ITRS roadmap [1]. As can be seen, changes in t_{ox} are proportional to the scaling factor.

Interconnect Current Density

In order to compensate for the decrease in electromigration reliability with scaling, designers have been reducing interconnect current density every technology generation. We assume a 33% reduction in interconnect current density every technology generation [9]. However, this implies less and less current for devices, and limits are being reached. It is expected that interconnect current density can not be reduced beyond the 90nm technology point. Hence the values for 90nm and 65nm are the same.

Power Scaling

The leakage power densities used for each technology point assume aggressive leakage control techniques are used [50]. The total power consumption (dynamic + leakage power), averaged across all applications, is also given (based on simulations). Finally, the relative total power density (which is the ratio of the total power consumption and area), averaged across all applications, is also given. As can be seen from Table 3.4, up to 90nm, scaling reduces the total power consumption of the core. Beyond 90nm, the increase in leakage power negates the benefits from reduced dynamic power, resulting in a net increase in total power. Also, it can be seen that the average power density goes up steadily with scaling (because voltage is not scaling down ideally and leakage power is going up).



(a) SpecFP Temperatures (b) SpecInt Temperatures

Figure 3.2: Maximum temperature at the hottest structure in Kelvin (shown on the vertical-axis) reached for each application for the different technology points (horizontal-axis) in our scaling analysis. The heat sink temperature averaged over all the applications is also shown.

3.4 Scaling Results

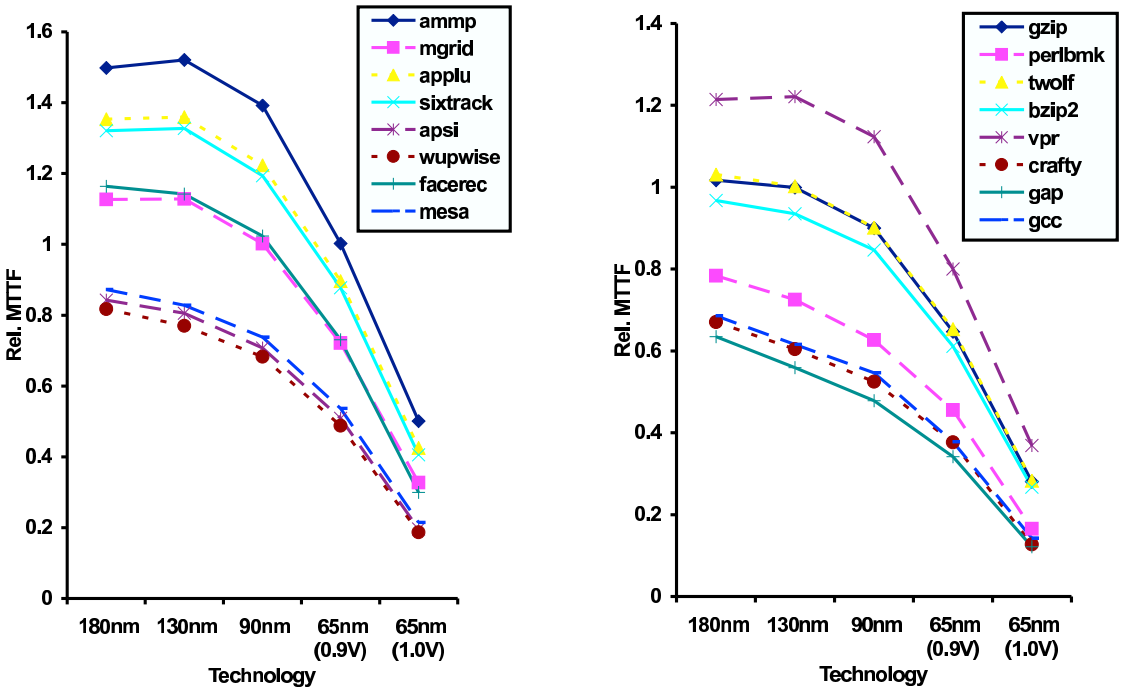
3.4.1 Temperature Analysis

We start by presenting results for temperature since they affect reliability so significantly. Figure 3.2 shows the maximum temperature reached by any structure on chip for each application for each technology generation. Also shown is the heat sink temperature, averaged over all applications (recall that we adjust the heat sink thermal resistance such that this temperature remains constant with scaling). As can be seen, while the heat sink temperature remains nearly constant with scaling, the temperature of the hottest structure increases. On average, from 180nm to 65nm (1.0V), the temperature of the hottest structure on chip increased by 15 degrees Kelvin. Application temperatures increase because the power density on chip (as seen in Table 3.4) is increasing with scaling.

The results also show that there is a significant range in temperatures across applications. There is high correlation between application power and temperature and some correlation with IPC. The hottest

applications (wupwise and apsi for SpecFP and crafty for SpecInt) in Figure 3.2 also have the highest power consumptions (and high IPCs) in Table 3.3. The same holds for the coolest applications in our suite (ammp for SpecFP and vpr for SpecInt).

3.4.2 Total MTTF Scaling

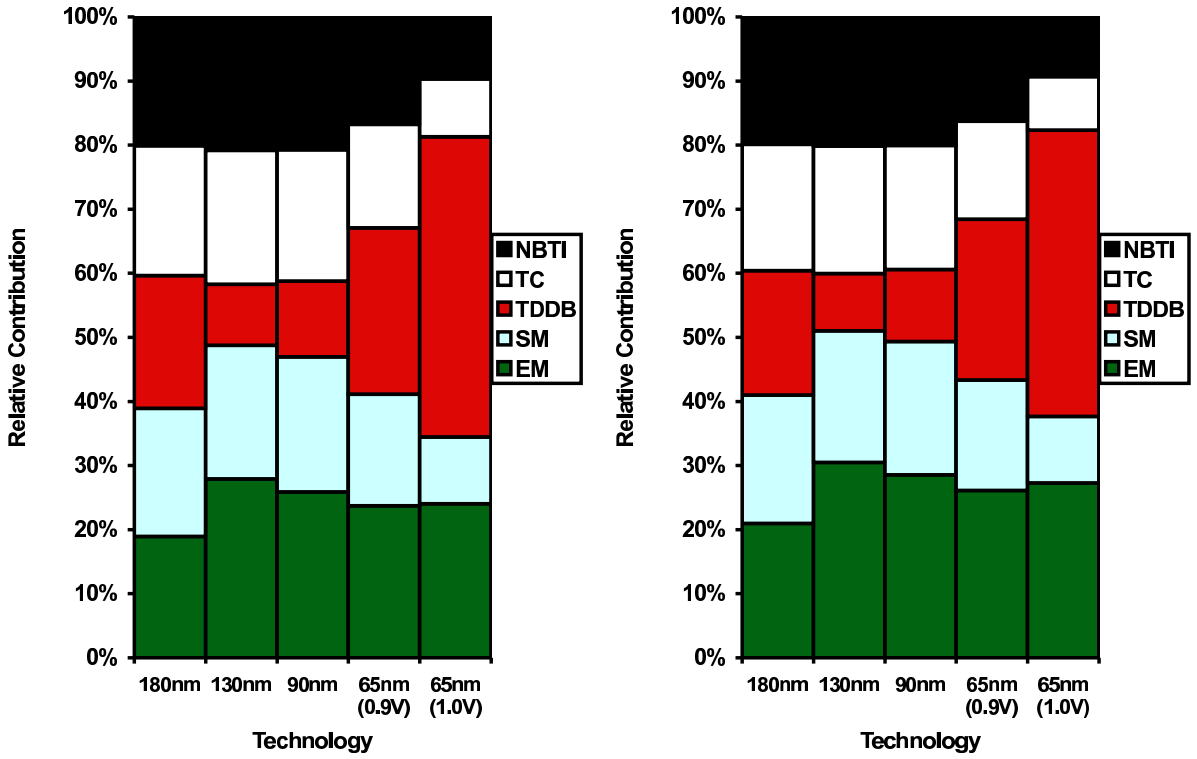


(a) SpecFP MTTFs

(b) SpecInt MTTFs

Figure 3.3: Normalized processor MTTF for each application for each technology point in our scaling analysis. The average MTTF across all applications is normalized to 1.0 at 180nm.

Figures 3.3 and 3.4 present the data for this section. Figures 3.3 (a) and (b) show the scaling behavior of the normalized processor MTTF for each application, for SpecFP and SpecInt respectively. The average MTTF across all applications at 180nm is normalized to 1.0. Figures 3.4 (a) and (b) show the relative probability of processor failure due to each each failure mechanism at different technology generations averaged across SpecFP and SpecInt respectively. At 180nm, the probability of failure due to each failure mechanism is equal (20%). These results will be discussed in more detail in Section 3.4.3.



(a) SpecFP (b) SpecInt

Figure 3.4: Relative probability of processor failure due to each failure mechanism at different technology generations in our scaling analysis. At 180nm, the probability of failure due to each mechanism is equal (20%).

Decrease in MTTF

As can be seen, there is a marked fall in the normalized MTTF with technology scaling. On average, the normalized MTTF of the SpecFP applications dropped by 63% from 180nm to 65nm (1.0V). The decrease seen in SpecInt was larger at 71%. Also, at each scaled technology point, the average normalized MTTF of SpecInt applications was lower than SpecFP applications. This is because of the higher power consumptions seen in the integer applications. There is a significant difference in MTTF from 65nm (0.9V) to 65nm (1.0V). As discussed in Section 3.3.6, many architectural structures can potentially not operate reliably at voltages lower than 1.0V. However, as can be seen, maintaining a constant voltage from 90nm to 65nm leads to a large drop in MTTFs. On the other hand, if the voltage does scale down from 90nm to 65nm, the decrease in normalized MTTF seen from 180nm to 65nm (0.9V) is brought down to (a still significant) 45% for SpecFP and 51% for SpecInt.

Workload Dependence of MTTF

In Figure 3.3, when considering the workload dependence on the normalized MTTF, there are two points of note.

First, Figure 3.3 shows that there is a large range in MTTF across applications. At 180nm, the range in MTTF across all applications is 74% of the average MTTF (which is normalized to 1.0 at 180nm). At 65nm (0.9 V), this range increases to 98% of the average MTTF. The largest range is seen at 65nm (1.0V) where the range is 128% of the average.

Second, we see that MTTF for applications correlates well with application temperature. The hottest applications (from Figure 3.2) have the lowest MTTFs. This is because, at any *given technology point*, the only difference in the MTTF of applications arises from temperature differences and from differences in the value of J (through the activity factor, p). However, the slope of the MTTF curves is steeper than the slope of the temperature curve. This is because of the more than linear dependence of MTTF on temperature (as can be seen in the temperature column in Table 3.1).

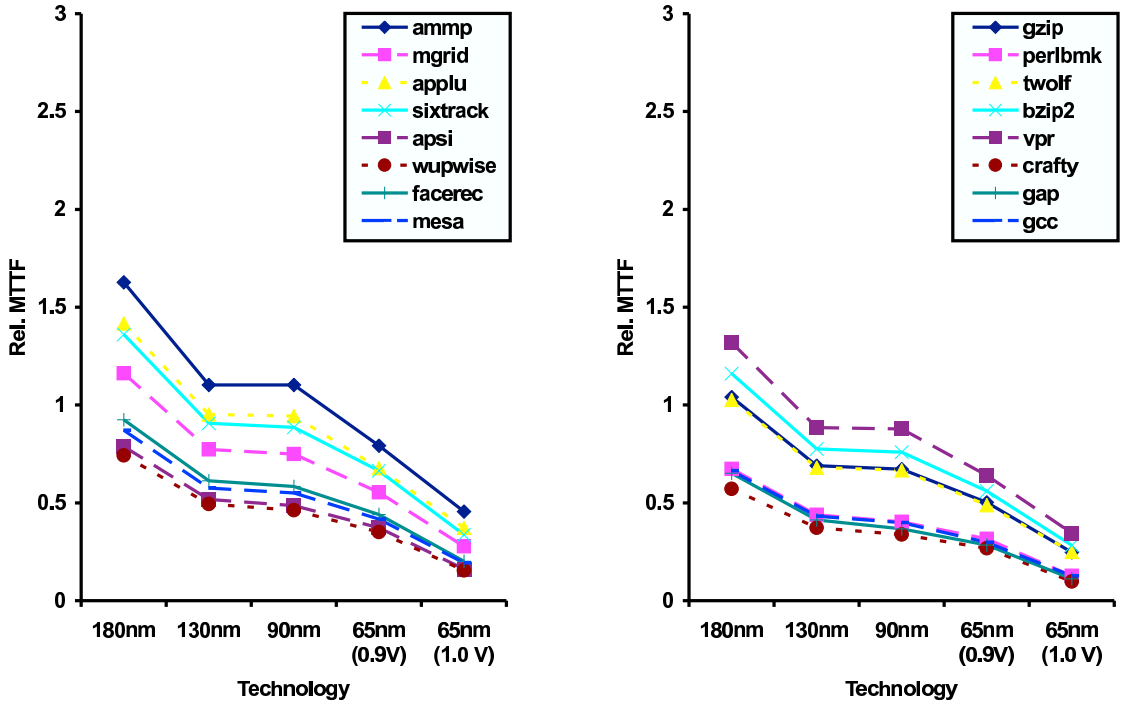
Our results indicate that future reliability qualification mechanisms should be targeted at specific applications or classes of applications. If an application oriented reliability qualification methodology is not used, the processor would be severely over-designed for most applications. This issue is also discussed in Chapter 5 where an application-aware dynamic reliability management scheme is proposed. Our quantification unequivocally shows the increasing importance of this and other application-aware reliability approaches, as processors are designed with increasingly smaller feature sizes.

3.4.3 Individual Failure Mechanisms

Next we examine scaling behavior of individual failure mechanisms, illustrated in Figures 3.5- 3.9. In each figure, the normalized MTTF due to the specific failure mechanism alone is shown. The average MTTF across all applications at 180nm is normalized to 1.0 for each failure mechanism.

EM Scaling

Scaling has a significant impact on electromigration MTTF – going from 180nm to 65nm (1.0V), the normalized MTTF decreases by 73% on average for SpecFP and 75% on average for SpecInt. Going from 180nm to 65nm (0.9V), the decrease is 49% for SpecFP and 51% for SpecInt. As can be seen from Table 3.1, the decrease is due to temperature as well as a reduction in interconnect dimensions (w and h). The temperature dependence is underscored by the difference in MTTF between 65nm (0.9V) and 65nm (1.0V) (where the only distinction is from temperature).



(a) EM SpecFP **(b) EM SpecInt**

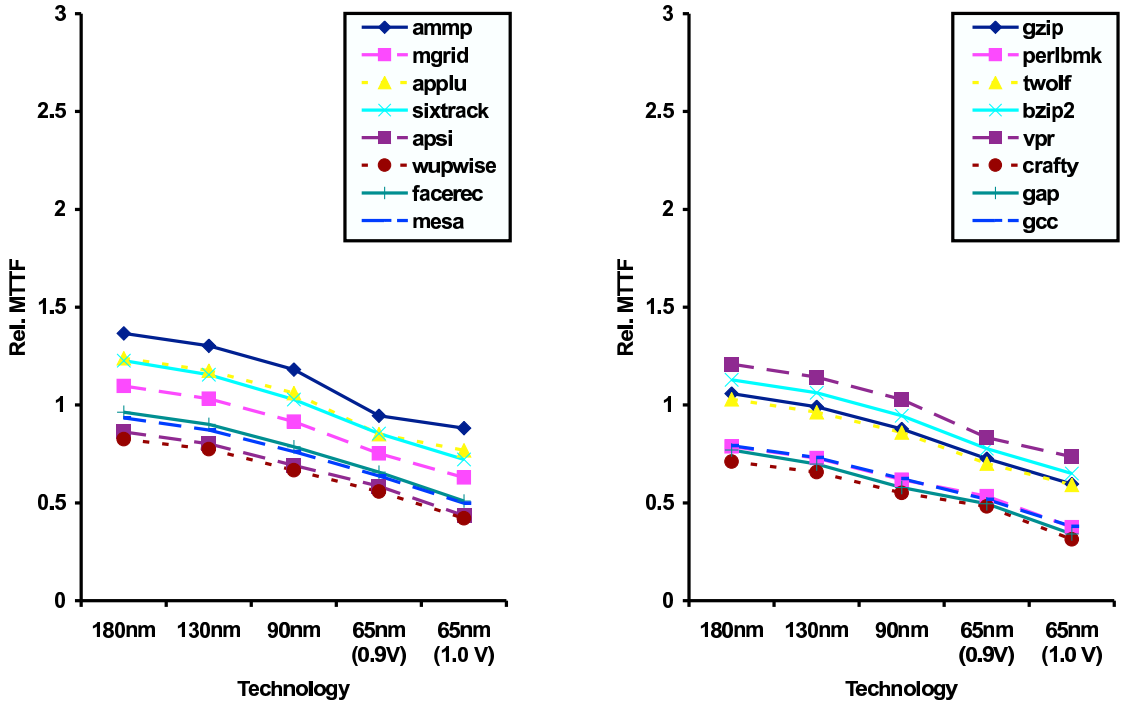
Figure 3.5: MTTF of EM alone for SpecFP and SpecInt at different technology generations. The average MTTF across all applications is normalized to 1.0 for each failure mechanism at 180nm.

SM Scaling

For SM, there is a 41% decrease in normalized MTTF values going from 180nm to 65nm (1.0V) and a 29% increase going from 180nm to 65nm (0.9V) for SpecFP on average. The corresponding values for SpecInt are 44% and 30%. Scaling impacts stress migration through an increase in temperature. The exponential dependence of stress migration failure rate on temperature (as shown in Table 3.1) can be seen in Figure 3.6. Like electromigration, the large drop in MTTF between 65nm(0.9V) and 65nm (1.0V) is entirely due to the exponential impact of temperature. However, this decrease is smaller than the decrease seen in electromigration due to the $|T - T_0|^{-m}$ term in the stress migration equation (Equation 2.4). This term improves reliability with scaling, but its impact is overshadowed by the exponential relationship.

TDDB Scaling

As can be seen in Table 3.1, TDDB MTTF value depends heavily on the values of V and t_{ox} used. There is also a more than exponential dependence on temperature. The negative effect of t_{ox} combined with

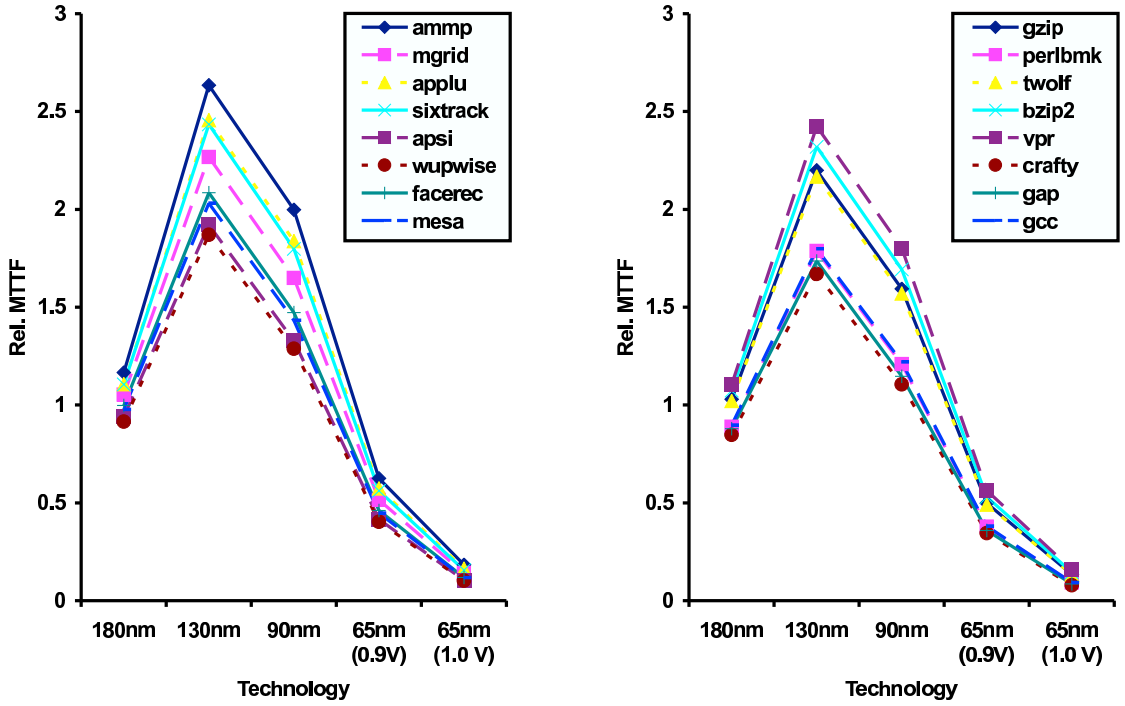


(a) SM SpecFP **(b) SM SpecInt**

Figure 3.6: MTTF of SM alone for SpecFP and SpecInt at different technology generations. The average MTTF across all applications is normalized to 1.0 for each failure mechanism at 180nm.

temperature results in an overall decrease in TDDB reliability with scaling, despite the positive effect of voltage scaling. This is compounded by the non-ideal scaling of voltage. As a result, these factors contribute to the huge decrease in normalized MTTF from 180nm to 65nm (1.0V) – 82% on average for SpecFP and 84% for SpecInt. The decrease from 180nm to 65nm (0.9V) is less severe, but still significant (50% for SpecFP and 52% for SpecInt).

Unlike the other failure mechanisms, the change in TDDB MTTF values does not completely follow the change in temperature. This is because of the voltage dependence of TDDB. Hence, although the temperature increases from 180nm to 130nm, the drop in voltage between these two technology points increases the MTTF value. The beneficial impact of voltage is highlighted by the large difference between the MTTF at 65nm (0.9V) and 65nm (1.0V) (the difference is magnified further due to the temperature difference between the two points).



(a) TDDB SpecFP **(b) TDDB SpecInt**

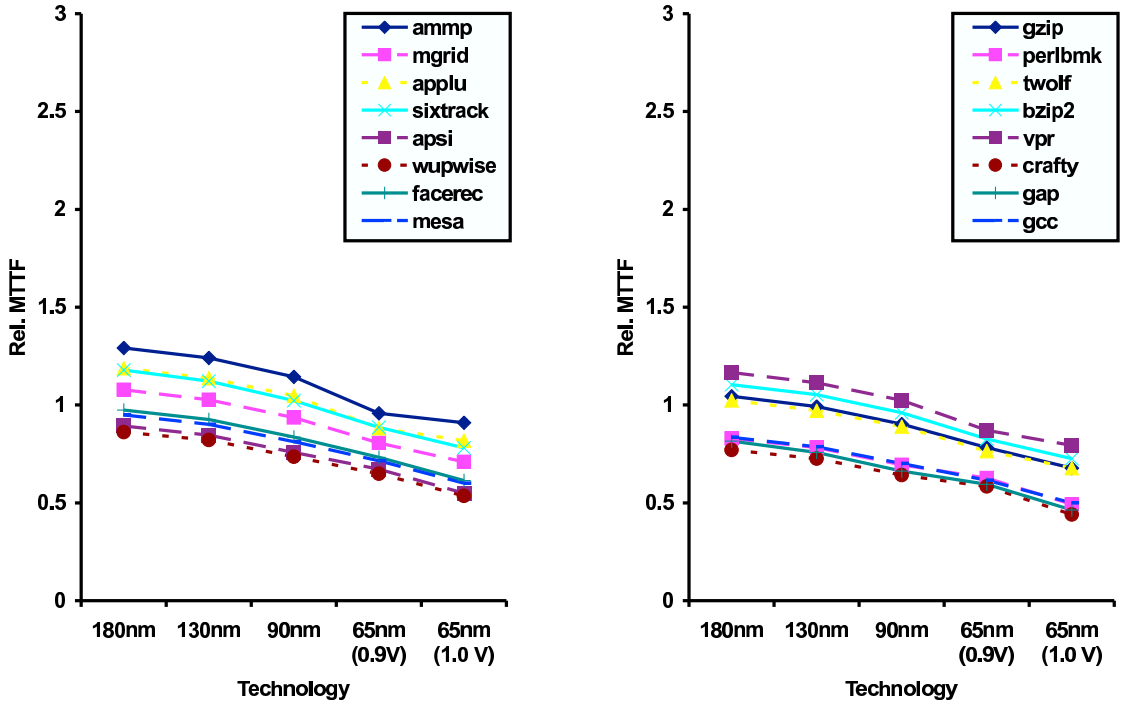
Figure 3.7: MTTF of TDDB alone for SpecFP and SpecInt at different technology generations. The average MTTF across all applications is normalized to 1.0 for each failure mechanism at 180nm.

TC Scaling

There is a 32% decrease in TC normalized MTTF going from 180nm to 65nm (1.0V) and a 23% decrease going from 180nm to 65nm (0.9V) for SpecFP on average. The corresponding values for SpecInt are 35% and 24%. Like stress migration, scaling impacts the MTTF due to thermal cycling through an increase in temperature. However, unlike stress migration which has an exponential dependence on temperature, thermal cycling varies as the power of q, which is the Coffin-Manson exponent (as seen in Table 3.1). In our experiments, we used a value of 2.35 for q. Hence, although there is an decrease in MTTF value due to temperature with scaling, the decrease is less steep than stress migration.

NBTI Scaling

There is a 36% decrease in NBTI normalized MTTF going from 180nm to 65nm (1.0V) and a 26% decrease going from 180nm to 65nm (0.9V) for SpecFP on average. The corresponding values for SpecInt are 40% and 27%. Like stress migration and thermal cycling, scaling impacts the NBTI MTTF through an increase



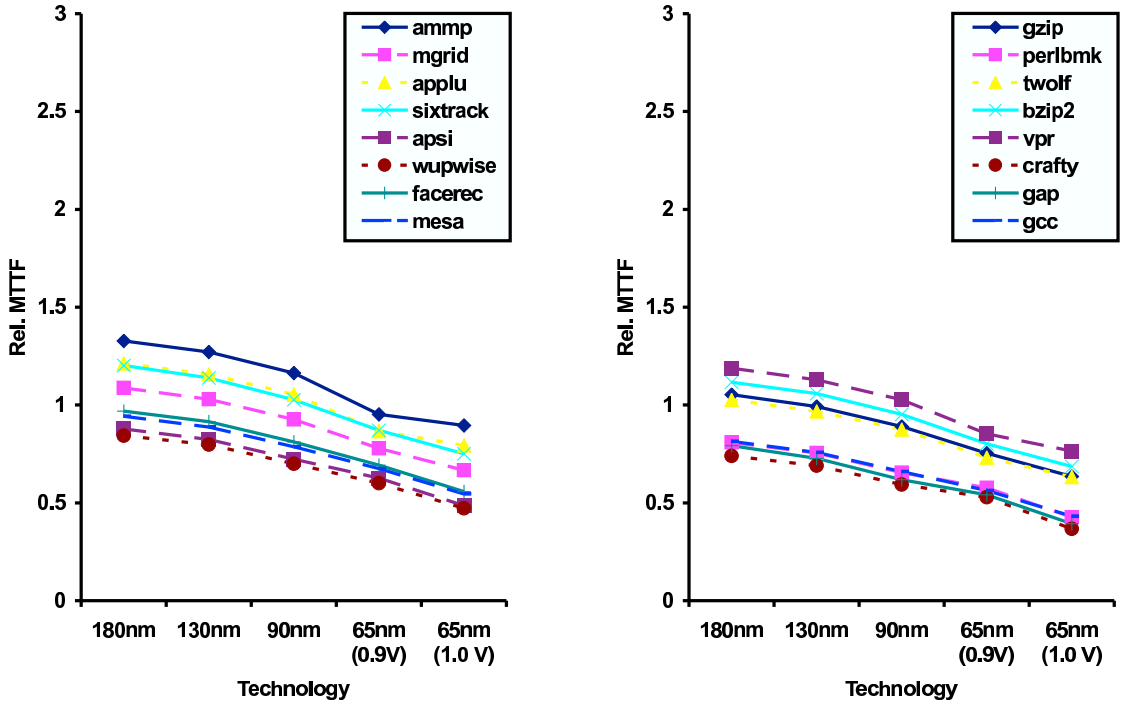
(a) TC SpecFP **(b) TC SpecInt**

Figure 3.8: MTTF of TC alone for SpecFP and SpecInt at different technology generations. The average MTTF across all applications is normalized to 1.0 for each failure mechanism at 180nm.

in temperature. This increase in temperature exponentially decreases NBTI MTTF. However, there are two other temperature dependent terms in the NBTI MTTF equation which improve MTTF with an increase in temperature. These terms impact MTTF as the natural logarithm of an exponential which is much less than exponential in effect. As a result, there is an exponential decrease in NBTI MTTF with scaling.

As discussed in Section 3.2.5, one limitation of our NBTI MTTF results is that we do not currently model the impact of feature size scaling and supply voltage scaling on NBTI MTTF. In particular, reduction in gate-oxide thickness will result in higher gate electric fields which will incur a further substantial decrease in NBTI MTTF.

Our individual failure mechanism scaling results mirror the results seen in Figure 3.4 which shows the relative probability of processor failure due to each failure mechanism, averaged across all SpecInt and SpecFP applications. At 180nm, the relative probabilities due to each mechanism are equal (20%). As can be seen, with scaling, the contribution of EM and TDDB grows substantially, while the relative contribution of SM, TC, and NBTI decreases. As we can see in Figures 3.5- 3.9, this is due to the compounded effect of



(a) NBTI SpecFP **(b) NBTI SpecInt**

Figure 3.9: MTTF of NBTI alone for SpecFP and SpecInt different technology generations. The average MTTF across all applications is normalized to 1.0 for each failure mechanism at 180nm. These results do not model the impact of voltage or feature size scaling on NBTI MTTF.

temperature and other scaling related parameters on the MTTF of TDDB and EM.

3.4.4 Comparison of RAMP 2.0 and RAMP 1.0 Scaling Results

As explained in Section 2.4, the SOFR model used in RAMP 1.0 is commonly used in industry to combine failure rates. However, this requires all failure mechanisms to be modeled with exponential failure distributions which leads to some inaccuracies. In order to quantify this inaccuracy, we repeated some of our scaling experiments using RAMP 1.0, allowing us to compare those results with our results using RAMP 2.0. In both cases, we modeled series-failure systems. At 180nm, the average MTTF across all applications is normalized to 1.0 for both RAMP 2.0 and RAMP 1.0.

Figure 3.10 shows the average MTTF across all SpecInt and SpecFP applications, using RAMP 2.0, which uses lognormal failure distributions (labeled as LOG), and RAMP 1.0, which uses exponential failure distributions (labeled as EXP), at different technology generations. The RAMP 2.0 results (lognormal results) are shown with dotted lines.

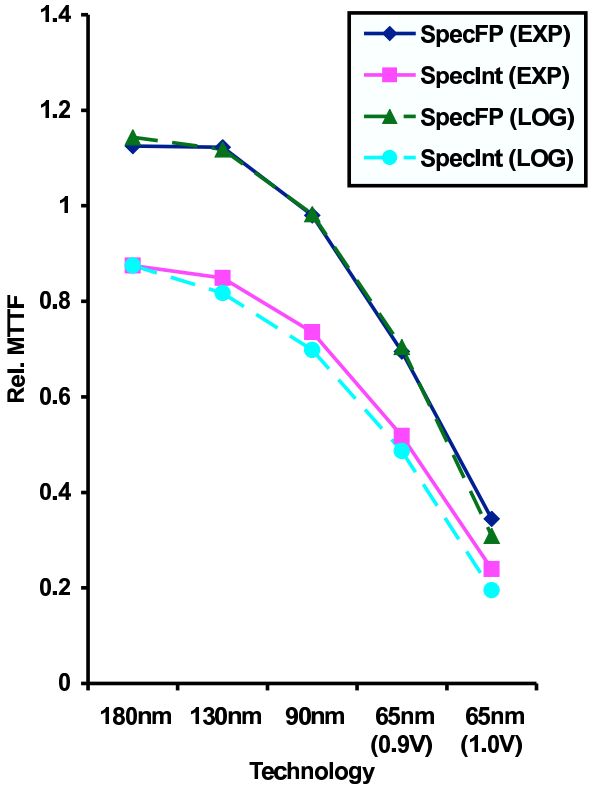


Figure 3.10: Average MTTF across all SpecInt and SpecFP applications, using RAMP 2.0, which uses lognormal failure distributions (labeled as LOG), and RAMP 1.0, which uses exponential failure distributions (labeled as EXP), at different technology generations.

As can be seen, the lognormal and exponential results diverge past 180nm, with the lognormal results being more conservative than the exponential results. More importantly, the magnitude of the difference between lognormal and exponential results increases as we scale to smaller technology generations. Specifically, at 65nm (0.9V), the average MTTF as seen with RAMP 1.0 is 16% higher than the average MTTF with RAMP 2.0 for SpecInt, and 4% higher for SpecFP. At 65nm (1.0V), RAMP 1.0 generates an MTTF 35% higher than RAMP 2.0 for SpecInt applications, and 19% higher for SpecFP applications.

These results clearly highlight the importance of using accurate failure distributions. Although RAMP 1.0 is easier to use due to the simplicity of the SOFR model, its use can lead to inaccurate results. This is particularly important because RAMP 1.0 provides *less conservative* results in our scaling analysis. Less conservative results can lead to unintentional under-designing of processors from a reliability perspective.

3.5 Summary

Advances in CMOS semiconductor technology, driven by aggressive device scaling, have been steadily improving processor performance. However, CMOS scaling is resulting in escalated power densities and processor temperatures, and accelerating the onset of problems due to long-term processor hardware failures or lifetime reliability.

In this chapter, we take a first step in understanding the implications of scaling in the deep-submicron era at the architect's level. Our results point to potentially large and sharp drops in long-term reliability, especially beyond 90 nm. Of the failure modes that were modeled, time-dependent dielectric breakdown (TDDB) and electromigration appear to present the steepest challenge. However, the impact of voltage and feature size scaling on NBTI was not modeled. Our results also illustrate how scaling is increasing the difference between failure rates assuming worst-case conditions vs. typical operating conditions.

Our results present two broad implications. First, it will become increasingly difficult to leverage a single microarchitectural design for multiple remaps across a few technology generations. Second, the need for workload specific, microarchitectural lifetime reliability awareness is illustrated.

Chapter 4

Future Improvements and Validation of RAMP

Before we move to our microarchitectural reliability enhancement techniques, it is important to understand the assumptions and limitations of RAMP. As with any simulator or model, a key requirement for RAMP's usability is validation of the assumptions in the model. It should be noted that although some aspects of RAMP have not been validated, most of our assumptions are grounded in "current practice", and were developed after extensive consultations with research and product groups that concentrate on processor reliability qualification at IBM. Further, the individual failure mechanism models, which are the key underlying components of RAMP, represent the state-of-the-art. Although a thorough validation of RAMP is beyond the scope of this thesis, we discuss some of the required steps. In addition, some of the components of RAMP need to be improved upon, as and when better models are available. In this chapter, we discuss some of the validation steps and improvements required.

4.1 Validation of RAMP

4.1.1 Calibration with Real Failure Data

As discussed in Section 2.5, the proportionality constants in Equations 2.1, 2.4, 2.6, 2.8, and 2.9 dictate the absolute MTTF value produced by RAMP. Since there are many steps involved in the reliability qualification process, determining the numerical values of these proportionality constants is difficult. Instead, we use T_{qual} as a proxy for reliability qualification cost. As mentioned earlier, T_{qual} is required only in situations where we need to compare absolute values (as in Section 5.5). We do not need to use qualification proxies when comparing relative values (as in Sections 3.4 and 6.3).

A true validation of RAMP would require corroboration with real field failure data. However, such data is hard to come by. In addition, even in situations where such data is available, reliable information about cause of and exact location of failure is rare. If such data is available for a large batch of processors, three important parameters can potentially be determined: (1) Ratio of failures among different failure mechanisms; (2) Ratio of failures among different structures; (3) Absolute value of MTTF of the batch of

processors.

With these parameters, we can determine the value of the proportionality constants in the equations, allowing us to calculate absolute MTTF values for any architecture and application.

4.1.2 Finer Than Structure Granularity Modeling

As mentioned, RAMP currently models reliability at the granularity of a microarchitectural structure. However, this abstraction averages over many aspects of reliability. For example, different interconnects in a structure will have different electromigration properties, and will have different lifetimes – this is not captured by RAMP currently. Modeling reliability more accurately by working at finer-than-structure granularities will require more detailed thermal and energy models and layout information. In addition, the simulation time required will increase considerably. With accurate reliability data, an optimal simulation granularity that balances accuracy and simulation time can be potentially determined.

4.2 Future Improvements

4.2.1 Time Dependence

RAMP 1.0 does not adequately incorporate time dependence of failure rates of the different failure mechanisms. RAMP 2.0 improves upon RAMP 1.0 by modeling failure mechanisms with lognormal distributions. Although lognormal distributions have been shown to be a good fit for wear-out mechanisms [31], other more complex failure distributions might provide an even better fit. Further work by device failure researchers is required to determine the appropriate failure distribution for each failure mechanism. Using Monte-Carlo or other similar methods, the failure distribution for each failure mechanism can be combined to obtain total processor MTTF. Additionally, we currently use time-averaging to account for the time-varying properties of applications in both RAMP 1.0 and 2.0. This assumption needs to be improved upon or validated.

4.2.2 Electric Field Model for NBTI

As explained in Section 3.2.5, thinning of the gate oxide due to scaling increases NBTI failures due to an increase in the electric field across the gate. We do not currently have accurate models for the impact of electric field changes on NBTI MTTF. Given that this is an important emerging failure mechanism, an electric field model for NBTI is important.

4.2.3 High Frequency Thermal Cycles

As explained in Section 2.1.4, the package goes through thermal cycling (TC) in two ways – large temperature variations at low frequencies and smaller variations at high frequencies. Although RAMP models large cycles, it does not model small cycles due to the lack of validated models for this mechanism. However, recent on-chip power management modes increase the number and impact of such high frequency thermal cycles, and this has increased further interest in this failure mode. This is a particularly intriguing failure mode in our context since it is worsened by conventional power management techniques, and our adaptation algorithms will have to carefully incorporate the adverse impact on reliability of common power management solutions. When available, RAMP should model the effect of high frequency thermal cycles.

4.2.4 Other Wear-Out Based Failure Mechanisms

Based on discussions with industry researchers, the failures currently modeled in RAMP were chosen as the most critical to processor lifetime reliability. However, the modular design of RAMP allows us to seamlessly add further failure mechanisms. Other critical failure mechanisms like Hot Carrier Injection (HCI) [18] should be studied and modeled if necessary.

4.2.5 Impact of Sensors

Any runtime temperature measurement scheme requires the use of sensors. There is a tradeoff between the measurement accuracy of sensors and their power, temperature, and area requirements. These requirements also restrict the number of sensors that can be used on any given chip. RAMP should account for the effect of sensor error in temperature and reliability simulations.

Chapter 5

Dynamic Reliability Management

5.1 Motivation for Microarchitectural Reliability Solutions

The scaling results discussed in Chapter 3 clearly highlight the emerging lifetime reliability problem. In future reliability constrained systems, microarchitectural reliability solutions will be beneficial, if not necessary. This is true for all market segments ranging from server class processors where lifetime reliability is an implicit requirement, to commodity processors where reliability impacts the number of processors shipped (yield) and resultant profit.

Extensive research has gone into techniques that can improve energy and thermal efficiency by exploiting microarchitectural features and adaptation capabilities (for e.g., [51, 10, 11, 52, 53, 54, 55, 56]). A similar approach can be used for lifetime reliability – the microarchitecture's unique knowledge of application run-time behavior can be leveraged to increase processor reliability. Such an approach to reliability is fundamentally different from existing methodologies where processor reliability is qualified during device design, circuit layout, manufacture, and chip test. Current reliability qualification mechanisms are based on worst-case temperature and utilization estimates and are oblivious to application behavior. However, different applications have different effects on the processor's lifetime due to variations in IPC, resource utilization, and temperature. In this chapter, we propose Dynamic Reliability Management (DRM), a technique that tracks and responds to application behavior to maintain the processor's lifetime reliability target. DRM can be used in two scenarios:

- **Under-designed processors:** An adaptive approach can be used to safely under-design processors from a reliability approach, thereby saving cost. In an approach similar to dynamic thermal management (DTM) [11], the processor reliability qualification can be based on expected processor utilization and temperature, rather than worst case values. This would result in significant design cost reductions and would provide higher processor yield. In situations where applications exceed the reliability design limit, the processor can adapt by throttling performance to maintain the system reliability target. Such an approach would be beneficial to commodity processors where increasing yield and reducing

cooling costs would have significant impact on profits, even if they incur some performance loss. This allows DRM to act as an enabling technology for reliability under-designed systems where the cost of traditional reliability qualification could be too prohibitive to stay on the required performance curve.

- **Over-designed processors:** As mentioned, current reliability qualification is based on worst case temperature and utilization; however, most applications will run at lower temperature and utilization resulting in higher reliability and longer processor lifetimes than required. If the processor cooling solution can handle it, this excess reliability can be utilized by the processor to increase application performance. For example, a judicious increase of voltage/frequency and/or microarchitectural resources could increase application performance while still maintaining system target reliability. Such an approach would be particularly beneficial in high-end server class processors. These processors tend to have expensive cooling and packaging and are over-designed from a reliability perspective, providing reliability margins that can potentially be used to increase performance.

In Chapter 6, we examine two additional reliability enhancement techniques, structural duplication and graceful performance degradation. These techniques are orthogonal in effect to DRM and can further improve processor reliability.

5.2 Dynamic Reliability Management (DRM)

Figure 5.1 motivates Dynamic Reliability Management (DRM). Three processors 1, 2, and 3, are depicted. They have reliability design points, T_{qual_1}, T_{qual_2}, and T_{qual_3}, such that $T_{qual_1} > T_{qual_2} > T_{qual_3}$. As discussed in Section 2.5, this implies that processor 1 is more expensive to qualify for reliability than processor 2, and processor 3 is the cheapest to qualify. Consider two applications, A and B (depicted on the y-axis). These two applications will have different MTTFs in the three processors, because the T_{qual} used to calculate the application's MTTF on each processor is different.

In processor 1, all applications meet the target MTTF, and in fact exceed it (i.e., their MTTF are higher than they are required to be). In processor 2, application A does not meet the target MTTF , but application B does. In processor 3, both applications do not meet the target MTTF. Hence, the expensive processor, 1, has been over-designed from a reliability perspective, while the cheaper processors, 2 and 3, have been under-designed.

Considering 2 and 3 first, although they are cheaper to design than 1, they can fail prematurely if no architectural intervention occurs, and so, do not represent acceptable design points by current reliability qualification methodologies. However, with DRM, we can design processors 2 and 3 to meet reliability targets

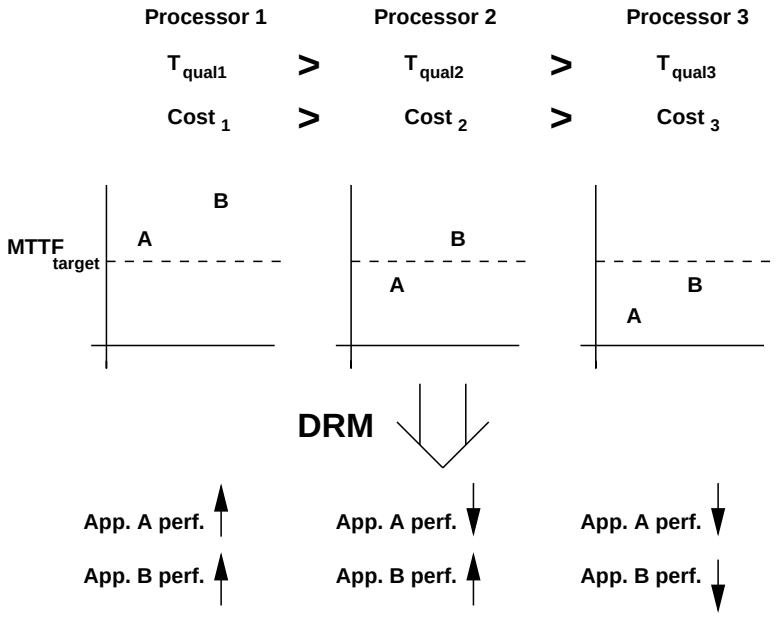


Figure 5.1: Dynamic Reliability Management (DRM). For different values of T_{qual}, FIT value of the processor running applications A and B is shown on the y-axis. DRM adapts the performance of the applications to meet the target FIT value.

by using processor adaptation to reduce processor temperature, utilization, voltage, and/or frequency, at the cost of throttling performance, but with higher reliability. Now, considering application B in processor 2 and both applications in processor 1, current systems will not exploit reliability over-design space. However, if the cooling solution can support it, DRM can be used to exploit the reliability margin and extract excess performance (e.g., by overclocking or increasing microarchitectural resources). Thus, DRM can be used both to decrease reliability qualification cost and to increase processor performance, while assuring reliability targets are met.

Like dynamic energy management (DEM) and dynamic thermal management (DTM), DRM requires adaptation response mechanisms and control algorithms to invoke these responses. We can leverage the extensive body of work on DEM and DTM (for e.g., [51, 10, 11, 52, 53, 54, 55, 56, 57]) for DRM as well. However, it is important to note that designing for energy, temperature, and reliability are distinct problems. Solving one does not automatically solve the other. Like energy, but unlike temperature, reliability is a long-term phenomenon and can be budgeted over time. Similarly, like temperature, but unlike energy, reliability is directly affected by power density (a spatially local parameter).

5.3 DRM Algorithm

The goal of our algorithm is to determine the highest performing architectural configuration (an architectural configuration consists of a given architecture running at a given frequency) which also maintains the target reliability (MTTF). Our algorithm utilizes two advantageous properties of reliability control not afforded by energy or thermal control: (1) Since typical processor MTTFs are in the order of tens of years, a key feature of reliability control is the extremely large time-frames afforded to the algorithm to determine the correct operating point. This is not the case for energy or thermal control where operating time frames are much smaller. (2) MTTF is an average function over the processor's entire lifetime. As a result, instantaneous MTTF values can violate the the target MTTF for periods of time as long as this is compensated later in the processor's life. Hence, in the process of determining the correct operating point, instantaneous MTTFs *can violate* the MTTF target. This is not the case for thermal control algorithms where exceeding the thermal limit can cause processor failure.

Our algorithm proceeds in two phases – a *profiling phase* and an *operating phase*. For each application, during the profiling phase, the algorithm determines the maximum performance that each hardware configuration can run at while staying within the MTTF target. The configurations are then ordered in decreasing order of performance. In the operating phase, the configuration with the highest performance that satisfies the MTTF target is used for the rest of the application's run.

The profiling phase is invoked at the start of the application. It measures the performance ($IPC \times frequency$) and MTTF of the application on each architectural configuration. To accurately determine the MTTF of an architectural configuration, each structure on chip has to achieve its correct steady state temperature. As described in [58], due to the large time constant of the heat-sink, this implies that the application has to run on each architectural configuration in the order of minutes. As a result, profiling all architectural configurations for each application can take hours. However, as discussed, due to the time frames involved, this profiling time will not have a noticeable effect on final processor reliability. At the end of the profiling phase, the architectural configurations are ordered in decreasing order of performance and the configuration with the highest performance that also satisfies the MTTF requirement is chosen. The operating phase of the algorithm which consists of the rest of the application's run uses the selected configuration.

Our DRM algorithm can be applied to single applications or workloads consisting of multiple applications. The key requirement is that each architectural configuration in the profiling phase should profile a representative and repetitive period of work. Depending on the usage of the profiled processor, this period can vary. For example, commercial servers might have varying usage depending on the day of the week. In

such a scenario, each architectural configuration should ideally be profiled over an entire week. As explained earlier, such long profiling periods are possible due to the large time frames afforded in reliability control.

5.3.1 DRM Adaptations

We study DRM by profiling a wide range of microarchitectural configurations, and voltage and frequency settings, and selecting configurations that would give maximum performance, for different values of T_{qual}. Specifically, the adaptations we explore are:

- **Microarchitectural adaptation (Arch).** For every application, for a range of T_{qual} values, we profile a range of microarchitectural configurations. At the end of the profiling phase, we select the architectural configuration that gives the best performance while still within the target MTTF value. The voltage and frequency is the same for all Arch configurations.
- **Dynamic voltage and frequency scaling (DVS).** For every application, for a range of T_{qual} values, we profile a range of voltages and frequencies, and select the one which gives the best performance while still within the target MTTF value. We use the most aggressive microarchitectural configuration supported.
- **Microarchitectural adaptation and DVS (ArchDVS).** In this case, we explore combinations of microarchitectural configurations and DVS settings, for each application, for different T_{qual} values.

5.4 DRM Experimental Methodology

5.4.1 Architectures Studied

The base non-adaptive processor studied is summarized in Table 5.1 . The base processor is similar to the MIPS R10000.We assume a centralized instruction window that integrates the issue queue and reorder buffer (ROB), but has a separate physical register file. Given that reliability concerns will be amplified in future technologies (as seen in Section 3.4, we model a 65nm processor, with a supply voltage, V_{dd}, of 1.0 V and a base frequency of 4 GHz. The core size, and size of different structures, was estimated from current processor sizes, scaled appropriately, and does not include the L2 cache. Although we model the performance impact of the L2 cache, we do not model its reliability. This is because the temperature of the L2 cache will be much lower than the temperature of the core.

For the DRM voltage and frequency adaptations, we vary the processor frequency from 2.8GHz to 4.4GHz. We always set the voltage such that it supports the frequency being simulated. The relationship between

Technology Parameters	
Process technology	65 nm
V_{dd}	1.0 V
Processor frequency	4.0 GHz
Processor core size (not including L2 cache)	$20.2mm^2$ (4.5mm x 4.5 mm)
Leakage power density at $383K$	0.5 W/mm^2
Base Processor Parameters	
Fetch/retire rate	8 per cycle
Functional units	6 Int, 4 FP, 2 Add. gen.
Integer FU latencies	1/7/12 add/multiply/divide (pipelined)
FP FU latencies	4 default, 12 div. (all but div. pipelined)
Instruction window (reorder buffer) size	128 entries
Register file size	192 integer and 192 FP
Memory queue size	32 entries
Branch prediction	2KB bimodal agree, 32 entry RAS
Base Memory Hierarchy Parameters	
L1 (Data)	64KB, 2-way associative, 64B line, 2 ports, 12 MSHRs
L1 (Instr)	32KB, 2-way associative
L2 (Unified)	1MB, 4-way associative, 64B line, 1 port, 12 MSHRs
Main Memory	16B/cycle, 4-way interleaved
Base Contentionless Memory Latencies	
L1 (Data) hit time (on-chip)	2 cycles
L2 hit time (off-chip)	20 cycles
Main memory (off-chip)	102 cycles

Table 5.1: Base non-adaptive processor for DRM experiments

voltage and frequency used was extrapolated from the information available for DVS on Intel's Pentium-M (Centrino) processor [59].

For the architectural adaptations used in DRM, we model 18 architectural configurations (consisting of combinations of the instruction window size, number of ALUs, and number of FPUs), ranging from a 128 entry instruction window, 6 ALU, 4 FPU processor, to a 16 entry instruction window, 2 ALU, 1 FPU processor. The issue width of the processor is equal to the sum of all active functional units and hence changes when we change the number of active functional units. Since we adapt the issue width of the processor with functional unit adaptation, we power down the selection logic corresponding to the functional units that are powered down. Also, when a functional unit is powered down, the corresponding part of the result bus, the wake-up ports to the instruction window, and write ports to the register file are also powered down. When a structure is powered down, since it has no current flow or supply voltage, it can not have any failures due to electromigration or TDDB. Hence, the failure rate due to electromigration and TDDB of any adaptive structure on chip is proportional to the powered on area of the structure. The leakage power consumption

Application	Type	IPC	Base power (W)
MPGdec (Mpeg video decoder)	Multi-media	3.4	39.7
MP3dec (Mp3 audio decoder)		3.0	37.1
H263enc (H263 video encoder)		2.2	35.5
bzip2	SPEC2k Integer	1.8	24.7
gzip		1.7	25.6
twolf		1.2	22.4
art	SPEC2k Float	0.7	17.3
equake		1.6	25.1
ammp		1.2	21.5

Table 5.2: Workload description for DRM experiments

of adaptive structures on chip is also proportional to the powered on area of the structure (Section 3.3.2 describes the leakage power modeling methodology.).

Finally, it should be noted that our base nonadaptive processor uses the most aggressive architectural configuration available. The architectural adaptations we model can only reduce the complexity of the processor, relative to base, and not increase it. Also, Arch can not change processor frequency. As a result, the maximum possible performance of any application with DRM algorithm Arch will be 1.0, where it will be running at the base configuration at the base frequency. On the other hand, DRM algorithms DVS and ArchDVS can increase the processor frequency greater than the base value , and can have a performance greater than 1.0.

5.4.2 Workload Description

Table 5.2 summarizes the nine applications used for our DRM experiments. In order to study the reliability implications of various application classes, we choose three multimedia applications, 3 SPEC2k integer applications, and 3 SPEC2k floating point applications. For each of the applications, the IPC and power consumption of the base non-adaptive processor is given. The base power consumption shown in Table 5.2 also includes leakage power.

As can be seen, a wide range of IPCs and power consumptions are observed. For the DRM study, it was more important to study applications which show a wide range of behavior, rather than perform a comprehensive study of the SPEC benchmark suite.

For the SPEC benchmarks, we fast forward 1.5 billion instructions to pass initialization code, and then we simulate 500 million instructions. The multimedia applications are frame based applications which do not have an explicit initialization phase. Hence, we simulate the multimedia applications for 500 million instructions (at least 400 frames) without fast forwarding.

5.4.3 Simulation Methodology

Simulator

We use the RSIM simulator [60] for performance evaluation. We use the Wattch tool [20] integrated with RSIM for power measurement. We derive temperature from power using the HotSpot tool [11]. The chip floorplan fed to HotSpot resembles the MIPS R10000 floorplan (without L2 cache), scaled down to $20.2mm^2$ (4.5 mm x 4.5 mm). Wattch assumes extensive clock gating for all the components of the processor with 10% of its maximum power charged to a component when it is not accessed in a given cycle. Temperature and reliability measurements are performed at the granularity of 1μ second. Our leakage power modeling methodology is identical to that described in Section 3.3.2.

Temperature Simulation

The temperature simulation methodology for our DRM experiments is identical to the methodology used in our scaling experiments (Section 3.3.3).

Reliability Calculation

We use RAMP 1.0 for our DRM reliability measurements. Since RAMP 2.0 requires post-processing for its Monte-Carlo methods, it can not be used for run-time reliability adaptation techniques like DRM.

After heat-sink temperature initialization, for each application, we fast forward 1.5 billion instructions to pass initialization code and profiled over the next 500 million instructions. The operating phase of the DRM algorithm was then evaluated over the same 500 million instructions.

5.5 DRM Results

5.5.1 Designing Processors for Different T_{qual}

Figure 5.2 shows the performance for all the applications, when using the combination of microarchitectural adaptation and DVS (ArchDVS) to control reliability by DRM for a range of T_{qual} values. Performance is represented as an increase or slowdown over the base non-adaptive processor, with a value of 1.0 representing no gain or loss. As mentioned in Section 2.5, we use T_{qual} as a proxy for reliability design cost. Results are shown for four values of T_{qual} – 400K, 370K, 345K, and 325K, which represent four qualification levels, ranging from most expensive to cheapest.

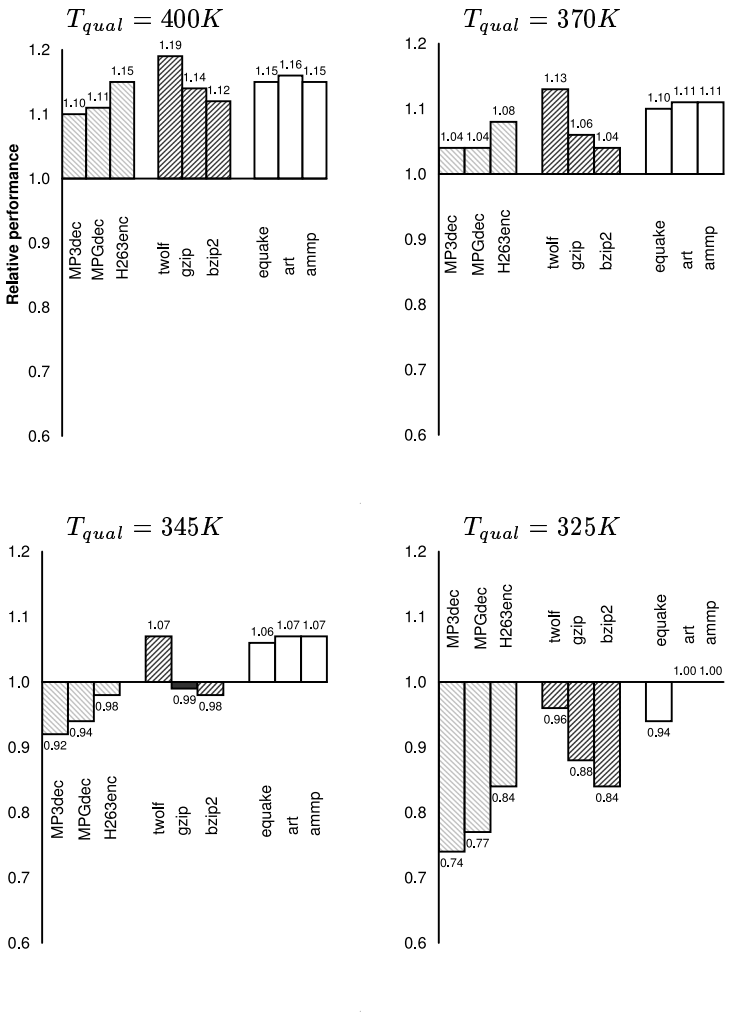


Figure 5.2: The performance of ArchDVS for DRM is shown on the y-axis, relative to the base non-adaptive architecture at 4 GHz. This is shown for all the applications for different T_{qual} values.

$T_{qual} = 400K$

The hottest temperature reached on chip by any application for our benchmark suite was near 400K. Hence, this value of T_{qual} represents a lower bound on the qualification temperature that would be chosen using current methodology for reliability qualification, based on worst-case conditions. As can be seen, all the applications experience significant performance gains (ranging from a gain of 10% for MP3dec to 19% for twolf) while still maintaining required processor reliability levels. This is because the operating conditions on chip while an application runs generally tend to be much lower than the worst case values, so all the applications can run at higher than base frequency. At the higher frequency, the temperature will occasionally exceed 400K but the average application MTTF will not fall below the target because lower instantaneous MTTFs are compensated by higher values at other times.

The performance gains experienced by the SPEC benchmarks tend to be higher on average than those of the multimedia benchmarks. This is because the multimedia benchmarks have higher IPCs, and consequently higher operating temperatures and activity factors, which gives them lower MTTFs on the base processor than the SPEC benchmarks.

Based on the above results, we can see that qualifying for worst case operating conditions is overly conservative – instead, we could either design to a lower T_{qual}, which would result in cost savings, or the base non-adaptive processor can be marketed at a higher frequency (while still meeting the reliability target).

$T_{qual} = 370K$

At a T_{qual} value of 370K, the applications with the lowest MTTFs on the base non-adaptive processor (MP3dec and MPGdec) have almost no performance gain. All the other applications have a performance gain ranging from 4% for bzip2 to 13% for twolf. This represents a processor which is qualified for reliability based on application behavior. Rather than selecting T_{qual} based on the worst case application operating temperature of 400K, T_{qual} was chosen such that the worst applications (MP3dec and MPGdec) just meet the reliability target. Such an *application oriented* approach to reliability qualification represents significant savings in qualification cost without any loss of performance (DRM never curtails performance in this scenario for these applications). Again, lower IPC applications see the largest performance gains (twolf and art).

$T_{qual} = 345K$

A T_{qual} value of 345K represents a processor qualified for the average application, rather than worst case application. As can be seen, the performance seen by all the applications with DRM was within 10% of the base value, and in four cases, was within 5%. This potentially represents an excellent cost-performance tradeoff design point, where DRM can be used to underdesign a processor, without incurring significant performance penalties. As is expected, high IPC applications experience the largest performance losses, while low IPC applications enjoy the largest gains.

$T_{qual} = 325K$

A T_{qual} value of 325 K represents a processor which has been drastically underdesigned from a reliability perspective. All applications, with the exception of art and ammp, experience a slowdown. The high IPC multimedia applications experience the largest slowdown, with MP3dec suffering a loss of 26% in performance. This scenario potentially represents a case where the cost benefit of designing for a cheaper T_{qual} is overshadowed by the loss in performance seen.

Implications of Designing for Reliability

From the above results, we see that there is potential for significant cost benefit, without any performance loss, using DRM. Changing the reliability design point from T_{qual} of 400K to 370K, saves design cost, without requiring any of the applications to slow down. This shows that worst case reliability qualification is overly conservative.

Using DRM, by allowing some performance degradation, we can further lower the value of T_{qual}. In our results, even at a T_{qual} of 345K, the performance loss seen was limited. Hence, a wide spectrum of T_{qual} values (in our case, 345K to 400K) are available to designers, for a reasonable performance tradeoff.

Finally, we see that the performance-cost tradeoff depends on the processor's intended application domain. For example, a processor designed for SPEC applications could be designed to a lower T_{qual}, than a processor intended for multimedia applications. In the situation that an application causes the processor to exceed the reliability target, DRM can be employed to maintain reliability.

Again, as explained in Section 2.5, we can not make quantitative assessments of cost versus performance because we do not have the exact proportionality constants used in RAMP's failure models.

5.5.2 Comparing Different DRM Adaptations

Figure 5.3 compares the performance (relative to the base non-adaptive processor) for the three DRM adaptations, Arch, DVS, and ArchDVS, for a range of T_{qual} values. As can be seen, DVS and ArchDVS significantly outperform Arch, performing an average of 27% better at a T_{qual} value of 335K. Also, ArchDVS chose to perform DVS on the base processor most of the time – hence, there is very little difference between DVS and ArchDVS.

DVS and ArchDVS, which can both adapt frequency and voltage, perform much better than Arch due to three main reasons:

1. Small drops in voltage and frequency result in large drops in temperature – this is because of the near cubic relationship between frequency and power, which translates into a large temperature drop. In comparison, Arch does not cause such a large drop in processor power, necessitating a larger performance hit.

2. As can be seen in Equation 2.6, there is a very large voltage dependence on TDDB MTTF. Hence, small drops in voltage and frequency increase the TDDB MTTF value drastically.

3. As mentioned, in Section 5.4.1, the performance due to Arch can never be greater than 1.0, since it cannot adapt the processor's frequency. Hence, in any scenario where processor performance can be

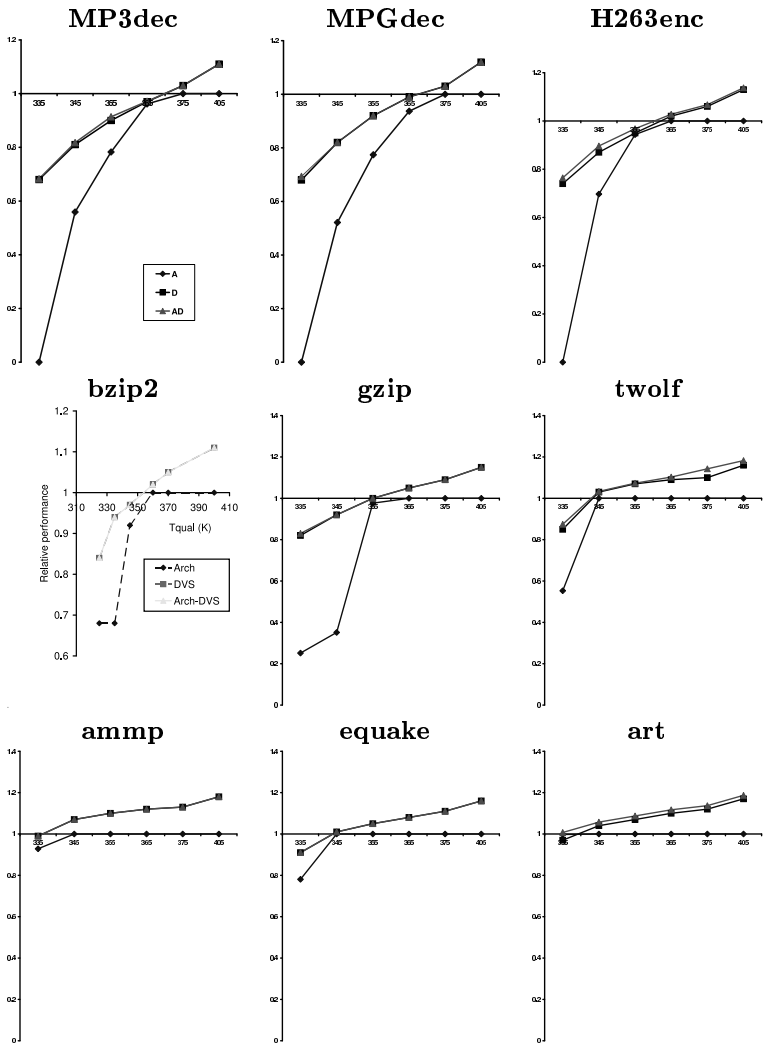


Figure 5.3: Comparison of different DRM adaptations. The x-axis represents different T_{qual} values and the y-axis is performance speedup or slowdown.

increased because of an overdesigned T_{qual}, DVS and ArchDVS will perform better than Arch (this is seen for T_{qual} values between 360K and 400K in Figure 5.3).

Hence, it is clear that DVS is more beneficial than the microarchitectural adaptations for the space we explored for DRM.

5.5.3 Comparing DRM and DTM

This section makes the case that DTM algorithms do not subsume reliability concerns and vice versa; i.e., both thermal and reliability constraints need to be considered as first-class design entities.

Figure 5.4 compares DRM and DTM using voltage and frequency scaling (DVS) for all the applications.

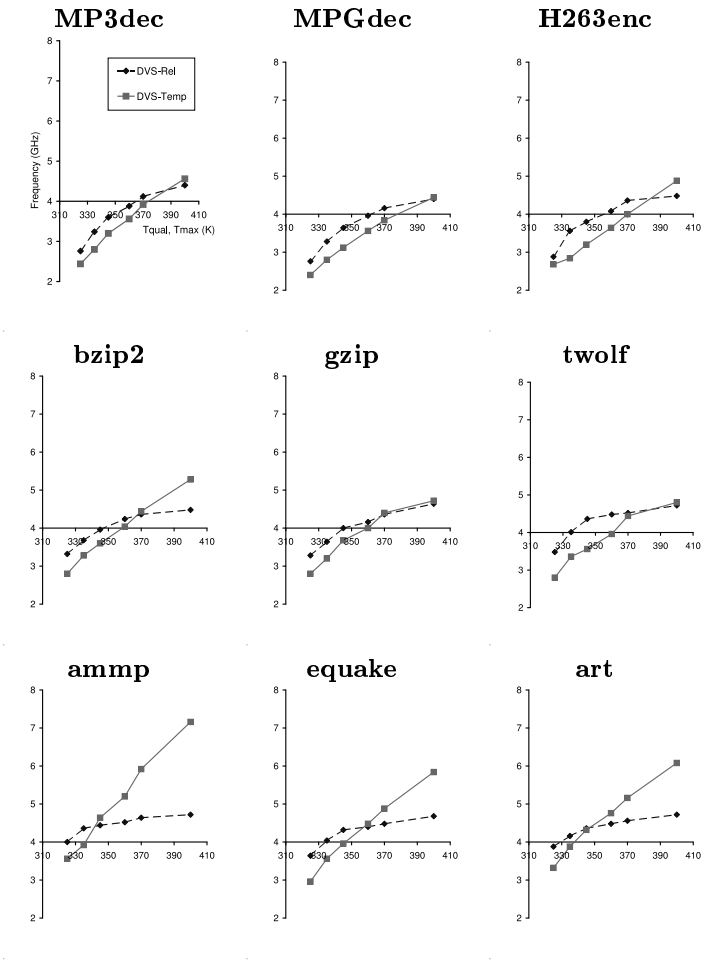


Figure 5.4: Comparing design for reliability and temperature. Temperatures on the x-axis represent T_{qual} for DRM and T_{max} for DTM. The frequency, in GHz, chosen by DVS for DRM (dotted line) and DTM (solid line) is shown on the y-axis. We evaluated values at 325K, 335K, 345K, 360K, 370K, and 400K.

Each point on the horizontal axis is a temperature value, which represents the qualifying temperature for DRM (T_{qual}), and the thermal design point (T_{max}), for DTM. For each of these temperatures, the optimal frequency chosen by DVS on the base non-adaptive processor for DRM (Curve DVS-Rel in the figure) and DTM (curve DVS-Temp) is shown on the vertical axis. That is, the DVS-Rel curve ensures highest performance at the target MTTF and the DVS-Temp curve ensures highest performance without exceeding T_{max}.

Unlike T_{max} which represents the maximum temperature the processor is allowed to reach, T_{qual} does not impose a temperature restriction on the processor. The temperature can exceed T_{qual} as long as the target MTTF value is maintained (because MTTF is also a function of voltage, frequency and utilization).

As can be seen, different frequencies are suggested by DRM and DTM. More significantly, at higher values

of T_{qual} and T_{max}, using the DTM suggested frequency would violate the system reliability requirement; while at lower values of T_{qual} and T_{max}, using the DRM suggested frequency would violate the system thermal requirement. This occurs because the slope of the DVS-Temp curve in the figure is generally steeper than the slope of the DVS-Rel curve. The reliability curve is less steep because of the exponential dependence of reliability on temperature. A small change in frequency creates a temperature change which is amplified exponentially in the reliability equation. This effect is further compounded by the large dependence of the TDDB MTTF on DVS voltage, as dictated by Equation 2.6. Finally, we can also see that the crossover point of the two curves is not fixed, and instead changes depending on the application. Hence, it is clear that the relationship between design for reliability and design for temperature is not obvious. Neither subsumes the other, and algorithms that jointly consider reliability and temperature (and energy) are important areas of future work.

5.6 Summary

In this chapter, we show how processors can be designed with less-than-worst-case temperature qualifications to conserve cost, without giving up performance. Specifically, we show that DRM provides a spectrum of cost-performance design points. This allows DRM to act as an enabling technology for reliability under-designed systems where traditional reliability qualification cost could be too prohibitive to stay on the required performance curve. Also, among the microarchitectural adaptations we studied, we show that DVS is the most beneficial for DRM.

Given that our algorithm adapts only once for an application, our DRM results represent a conservative bound on the performance and reliability benefits of DRM. A fine grained algorithm which adapts more often can potentially provide larger performance and reliability gains.

Finally, we see that the implications of adaptive control mechanisms for reliability are similar to earlier published dynamic energy management and thermal management methods. However, our experimental results clearly show that the tradeoffs involved in our new DRM methodology are significantly (if not fundamentally) different from those reported in prior adaptive work.

Chapter 6

Exploiting Structural Redundancy for Lifetime Reliability

Redundancy is a commonly used technique for reliability enhancement (for e.g., [16]). However, most previous work for lifetime reliability focused on redundancy at the processor granularity. Due to the large area overheads involved in duplicating entire processors, such redundancy does not provide a cost-effective reliability solution. Structural redundancy addresses some of these shortcomings of processor redundancy by incurring lower area overheads.

In this chapter, we examine two techniques that utilize structural redundancy for reliability, *structural duplication (SD)* and *graceful performance degradation (GPD)*. In Section 6.1, we describe SD and GPD. Our experimental methodology to evaluate the benefits of structural redundancy is presented in Section 6.2 and our results are presented in Section 6.3.

6.1 Structural Redundancy for Lifetime Reliability

In a reliability-constrained scenario, some performance and/or cost will have to be traded off for reliability. In this section, we examine methods by which structural redundancy can be used to enhance the processor so that it may efficiently exploit this performance and cost overhead. These enhancements to the processor allow run-time reconfiguration resulting in longer processor lifetimes.

6.1.1 Structural Duplication (SD)

In SD, extra structural redundancy is added over and above the required base processor resources during microarchitectural specification. The extra structures that are added are designated as *spares*, and are power gated and not used at the beginning of the processor's lifetime. During the course of the processor's life, if a structure with an available spare fails, the processor reconfigures and uses the spare structure. This extends the processor's life beyond the point when it would have normally failed, and instead, processor failure occurs only when a structure without a spare, or all available spares fail. It is important to note that spare structures are added over and above the required processor resources for optimal performance.

Most modern high-performance processors have enough redundancy to exploit all the available parallelism in common applications, resulting in very little performance benefit from the spares. As a result, the spares would be power gated to prevent any unnecessary wear-out, and would be powered on only when the original structure fails.

SD increases processor reliability without any loss of performance, relative to the base processor. However, due to increased die area, duplication adds a cost overhead to the base microarchitecture.

6.1.2 Graceful Performance Degradation (GPD)

GPD allows existing processor redundancy to be leveraged for lifetime enhancement without the addition of extra units. As mentioned, most modern high-performance microprocessors already use redundancy to exploit available parallelism in common applications. However, only a subset of these units is required for functional correctness. If a structure fails at run-time, a processor with GPD disables the failed structure and continues to function, thereby extending its lifetime beyond its original point of failure. Processor failure then occurs only when *all* redundant structures of any type fail.

Unlike SD, GPD does not add an area overhead to the base processor as no extra units are added. However, disabling redundant structures that fail lowers the processor's performance for the latter part of the processor's lifetime. Hence, the *guaranteed* performance of a processor with GPD is its performance in the fully degraded state. In this dissertation, we report GPD results for both guaranteed and actual performance.

6.1.3 Structural Duplication + Graceful Performance Degradation (SD+GPD)

We also examine architectures that use a combination of SD and GPD. Such processors can have spares for structures that are *also* allowed to degrade. Hence, after all available spares for a structure are used, the structure is allowed to degrade. Processor failure occurs only when all available spares fail **and** all available existing redundancy is used. This technique incurs both a performance overhead and a cost overhead. However, the benefits in reliability are larger.

Figure 6.1 illustrates the differences between the three techniques. Consider a base processor with two structures, A and B. Now, if the lifetimes of structures A and B for a random instance of the base processor are t_A and t_B, the base processor's lifetime in that instance is $MIN(t_A, t_B)$, as the first structure to fail would cause the processor to fail. Next, consider the base processor with SD, where another structure C is added as a spare to A and B. If the lifetime of C for the same instance of the processor is t_C, then the

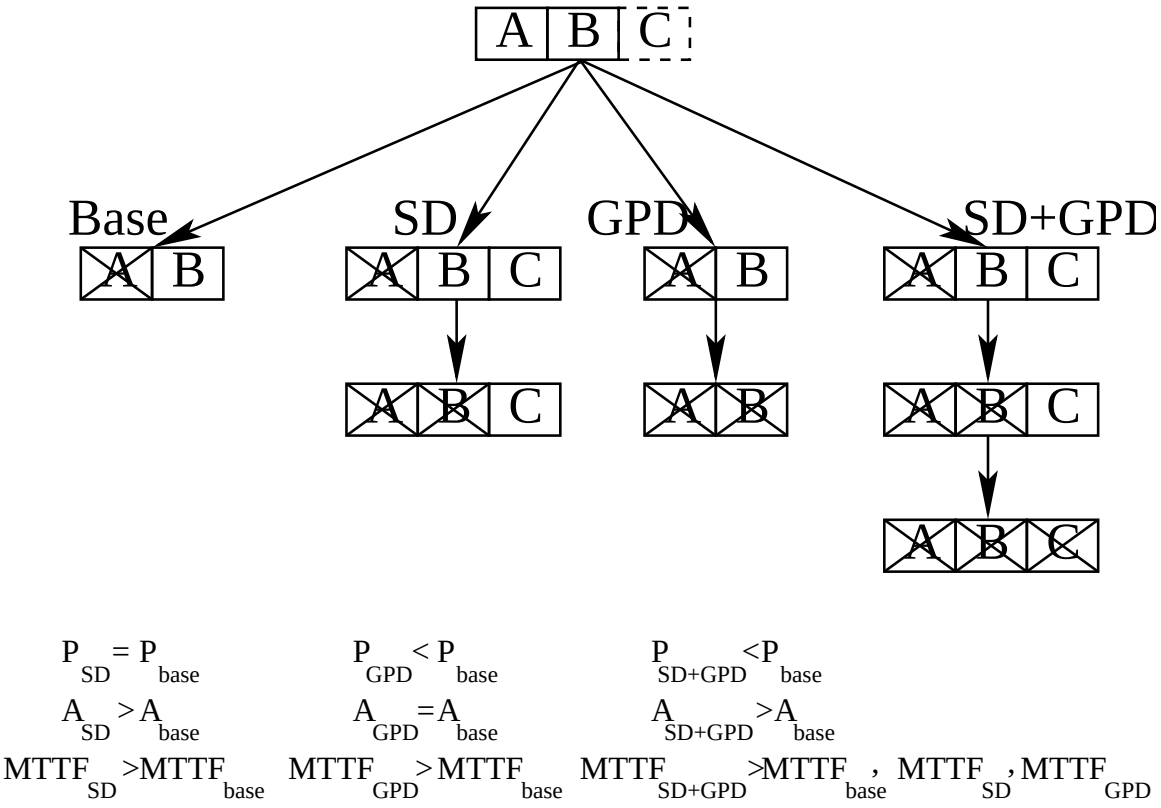


Figure 6.1: Steps to failure for a base processor, base processor with SD, with GPD, and with SD+GPD. The relationship between the performance (P), area (A), and MTTF of each of the processors is also given.

processor's lifetime would be $MIN((MIN(t_A, t_B)+t_C), MAX(t_A, t_B))$. Since the spare C is turned on only after A or B fails, C's lifetime is added to A or B. The processor fails only when either the spare or the remaining original structure fails.

Next, consider the base processor with GPD. The processor continues to function even if one of A or B were to fail. Hence, the lifetime of the processor with GPD is $MAX(t_A, t_B)$, since both structures have to fail for processor failure.

Finally, consider a processor with SD+GPD. A spare C is added for A and B. In addition, the processor requires all units to fail before total failure. In this case, the lifetime of the processor would be $MAX((MIN(t_A, t_B) + t_C), MAX(t_A, t_B))$. The spare C is used as soon as one of the original structures fails. The processor then fails only when both the spare and the remaining original structure fail.

6.1.4 Design Issues

A key requirement for SD, GPD, and SD+GPD is the ability of the processor to detect and disable structures that have failed during normal processor operation. Detecting errors is a critical issue for hard and soft error

tolerance, and there is significant ongoing work on detection techniques. However, much work still has to be done on the subject – currently, efficient detection techniques with high coverage for processor logic do not exist, and a detailed discussion of such functionality is beyond the scope of this paper. However, we expect detection and coverage issues to impact SD and GPD similarly, allowing a relative comparison of the techniques.

Also, both SD and GPD require additional hardware for detection and disabling/enabling of failed units. This extra hardware and resultant wiring will adversely affect processor power and performance (due to the larger communication distance between critical units). Accounting for these effects requires a detailed design for these techniques which is beyond the scope of this dissertation.

6.2 Structural Redundancy Experimental Methodology

6.2.1 Base Processor and Performance Simulation

Our base processor and performance simulation methodology is identical to that used in our scaling experiments (Section 3.3). The base processor we use for our simulations is a 65nm, out-of-order, 8-way superscalar processor, conceptually similar to a single core POWER4-like processor [43]. The 65 nm processor parameters were derived by scaling down parameters from the 180nm POWER4 processor. Table 6.1 in Section 3.3 summarizes the base processor modeled.

6.2.2 Power, Temperature, and Reliability Models

Our power and temperature simulation methodology is identical to that used in our scaling experiments (Section 3.3). For processor power dissipation, we use the PowerTimer toolset [45]. For temperature simulation, we use the HotSpot tool [11]. We also use the leakage power methodology described in Section 3.3.2.

We use RAMP 2.0 for our reliability measurements. For a simulated application, based on temperature estimates from HotSpot and power estimates from PowerTimer sampled at a granularity of 1 μsecond, RAMP 2.0 calculates an MTTF estimate for each structure and failure mechanism on the processor. The Monte-Carlo simulation method is then used to determine the MTTF of the processor. Given that we only compare relative increases and decreases in reliability, we do not need to use T_{qual} and other qualification proxies.

In order to quantify the difference between using lognormal and exponential distributions for failure mechanisms, we repeat some of our experiments with RAMP 1.0. These results are described in Section 6.3.7.

Technology Parameters	
Process technology	65 nm
V_{dd}	1.0 V
Processor frequency	2.0 GHz
Processor size (not including L2)	11.52 mm^2 (3.6 mm x 3.2 mm)
Leakage power density at $383K$	0.60 W/mm^2
Base Processor Parameters	
Fetch/finish rate	8 per cycle
Retirement rate	1 dispatch-group (=5, max) per cycle
Functional units	2 Int, 2 FP, 2 Load-Store
	1 Branch, 1 LCR
Integer FU latencies	1/7/35 add/multiply/divide (pipelined)
FP FU latencies	4 default, 12 div. (pipelined)
Reorder Buffer size	150
Register file size	120 integer, 96 FP
Memory queue size	32 entries
Base Memory Hierarchy Parameters	
L1 (Data)	32KB
L1 (Instr)	32KB
L2 (Unified)	2MB

Table 6.1: Base 65 nm POWER4-like processor used for structural redundancy experiments

6.2.3 Die Cost Model

In order to evaluate the cost impact of area increases imposed by structural duplication, we use the Hennessy-Patterson die cost model [61]. The cost, C, of a die of area, A is:

$$C \propto \frac{1}{(\frac{\pi r_{wafer}^2}{A} - \frac{2\pi r_{wafer}}{\sqrt{2A}})} \times (1 + \frac{DA}{\alpha})^{\alpha} \tag{6.1}$$

where r_{wafer} is the wafer radius, D is the defects per unit area during manufacture of the wafer, and α is a parameter that corresponds inversely to the number of masking levels. We assume a $300mm$ wafer process, $D = 0.6$ per square centimeter, and $\alpha = 4.0$ [61]. In our experiments, we normalize our base processor cost to 1.0 (for a base area of $11.52mm^2$).

6.2.4 Workload Description

Our experimental results are based on an evaluation of 16 SPEC2000 benchmarks (8 SpecInt + 8 SpecFP) listed in Table 6.2. The SPEC2000 trace repository used in this study was generated using the Aria trace facility in the MET toolkit [48], and was generated using the full reference input set. Sampling was used to limit the trace length to 100 million instructions per program. The sampled traces have been validated with the original full traces for accuracy and correct representation [49].

Type	Application	Max. Temp. (K)
Spec2000	ammp	341.27
Float	sixtrack	342.76
	applu	343.82
	mgrid	345.63
	mesa	345.87
	facerec	346.52
	apsi	348.49
	wupwise	348.56
SpecFP average		**345.36**
Spec2000	vpr	341.40
Int	twolf	343.22
	bzip2	342.52
	gzip	343.49
	perlbmk	347.13
	gcc	348.22
	gap	348.93
	crafty	349.55
SpecInt average		**345.52**

Table 6.2: Workload used for structural redundancy experiments. The maximum temperature seen with each application is also given.

6.2.5 Processor Configurations Evaluated

The base 65nm POWER4-like processor evaluated has a total area of $11.52mm^2$. Figure 6.2 shows the floorplan for the processor. The chip is divided into 7 distinct structures: floating point unit (FPU), fixed point unit (FXU), instruction decode unit (IDU), instruction scheduling unit (ISU), load store unit (LSU), instruction fetch unit (IFU), and branch prediction unit (BXU). Again, it is important to note that we do not calculate the reliability of the L2 cache.

SD Configurations

To limit our configuration space, we do not allow all the structures on chip to be replicated individually for SD. Instead, we clubbed the processor's structures into 5 logical groups that can be replicated for spares – FPU, FXU, BXU+IFU, LSU, IDU+ISU. Table 6.3 summarizes these groups and the area overhead imposed on the processor by replicating each group. With these 5 groups, based on whether a group is replicated or not in the processor, we create 32 (2^5) SD configurations. If more than one group is replicated, the area overhead for that processor is the sum of the areas of the replicated groups.

For the base processor without SD, the average failure rate across all applications for each structure is initialized to a value proportional to the structure's area. The failure rate of a duplicate structure is initialized to be equal to the original structure.

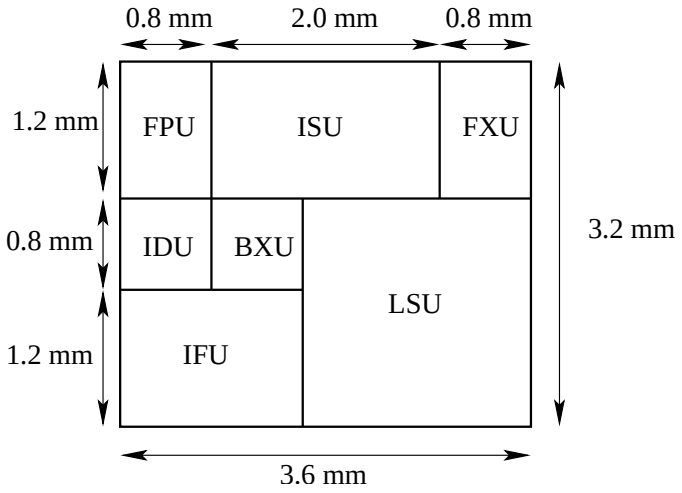


Figure 6.2: Chip layout for the 65 nm POWER4-like processor used in our structural redundancy experiments.

Group	Units	Area mm^2	Original Configuration	Degraded Configuration
1	FPU	0.96	2 float units + 96 float regs	1 float unit + 48 float regs
2	FXU	0.96	2 int units + 120 int regs	1 int unit + 60 int regs
3	BXU+IFU	2.56	16K BHT + 32KB ICache	8K BHT + 16KB ICache
4	LSU	4.0	2 load queues + 32KB DCache	1 load queue + 16KB DCache
5	IDU+ISU	3.04	N/A	N/A

Table 6.3: Groups replicated in SD and allowed to degrade in GPD. The IDU+ISU is not allowed to degrade. The areas of each group for SD and the structures in the original and degraded group for GPD are also given.

GPD Configurations

Like SD, we limit our configuration space in GPD by not allowing every structure to degrade individually. Instead, the structures are grouped into 4 logical groups that can degrade – FPU, FXU, BXU+IFU, LSU. Unlike structural duplication, we do not allow the IDU+ISU to degrade, as there is no simple method to halve the logic in these units. Each group can be in one of two states, full size or degraded to half size. That is, the group can be fully functional, or if a failure occurs in a structure, the half of the group that contains the failure would be shut down (although many structures like the caches can degrade to levels other than half size, we do not study them to limit the configuration space).

With these 4 groups, based on whether a group is allowed to degrade to half size or not, 16 (2^4) configurations including the base can be created. Table 6.3 shows the configuration of the groups before and after degradation.

For structures that are allowed to degrade, we calculate the failure rate of each half of the structure seperately. In the base case, the failure rates of each half is calculated such that the failure rate of the whole

structure (the two halves in a series configuration) is the same whether the structure is allowed to degrade or not. When using lognormal failure distributions, the base case failure rate for each half is calculated using a Monte-Carlo simulation.

SD+GPD Configurations

SD and GPD can act orthogonally on the processor (a duplicated structure can also degrade). Hence, the number of configurations for SD+GPD is the cross product of the number of SD configurations and GPD configurations ($2^5 \times 2^4 = 512$).

6.3 Structural Redundancy Results

6.3.1 SD Results

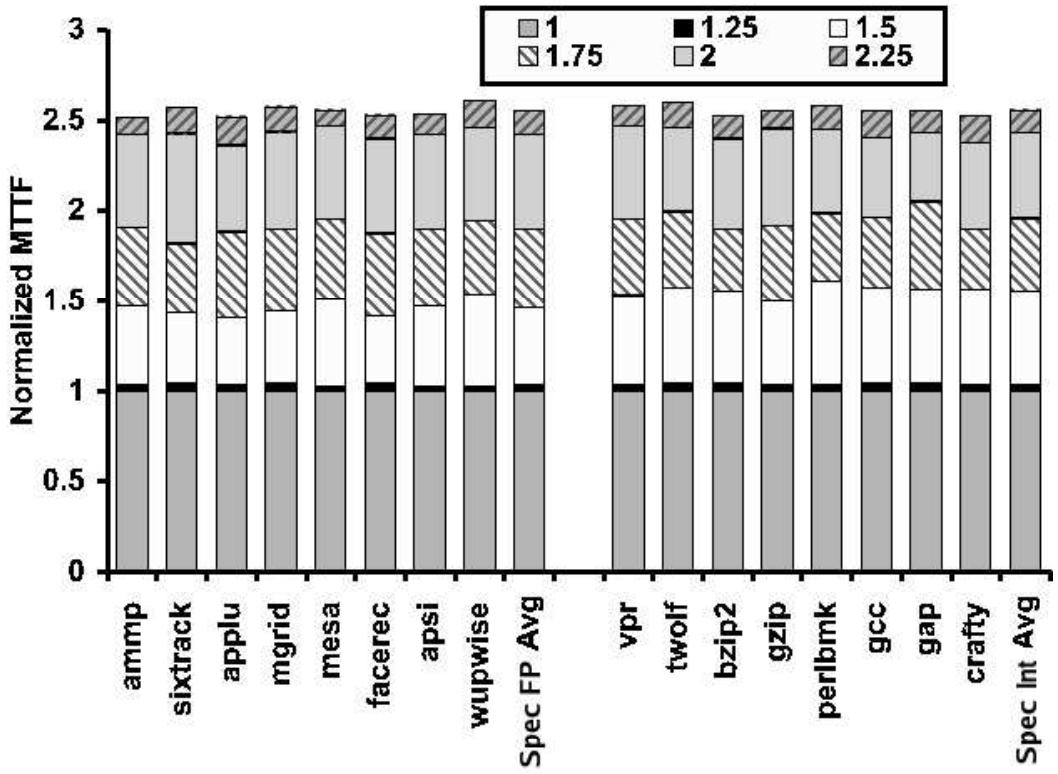


Figure 6.3: Reliability benefit from SD for different costs. The vertical axis shows normalized MTTF, with the MTTF of the application on the base processor normalized to 1.0 (the bottom segment of each bar). Each additional segment in the bars represents the normalized gain in MTTF from moving to higher costs.

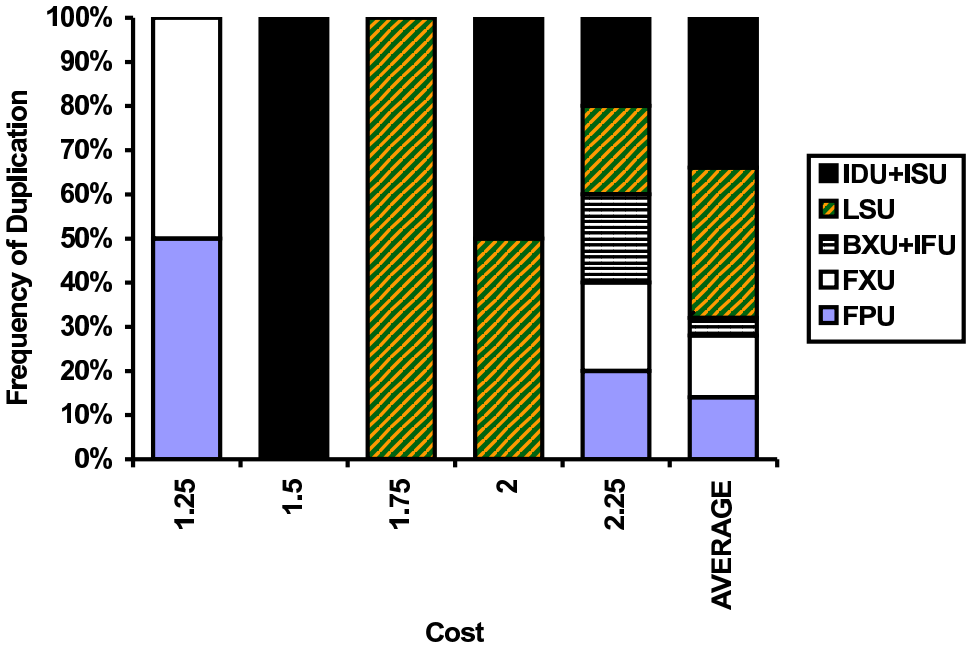


Figure 6.4: Fraction of applications for which different groups of structures are chosen for duplication with SD, for different costs. The average frequency across all costs is also given.

Figure 6.3 shows the SD reliability benefit for various cost points for each of our applications, and also the average for all SpecFP and SpecInt applications. The vertical axis shows normalized MTTF. The results are presented in a stacked-bar format. The MTTF of each application on the base processor (which has a cost of 1.0), is the lowest segment in each bar, and is normalized to 1.0. Each additional segment in the bars represents the incremental normalized MTTF benefit obtained from moving to higher costs. For each segment, we selected the SD configuration that had the highest MTTF among the configurations that satisfied the cost requirement. Figure 6.4 shows the fraction of applications for which different groups of structures are chosen for duplication with SD, for different costs. In addition, the average frequency across all costs is also shown.

As seen in Figure 6.3, SD provides significant reliability benefit, particularly for higher cost values. At a cost of 2.25 times the base cost, SD provides an average MTTF 2.53 times better than base MTTF. However, at a cost of 1.25 times the base cost, the MTTF is only 4% greater than base MTTF. These results can be understood with Figure 6.4 – for a cost of 1.25 times the base cost, only the FPU and FXU are chosen for duplication. Although the FPU and FXU do not provide large reliability benefit, they are the only structures that have areas small enough to satisfy the cost limit at 1.25 times the base cost (Table 6.3). As we move to higher cost points (left to right in Figure 6.4), larger structures which have higher failure rates can be duplicated, resulting in significant impact on reliability. At 1.5 times the base cost, the IDU+ISU can be

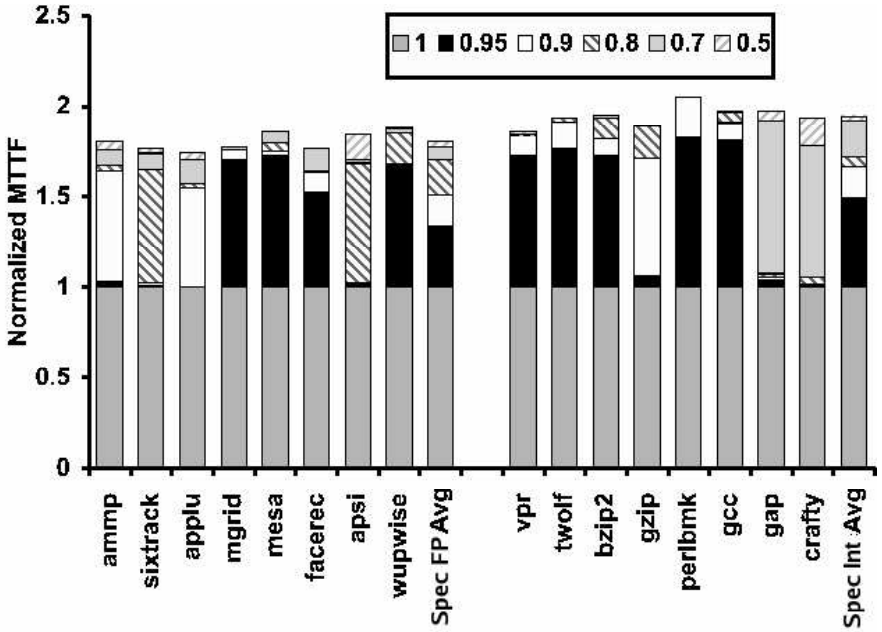


Figure 6.5: Reliability benefit from GPD for different *guaranteed* performance levels. The vertical axis shows normalized MTTF, with the lowest segment in each bar representing the normalized base MTTF of the application (performance of 1.0). Each additional segment shows the incremental MTTF benefit from moving to lower performance values.

duplicated, and at 1.75 times the base cost, the LSU can be duplicated. For points beyond 1.75 times the base cost, combinations of structures are used in SD.

6.3.2 GPD Results

Figures 6.5 and 6.6 show the GPD reliability benefit for various performance levels for each of our applications, and also the average for all SpecFP and SpecInt applications. Like Figure 6.3, the vertical axis represents normalized MTTF. The MTTF of each application on the base processor (which has a performance of 1.0), is the lowest segment in each bar, and is normalized to 1.0. Each additional segment shows the incremental benefit from moving to lower performance. Figure 6.5 shows *guaranteed* performance values, while Figure 6.6 shows *actual* performance values. Unlike SD, where the cost overhead of a configuration applies for the entire lifetime of the processor, the performance degradation in GPD is not seen for the entire lifetime of the processor. At the beginning of the processor's lifetime, it will run at full performance. The degraded performance level is encountered only after one or more structures on chip fail. Due to the statistical nature of wear-out failures, for a given processor, no performance greater than the degraded value

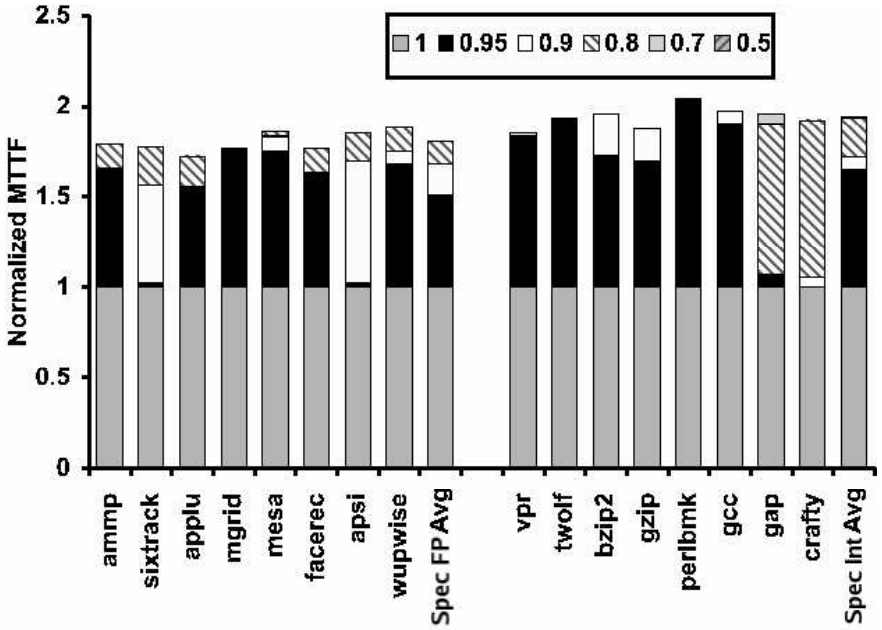


Figure 6.6: Reliability benefit from GPD for different *actual* performance levels. The vertical axis shows normalized MTTF, with the lowest segment in each bar representing the normalized base MTTF of the application (performance of 1.0). Each additional segment shows the incremental MTTF benefit from moving to lower performance values.

can be *guaranteed* (in a random batch of processors, some might have structures failing immediately). Figure 6.5 presents GPD results for this lowest guaranteed performance level. However, most processors will have a higher *actual* performance (which is the time-weighted average of all the IPCs seen during the lifetime of the processor). These actual performance values are reported in Figure 6.6. These actual performance numbers indicate that in many scenarios, the user will achieve better than guaranteed performance. For each performance value (guaranteed or actual), we identified the GPD configuration which had the highest MTTF among the configurations which satisfied the performance requirement.

As can be seen, GPD results in significant MTTF benefit, particularly for small performance overheads. A guaranteed loss of 5% in performance (performance value of 0.95 in Figure 6.5) provides an average MTTF 1.42 times better than base MTTF. An actual loss of 5% in performance (performance value of 0.95 in Figure 6.6) provides an average MTTF 1.61 times better than base MTTF. As we move to lower performance values, the incremental MTTF benefit from GPD reduces on average. Also, as expected, much smaller decreases in actual performance provide the same reliability benefit as larger decreases in guaranteed performance.

The results in Figures 6.5 and 6.6 show that processor resources in current high performance microprocessors likely exceed the requirements for performance and functionality of many applications. Most applications do not regularly use all the extra replicated units. As a result, when a failure occurs in one of these relatively unused structures, the processor can degrade to half the structure's size without a significant loss in performance, but with large reliability benefit. Once all the structures that are not used have degraded, further performance reductions result in much smaller reliability benefit.

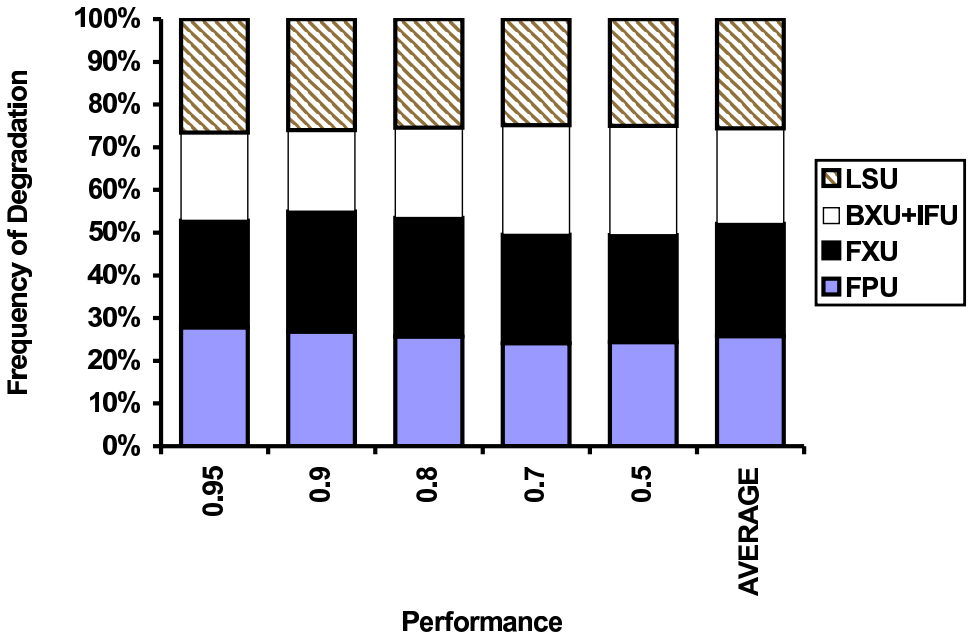


Figure 6.7: Fraction of applications for which different groups of structures are chosen for degradation with GPD, for different guaaranteed performance levels. The average frequency across all performance levels is also given.

As in Figure 6.4, Figure 6.7 shows the fraction of applications for which different groups of structures are chosen for degradation with GPD, for different guaranteed performance levels. The average frequency across all performance levels is also given. Unlike SD where different structures were chosen for duplication at different costs, all structures are chosen with nearly the same frequency for degradation in GPD. For higher performance values (left side of Figure 6.7), the frequencies are similar because different applications in our workload rely on different processor structures for performance. This shows that no structure in the fully functional state is performance critical for all applications. For lower performance values (right side of Figure 6.7), the frequencies are similar because most applications have reached the fully degraded state, shutting down half of every structure.

6.3.3 SD+GPD Results

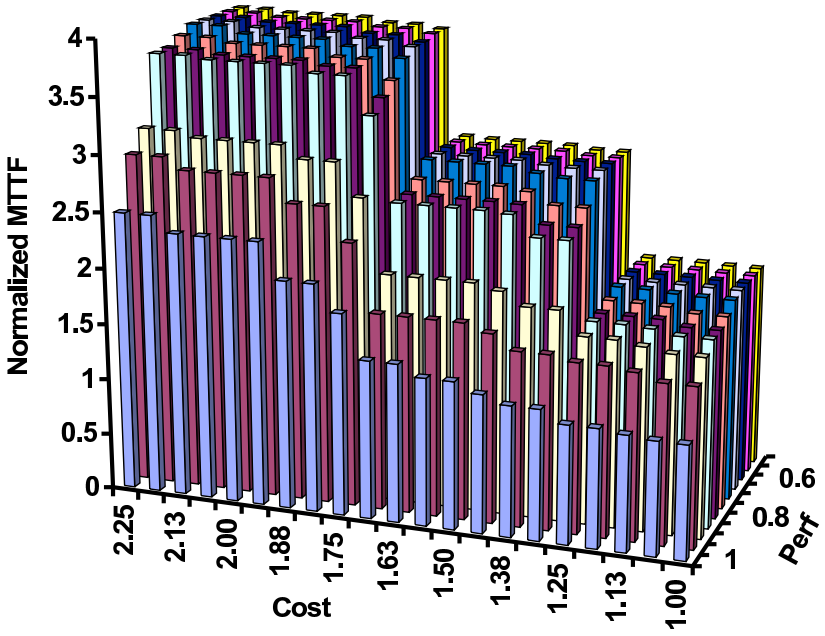


Figure 6.8: Highest SD+GPD MTTF (averaged across all our applications) possible for each cost and *guaranteed* performance constraint. Each MTTF value (represented by the height of the bars) is the average normalized MTTF across all applications, where the average MTTF at a performance of 1.0 and a cost of 1.0 is normalized to 1.0

Figures 6.8 and 6.9 show the reliability benefit from combining SD and GPD. The figures show the highest MTTF possible for each cost and performance constraint, averaged across all applications. That is, for each point with cost=C and performance=P, we report the highest MTTF (averaged across all applications) among all the SD+GPD configurations with cost $\leq$ C and performance $\geq$ P. Each MTTF value (represented by the height of the bars) is the average normalized MTTF across all applications, where the average MTTF at a performance of 1.0 and a cost of 1.0 (no SD or GPD) is normalized to 1.0. In the figure, when performance is 1.0, the values show average MTTF using only SD. When cost is 1.0, the values show average MTTF using only GPD. Every other point in the figures shows average MTTF for some degraded performance level and cost value (SD+GPD). Like Figures 6.5 and 6.6, Figures 6.8 and 6.9 represent *guaranteed* and *actual* performance levels, respectively.

As can be seen, SD+GPD (points with both a performance loss and cost increase) provides larger MTTF benefit than SD or GPD alone. In particular, at the extreme point, a guaranteed loss of 50% or an actual

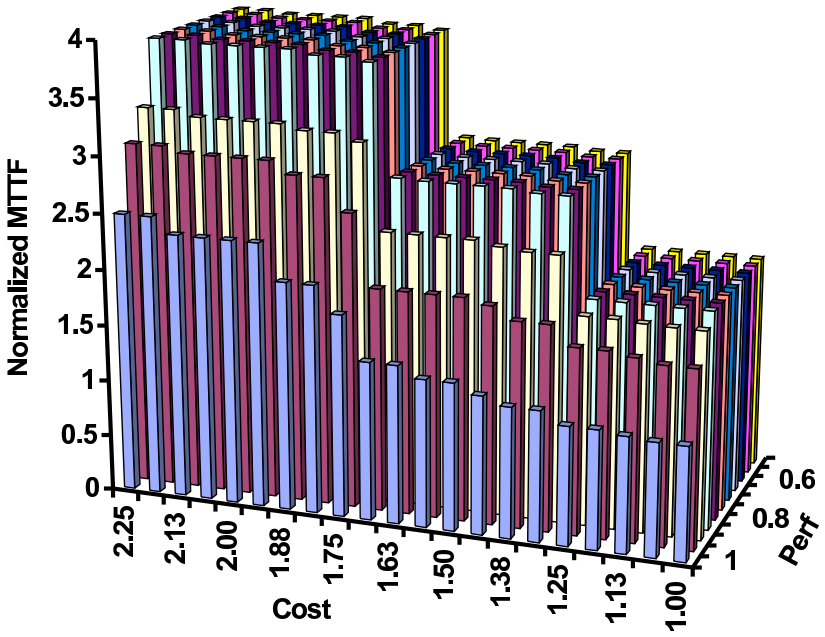


Figure 6.9: Highest SD+GPD MTTF (averaged across all our applications) possible for each cost and *actual* performance constraint. Each MTTF value (represented by the height of the bars) is the average normalized MTTF across all applications, where the average MTTF at a performance of 1.0 and a cost of 1.0 is normalized to 1.0

loss of 15% in performance (performance value of 0.5 in Figure 6.8 and 0.85 in Figure 6.9), coupled with a cost 2.25 times the base cost, provides an average MTTF 3.89 times better than base MTTF. As discussed in Section 6.3.1, SD provides low average reliability benefit at very low cost values, but large benefits at higher cost values, for any given performance level. Similarly, as discussed in Section 6.3.2, GPD provides a larger incremental reliability benefit for smaller performance degradations (larger performance values), for any given cost. Also, the overall increase from SD is higher than that for GPD. Finally, as expected, much smaller decreases in actual performance provide the same reliability benefit as larger decreases in guaranteed performance. As explained earlier, this is due to the processor running at full performance at the beginning of its lifetime.

6.3.4 Comparison of SD, GPD, and SD+GPD using Performance/Cost

In order to understand performance and cost tradeoffs simultaneously, we use the ratio of performance and cost ($\frac{P}{C}$), a standard industrial metric, to evaluate SD, GPD, and SD+GPD. The normalized $\frac{P}{C}$ for all our

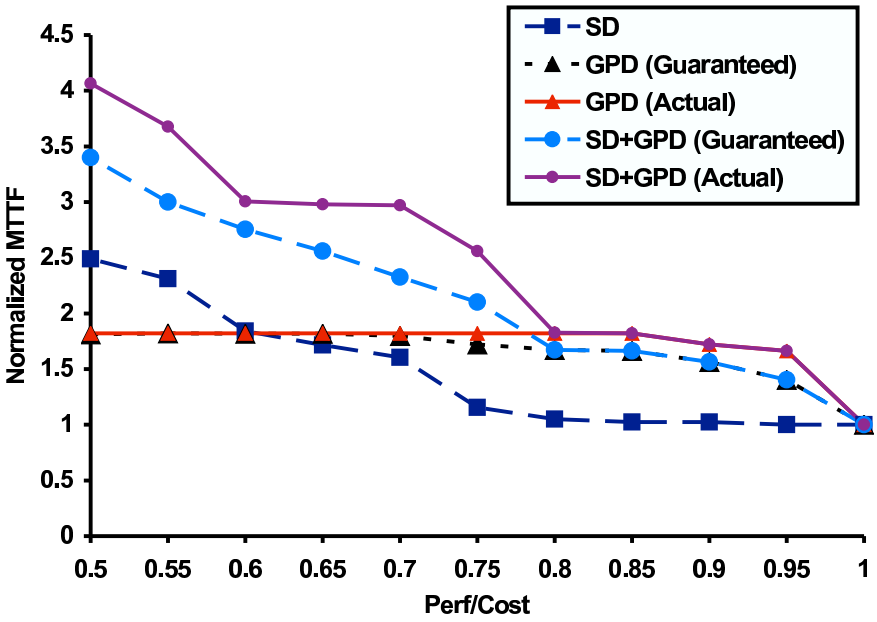


Figure 6.10: Average normalized MTTF benefit versus $\frac{P}{C}$ for SD, GPD, and SD+GPD across all applications. For GPD and SD+GPD, both *guaranteed* and *actual* performance values are given.

applications on the base processor is 1.0. In SD, cost will increase, leading to $\frac{P}{C}$ values lower than 1.0. In GPD, performance will decrease, leading to $\frac{P}{C}$ values lower than 1.0, and in SD+GPD, both increases in cost and decreases in performance lower the value of $\frac{P}{C}$. Figure 6.10 shows the average MTTF benefit across all our applications from each of the three techniques for a range of $\frac{P}{C}$ values. The vertical axis represents normalized MTTF. The horizontal axis represents different $\frac{P}{C}$ design points. For both GPD and SD+GPD, both guaranteed and actual performance levels are given.

The results in Figure 6.10 clearly reflect the trends seen in Figures 6.3, 6.5, 6.6, 6.8, and 6.9. At high $\frac{P}{C}$ values (low performance or cost overhead), GPD provides much more benefit than SD. However, the benefit from GPD tapers off as we move to lower values of $\frac{P}{C}$. On the other hand, SD provides much more MTTF benefit at lower $\frac{P}{C}$ values, and overtakes GPD. The combination of both techniques always provides the highest MTTF benefit. This is intuitive because SD+GPD can choose any configuration SD or GPD can choose, in addition to the cross product of the two. However, SD+GPD chooses the same configurations as GPD chooses at high values of $\frac{P}{C}$. Finally, since processors run at full performance at the beginning of their lifetime, the same MTTF benefit for GPD (Actual) and SD+GPD (Actual) comes at higher $\frac{P}{C}$ values than GPD (Guaranteed) and SD+GPD (Guaranteed).

6.3.5 Using Redundancy to Mitigate the Effect of Scaling on Lifetime Reliability

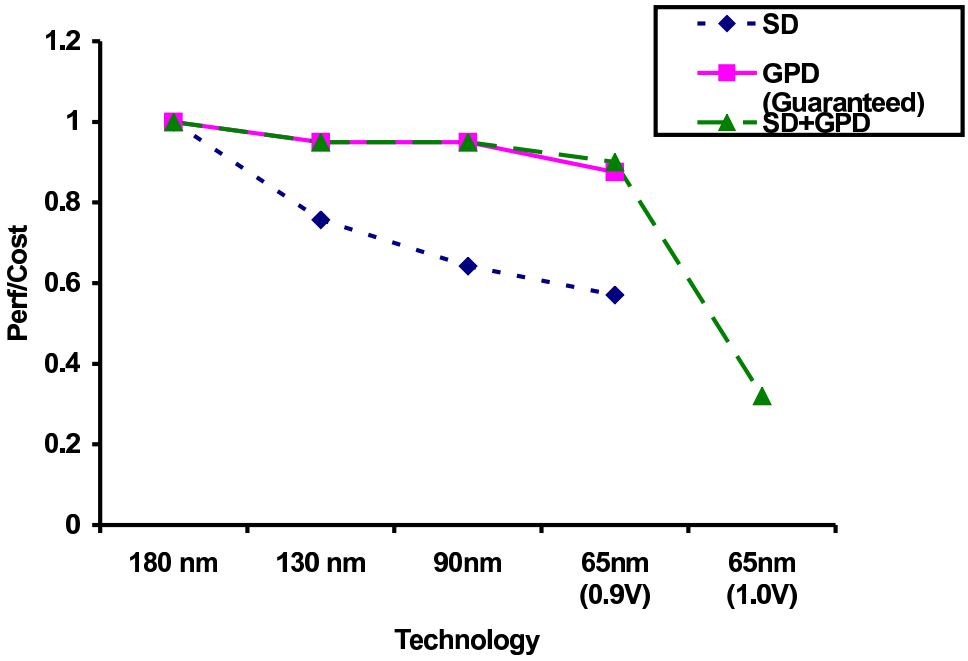


Figure 6.11: Average $\frac{P}{C}$ across all applications required by SD, GPD, and SD+GPD to achieve target reliability for different technology points. For GPD and SD+GPD, only *guaranteed* performance values are given.

In this section, we analyze the use of structural redundancy to mitigate the effects of technology scaling on lifetime reliability. As discussed in Chapter 3, technology scaling is drastically reducing lifetime reliability. Lifetime reliability enhancement techniques like structural redundancy will be required to maintain target reliability requirements.

Figure 6.11 shows the decrease in the ratio of performance and cost ($\frac{P}{C}$) required by SD, GPD, and SD+GPD, to maintain reliability when scaling from 180nm to 65nm. The figure shows average $\frac{P}{C}$ across all 16 applications in our benchmark suite. At 180nm, the average MTTF across all applications is normalized to 1.0 and is the target reliability. The vertical axis shows the $\frac{P}{C}$ values required to maintain target reliability for the technology generations on the horizontal axis. At technology generations past 180nm, target reliability can be achieved only with lower $\frac{P}{C}$ values. This decrease in $\frac{P}{C}$ comes either from a loss in performance, an increase in cost, or a combination of both. For GPD and SD+GPD, only guaranteed performance values are given.

As can be seen in Figure 6.11, a significant decrease in $\frac{P}{C}$ is required to maintain target reliability. At 65nm (0.9V), SD requires a decrease in $\frac{P}{C}$ to 0.62, GPD requires a decrease in $\frac{P}{C}$ to 0.87, and SD+GPD

requires a decrease in $\frac{P}{C}$ to 0.89. Only SD+GPD can achieve the target reliability at 65nm (1.0V), requiring a decrease in $\frac{P}{C}$ to 0.36. At 130nm, 90nm, and 65nm (0.9V), the decrease in MTTF due to scaling is less than 40%. As a result, GPD achieves the target reliability at these technology points more efficiently than SD. Since SD+GPD can choose any configuration SD or GPD can choose, it achieves target reliability more efficiently than either technique individually.

6.3.6 Discussion

The above results present some clear guidelines for the use of structural redundancy for reliability:

- Due to the high level of redundancy already built into current high-performance processors to exploit application parallelism, GPD is an attractive technique for performance-effective reliability benefit. This is particularly true for scenarios where only limited performance or area resources can be diverted to reliability because of cost issues. However, the benefit from GPD is limited – once extra redundant units degrade, the remaining units are essential for processor performance and functionality and cannot degrade further.

- SD is an attractive option when larger cost overheads are available, because large critical structures on chip can be duplicated. Unlike GPD, the benefit from SD does not taper off. Hence, in scenarios where reliability is more important than cost, SD is the more beneficial technique.

- Finally, the combination of SD and GPD, SD+GPD, always provides the highest reliability increases because it can exploit the benefits of both SD and GPD.

6.3.7 Comparison of Structural Redundancy Techniques on RAMP 2.0 and RAMP 1.0

As explained in Section 3.4.4, the SOFR model is commonly used to combine failure rates. However, this leads to some inaccuracies due to the use of exponential failure distributions. In order to quantify this inaccuracy, we repeated some of our structural redundancy experiments using RAMP 1.0, allowing us to compare those results with our results using RAMP 2.0.

Figure 6.12 shows the average normalized MTTF benefit versus $\frac{P}{C}$ for SD, GPD and SD+GPD across all applications, using RAMP 2.0, which uses lognormal failure distributions (labeled as LOG), and RAMP 1.0, which uses exponential failure distributions (labeled as EXP). The GPD results are for guaranteed performance values only. The RAMP 1.0 results (exponential results) are shown with dotted lines. The

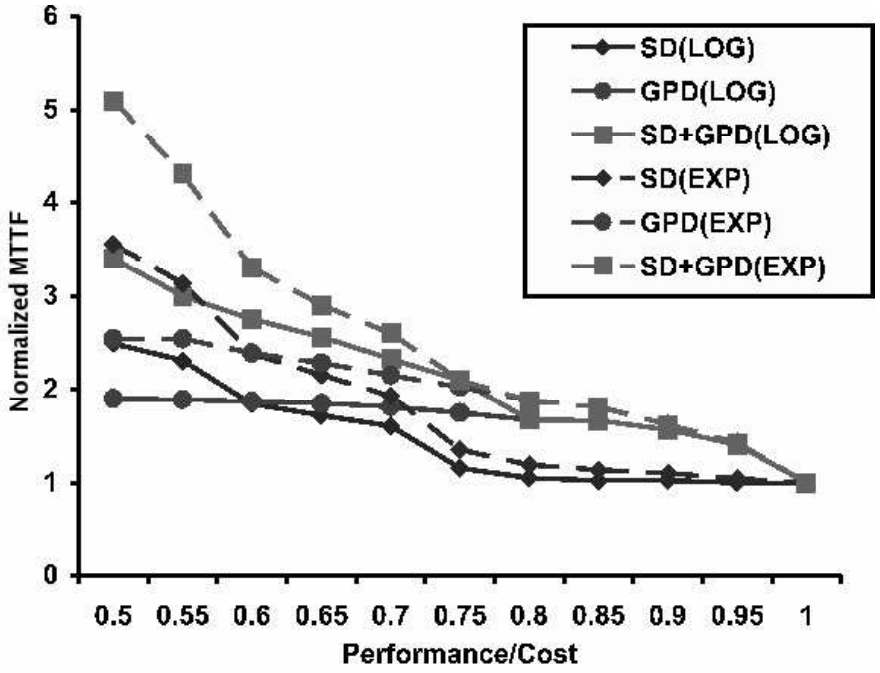


Figure 6.12: Average normalized MTTF benefit versus $\frac{P}{C}$ for SD, GPD, and SD+GPD across all applications, using RAMP 2.0, which uses lognormal failure distributions (labeled as LOG), and RAMP 1.0, which uses exponential failure distributions (labeled as EXP). The GPD results are for guaranteed performance values only.

normalized $\frac{P}{C}$ for all of the curves is 1.0 on the base processor. As cost increases or performance decreases, the $\frac{P}{C}$ values drop.

As can be seen, the lognormal and exponential results diverge for all values of $\frac{P}{C}$ less than 1.0, with the lognormal results being more conservative than the exponential results. More importantly, as me move to points with more structural redundancy (lower $\frac{P}{C}$), the magnitude of the difference between lognormal and exponential results increases. Specifically, at a $\frac{P}{C}$ value of 0.5, the exponential SD results are 39% higher than the lognormal SD results, the exponential GPD results are 36% higher than the lognormal GPD results, and the exponential SD+GPD results are 44% higher than the lognormal SD+GPD results.

As in Section 3.4.4, these results clearly highlight the importance of using accurate failure distributions. Although RAMP 1.0 is easier to use due to the simplicity of the SOFR model, its use can lead to inaccurate results. This is particularly true because RAMP 1.0 provides *less conservative* results. Less conservative results can lead to unintentional under-designing of processors from a reliability perspective.

6.4 Summary

Aggressive scaling of CMOS devices is accelerating the onset of wear-out related lifetime reliability problems. This implies that future processors will be designed in reliability-constrained environments where some processor performance or die cost will have to be sacrificed for reliability. In this chapter, we examined the efficient usage of these performance and cost tradeoffs through structural redundancy.

Specifically, we evaluated two techniques, structural duplication (SD) and graceful performance degradation (GPD). In SD, extra or spare structures are added to the processor during microarchitectural definition. Spare structures can be turned on during the processor's lifetime when the original structure fails, thereby extending processor lifetime. Although SD results in no performance loss relative to the base processor, the spare structures incur an area and resultant cost overhead for the processor. GPD, on the other hand, does not require extra structures to be added to the base processor. Instead, GPD exploits existing structural redundancy on chip for reliability. If a redundant structure fails in a processor with GPD, the structure can be shut down and the processor would still be functional. This however, comes at a performance loss to the processor.

Our analysis provides clear guidelines for the use of SD and GPD for reliability enhancement. If only limited performance or area resources can be diverted to reliability, GPD presents a more attractive option for reliability enhancement for our systems. On the other hand, in scenarios where reliability is more important than performance or cost, SD is the more beneficial technique. A combination of SD and GPD (SD+GPD) provides the highest reliability increases for the lowest performance and cost overheads because it can exploit the benefits of both techniques.

Chapter 7

Related Work

In this chapter, we discuss work related to the core contributions of this thesis.

7.1 Previous Work on Reliability Modeling

7.1.1 Device-Level Reliability Models

There is an extensive body of work in the device design and manufacturing community on understanding and modeling hard failure mechanisms. Electromigration is one of the best understood failure mechanisms and much research has been performed on techniques to mitigate the effects of electromigration [62, 18, 21, 22, 23, 24, 25, 26, 27]. Stress migration is similar in effect to electromigration. The exact mechanisms behind stress migration are still not completely understood and research is ongoing on the subject [18, 21]. Time-dependent dielectric breakdown (TDDB) is another extremely important failure mechanism and has also been extensively studied [18, 63, 8, 64, 65, 5, 29]. Thermal cycling and thermal shock are effects that are most noticeable in the package [18, 66]. The problems due to thermal cycling are exacerbated with the introduction of low-k dielectrics which tend to be more brittle [67]. Finally, negative-bias temperature instability (NBTI) has recently emerged as another critical hard error failure mechanism [19]. Although these device-level models form the core of RAMP, most of this work looks at different failure mechanisms separately – it does not attempt to unify the mechanisms to form a system wide failure model. Further, it is also application oblivious.

7.1.2 Microarchitectural Lifetime Reliability Models

Recently, Lu et al. proposed a physics based microarchitectural model for electromigration that modeled IC interconnects under dynamic thermal and electrical current stresses [68]. Their model extends Black's original model for elecromigration [18] by accounting for temporal variations in temperature in the interconnects. Additionally, application characteristics were taken into account in their reliability calculations. In [69], Lu et al. use their electromigration model to evaluate a technique called reliability banking. Reliability banking

is similar in effect to DRM, where dynamic voltage scaling (DVS) is used to accumulate or bank reliability during low utilization phases, for use in high utilization phases, thereby improving processor performance during dynamic reliability control.

Although Lu et al. account for application behavior in their reliability calculations, they only model electromigration. Due the large voltage dependence of gate-oxide breakdown, not modeling the effect of DVS on this failure mechanism can lead to large errors in their reliability analysis. In addition, they examine reliability at the granularity of the entire processor, and not at a structure or finer level. Finally, they assume a constant worst-case value for interconnect current density and do not account for variations in current density with application behavior.

7.1.3 Soft Error Related Work

As mentioned, most recent work on architectural awareness of reliability has not addressed hard errors and instead concentrated on soft errors. The semiconductor research and development community has put forth a great deal of effort in building models and estimators of soft error rates for future generation CMOS devices [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84]. These models range from physics-based models to microarchitectural models. Some recent work has focused on understanding soft error propagation in high-performance microprocessors [85, 86, 84] in order to identify vulnerable logic blocks that require additional soft-error hardening. This work has also led to techniques that can mitigate the effects of soft errors. Although some of these soft error correction schemes can be used to increase tolerance to hard errors, the bulk of this research will not impact the hard failure rate.

7.2 Hardware Techniques for Reliability Enhancement

Current techniques for enhancing hard failure reliability focus on fault-tolerant computing methods like redundancy [17, 16] and efficient failure recovery methods [87]. These techniques are typically used in server class processors.

Redundancy is also utilized in array structures for lifetime enhancement. Many current memory systems utilize built-in self test (BIST) and built-in self repair (BISR) techniques to handle defective memory elements [88]. Recently, Bowers et al. proposed self-repairing array structures (SRAS), a technique to mask hard faults in array structures like the reorder buffer and branch history table [89]. These techniques are limited to array structures and replicate at the granularity of individual array entries.

There exists an extensive body of work on microprocessor yield enhancement [90, 91, 89]. These tech-

niques are not run-time techniques and are instead used during processor testing. By detecting and disabling memory structure elements, these techniques increase processor yield [90]. Shivakumar et al. extend this concept and propose disabling defective redundant microarchitectural structures including functional units during testing to improve yield [91]. They define three types of processor redundancy that can be used for yield enhancement – component level redundancy in structures like the functional units, array redundancy in memory structures, and dynamic queue redundancy in structures like the register file.

7.3 Related Work on Architectural Adaptation

There has been much work on architectural techniques for energy and thermal control. Many of the core concepts behind these techniques can be adapted for reliability control.

[92, 93, 94, 95, 96] propose algorithms that use DVS for energy control. Several researchers have proposed architecture adaptations for energy savings including speculation control [97], changing instruction window size [98, 99, 100, 101, 102], changing the active functional units [103, 104], changing issue width [103, 101], gating instruction fetch [105], and shutting off parts of the cache [106, 99]. [56] and [53] propose algorithms that use DVS and architectural adaptation in concert for energy savings. Similarly, hardware adaptation has been used for thermal control in the form of dynamic thermal management (DTM) [11, 52, 55, 107].

Chapter 8

Conclusions and Future Directions

This chapter summarizes the analysis and insights of this dissertation and discusses future directions for research motivated by the results of this dissertation.

8.1 Conclusions

In this era of power-constrained design, the effect of escalating on-chip temperatures on chip reliability is of increasing concern to processor and system developers. The effects of technology scaling in the deep sub-micron range are adding to this reliability concern.

In this dissertation, we address lifetime reliability issues from a microarchitectural perspective. Traditionally, microarchitects have treated the issue of processor lifetime reliability as a manufacturing problem, best left to be handled by device and process engineers. However, we observe that the microarchitecture's ability to track application behavior can potentially be leveraged to the benefit of reliability qualification, enabling reduced reliability design costs, increased processor yield, and/or increased performance. In this dissertation, we design tools and techniques for microarchitectural lifetime reliability awareness and enhancement.

To address the need for microarchitectural lifetime reliability analysis tools, we present a new modeling methodology, called RAMP, for enabling early-stage power/performance/reliability tradeoffs. Driven by technology, packaging, and chip floorplan parameters, this methodology can be used in conjunction with an existing cycle-accurate microarchitectural simulator to estimate variations in MTTF with the characteristics of the input workload. RAMP deals primarily with the impact of temperature on *wearout* driven (un)reliability. It also considers the effect of degradations in reliability caused by technology scaling, such as the effect of thinner gate-oxides and smaller interconnects. Specifically, RAMP uses state-of-the-art analytical models for five important intrinsic failure mechanisms: time-dependent dielectric breakdown (TDDB) and negative-bias temperature instability (NBTI) in the transistors, electromigration (EM) and stress migration (SM) in the interconnects, and thermal cycling (TC) in the package.

We propose two implementations of RAMP; RAMP 1.0 is a simpler implementation that can be applied

both in real hardware and used in a simulator. A key feature of RAMP 1.0 is its ability to be used for run-time reliability management. RAMP 2.0 is a more complex implementation which improves on the accuracy of RAMP 1.0 by modeling failure mechanisms with lognormal distributions and using Monte-Carlo methods to model series-parallel failure systems.

In this dissertation, we also take a first step in establishing the basic understanding (at the architect's level) of the reliability implications of scaling in the deep-submicron era. By augmenting the failure models in RAMP with scaling specific parameters, we quantified the decrease in reliability of a contemporary superscalar processor from 180nm to 65nm. Our results show large and sharp drops in long-term reliability, especially beyond 90 nm. Of the failure modes that were modeled, time-dependent dielectric breakdown (TDDB) and electromigration appear to present the steepest challenge. Our results also illustrate how scaling is increasing the difference between failure rates assuming worst-case conditions vs. typical operating conditions, as well as amplifying the differences among different applications. An important practical consequence of our scaling results is that, in contrast to current practice, leveraging a single design for multiple remaps across a few technology generations (with only minor design tweaks) will become increasingly difficult.

Our scaling results clearly highlight the emerging lifetime reliability problem. In future reliability constrained systems, microarchitectural reliability solutions will be beneficial, if not necessary. In this dissertation, we explore three microarchitectural techniques for lifetime reliability enhancement. First, we propose dynamic reliability management (DRM), wherein the processor uses architectural adaptation to respond to changing application behavior to maintain its lifetime reliability target. Currently, processor reliability design is done in an application-oblivious fashion and is instead based on worst-case operating conditions. This results in an unnecessary performance and/or cost overhead. By accounting for application behavior, DRM can be used to improve performance and/or lower cost, providing the designer with a spectrum of reliability design points.

We also examined two other application-specific techniques that use structural redundancy for lifetime enhancement: structural duplication (SD) and graceful performance degradation (GPD). In SD, extra or spare structures are added to the processor during microarchitectural definition. Spare structures can be turned on during the processor's lifetime when the original structure fails, thereby extending processor lifetime. Although SD would result in no performance loss in the processor, the spare structures would incur an area (and resultant) cost overhead for the processor. GPD, on the other hand, does not require extra structures to be added to the processor. Instead, GPD exploits existing structural redundancy on chip for reliability. If a redundant structure fails in a processor with GPD, the structure can be shut down and

the processor would still be functional. This however, would come at a performance loss to the processor. Our analysis provides clear guidelines for the use of SD and GPD for reliability enhancement. If only limited performance or area resources can be be diverted to reliability, GPD presents a more attractive option for reliability enhancement. On the other hand, in scenarios where reliability is more important than performance or area, SD is the more beneficial technique. A combination of SD and GPD (SD+GPD) provides the highest reliability increases for the lowest performance and area overheads because it can exploit the benefits of both techniques.

8.2 Future Directions

This dissertation lays the basic foundation for microarchitectural analysis of lifetime reliability and provides new tools and techniques to handle this critical emerging technology challenge. There exist many potential avenues for future work.

8.2.1 Extensions to the RAMP Methodology

RAMP can potentially be used to model other types of failures (e.g., soft errors) and other system components (e.g., memory chips).

Currently, RAMP focuses on intrinsic hard errors. RAMP can easily be extended to model extrinsic or early-life hard failures which are a result of manufacturing defects. Extrinsic failures have the same failure modes and similar failure distributions as intrinsic failures. By accounting for extrinsic failures, RAMP can provide a complete picture of hard error failures in processors. This is of particular interest to the chip industry as fabrication and testing yield is almost completely dictated by early-life failures.

Similarly, components of RAMP can be used to integrate the effect of soft-error failures on processor MTTF. If the MTBF and failure distribution of soft-error failures are known, the Monte-Carlo simulation techniques in RAMP 2.0 can be used to combine them with hard error failures, providing a complete picture of processor reliability.

In addition to modeling different classes of processor failures, RAMP can be used as a starting point to model the reliability of other system components. Since RAMP calculates reliability of CMOS systems, some of its components can be used to model memory chips, graphics processors, and other system components. Similarly, components of RAMP can be used to model simpler CMOS systems ranging from DSPs to ASICs. In future large-scale pervasive computing environments, many processing units of varying complexity will be interlinked. In such a scenario, total system reliability and the reliability of every component will have

to be understood. RAMP could act as a starting point in developing a complete picture of the reliability of such an environment, allowing monitoring and optimization to be performed in a holistic fashion.

In parallel, as discussed in Chapter 4, RAMP should be refined and validated to increase confidence in its results.

8.2.2 Joint Management of Energy, Temperature, and Reliability

This dissertation also provides preliminary directions in handling reliability, temperature, and energy in a unified fashion from a microarchitectural perspective. Currently, these concerns are handled independently, and at different stages of processor design. Despite the similarities in their root causes, the relationship between processor energy, temperature, and reliability is not obvious. For example, in Section 5, we compare processor design for temperature (DTM) with design for reliability (DRM), and show that neither technique subsumes the other. This implies that much more work has to be done on understanding these effects individually and in concert. A unified framework for reliability, energy, and temperature would potentially have a huge impact on processor design, allowing optimal cost/performance tradeoffs based on the requirements of the target system.

References

[1] "Critical Reliability Challenges for The International Technology Roadmap for Semiconductors," in *Intl. Sematech Tech. Transfer 03024377A-TR*, 2003.

[2] S. S. Mukherjee *et al.*, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," in *Proc. of the 36th Intl. Symp. on Microarch.*, 2003.

[3] N. J. Wang *et al.*, "Characterizing the Effects of Transient Faults on a High-Performance Processor Pipeline," in *Proc. of the Intl. Conf. on Dependable Systems and Networks*, 2004.

[4] M. G. Pecht *et al.*, "Guidebook for Managing Silicon Chip Reliabilty," CRC Press, 1999.

[5] E. Y. Wu *et al.*, "Interplay of Voltage and Temperature Acceleration of Oxide Breakdown for Ultra-Thin Gate Dioxides," in *Solid-state Electronics Journal*, 2002.

[6] N. P. Mencinger, "A mechanism-based methodology for processor package reliability assessments," in *Intel Technology Journal*, Q3,2000.

[7] K. Seshan *et al.*, "The Quality and Reliability of Intel's Quarter Micron Process," in *Intel Technology Journal*, Q3,1998.

[8] "Reliability in CMOS IC Design: Physical Failure Mechanisms and their Modeling," in *MOSIS Technical Notes, http://www.mosis.org/support/technical-notes.html.*

[9] S. Borkar, "Design Challenges of Technology Scaling," in *IEEE MICRO*, Jul-Aug 1999.

[10] R. Sasanka, C. J. Hughes, and S. V. Adve, "Joint Local and Global Hardware Adaptations for Energy," in *Proc. of the 10th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002.

[11] K. Skadron *et al.*, "Temperature-Aware Microarchitecture," in *Proc. of the 30th Annual Intl. Symp. on Comp. Arch.*, 2003.

[12] J. Srinivasan *et al.*, "The Case for Lifetime Reliability-Aware Microprocessors," in *Proc. of the 31st Annual Intl. Symp. on Comp. Architecture*, June 2004.

[13] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Lifetime Reliability: Toward an Architectural Solution," in *IEEE Micro*, 2005.

[14] J. Srinivasan *et al.*, "The Impact of Technology Scaling on Lifetime Reliability," in *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, June 2004.

[15] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting Structural Duplication for Lifetime Reliability Enhancement," in *Proc. of the 32nd Annual Intl. Symp. on Comp. Architecture.*

[16] "Compaq NonStop Himalaya S-Series Server Description Manual," in *Compaq Technical Manual 520331-001, http://www.compaq.com.*

[17] L. Spainhower *et al.*, "IBM s/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective," in *IBM Journal of R&D*, September/November 1999.

[18] "Failure Mechanisms and Models for Semiconductor Devices," in *JEDEC Publication JEP122-A*, 2002.

[19] S. Zafar *et al.*, "A model for negative bias temperature instability (nbti) in oxide and high k pfets," in *2004 Symposia on VLSI Technology and Circuits*, June, 2004.

[20] D. Brooks *et al.*, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proc. of the 27th Annual Intl. Symp. on Comp. Arch.*, 2000.

[21] "Sony Semiconductor Quality and Reliability Handbook," in *Sony Global Corp.*, 2001.

[22] J. Kitchin and T.S.Sriram, "A Statistical Approach to Electromigration Design for High Performance VLSI," in *Fourth International Workshop on Stress Induced Phenomena in Metallization*, 1998.

[23] "Electromigration for Designers, Cadence Design Systems White Paper," in *http://www.cadence.com/whitepapers/electromigration.html*, 1999.

[24] A. Christou, "Electromigration and Electronic Device Degradation," 1994.

[25] A. Chandrakasan, W. J. Bowhill, and F. Fox, "Design of High-Performance Microprocessor Circuits," 2001.

[26] A. Dasgupta *et al.*, "Electromigration Reliability Enhancement Via Bus Activity Distribution," in *Design Automation Conference*, 1996.

[27] C. K. Hu *et al.*, "Scaling Effect on Electromigration in On-Chip Cu Wiring," in *International Electron Devices Meeting*, 1999.

[28] E. Eisenbraun *et al.*, "Integration of CVD W- and Ta-based Lines for Copper Metallization," in *MKS white paper, http://www.mksinst.com/techpap.html*, 2000.

[29] J. H. Stathis, "Reliability Limits for the Gate Insulator in CMOS Technology," in *IBM Journal of R&D, Vol. 46*, 2002.

[30] K. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications.* Prentice Hall, 1982.

[31] "Assessing Product Reliability, Chapter 8, NIST/SEMATECH e-Handbook of Statistical Methods," in *http://www.itl.nist.gov/div898/handbook/.*

[32] J. McPherson *et al.*, "Comparison of E and 1/E TDDB Models for Si02 under long-term/low-field test conditions," in *Technical Digest IEEE International Electron Devices Meeting (IEDM)*, 1998.

[33] H. Masuda *et al.*, "Assessment of a 90nm PMOS NBTI in the Form of Products Failure Rate," in *Proceedings of the IEEE 2005 International Conference on Microelectronic Test Structures*, 2005.

[34] R. A. Johnson, *Probability and Statistics for Engineers.* Prentice Hall of India, 2002.

[35] "Methods for calculating failure rates in units of fits," in *JEDEC Publication JESD85*, 2001.

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, 1992.

[37] J. E. Gentel, *Random Number Generation and Monte Carlo Methods.* Springer, 2003.

[38] K.H.Min *et al.*, "Comparative study of tantalum and tantalum nitrides as a diffusion barrier for cu metallization," in *Journal of Vacuum Science and Technology*, 1996.

[39] T.McPherson *et al.*, "760 mhz g6 s/390 microprocessor exploiting multiple vt and copper interconnects," in *ISSCC Digest of Technical Papers*, 2000.

[40] G.Yoh and F. Najm, "A statistical model for electromigration failures," in *International Symposium on Quality Electronic Design*, 2000.

[41] E. T. Ogawa *et al.*, "Leakage, Breakdown, and TDDB Characteristics of porous low-k silica based interconnect materials," in *International Reliability Physics Symposium*, 2003.

[42] J.H.Stathis and D.J.DiMaria, "Reliability projections for ultra-thin oxides at low voltage," in *International Electron Devices Meeting*, 1998.

[43] C. Moore, "The POWER4 System Microarchitecture," in *Microprocessor Forum*, 2000.

[44] M. Moudgill *et al.*, "Validation of turandot, a fast processor model for microarchitectural exploration," in *IEEE Intl Perf., Computing, and Communications Conf.*, 1999.

[45] D. Brooks *et al.*, "Power-aware Microarchitecture: Design and Modeling Challenges for the next-generation microprocessor," in *IEEE Micro*, 2000.

[46] J. S. Neely *et al.*, "Cpam: A common power analysis methodology for high-performance vlsi design," in *9th Topical Meeting on the Electrical Performance of Electronic Packaging*, 2000.

[47] S. Heo *et al.*, "Reducing Power Density Through Activity Migration," in *Intl. Symp. on Low Power Elec. Design*, 2003.

[48] M. Moudgill *et al.*, "Environment for powerpc microarchitectural exploration," in *IEEE Micro*, 1999.

[49] V. Iyengar, L. H. Trevillyan, and P. Bose, "Representative Traces for Processor Models with Infinite Cache," in *Proc. of the 2nd Intl. Symp. on High-Perf. Comp. Architecture*, 1996.

[50] P. Bose, "Power-Efficient Microarchitectural Choices at the Early Design Stage," in *Keynote Address, Workshop on Power-Aware Computer Systems*, 2003.

[51] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan, "Variability in the Execution of Multimedia Applications and Implications for Architecture," in *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[52] J. Srinivasan and S. V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," in *Proc. of the 2003 Intl Conf. on Supercomputing*, 2003.

[53] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas, "A Framework for Dynamic Energy Efficiency and Temperature Management," in *Proc. of the 33rd Annual Intl. Symp. on Microarchitecture*, 2000.

[54] J. Flinn and M. Satyanarayanan, "Energy-Aware Adaptation for Mobile Applications," in *Proceedings of the 17th ACM Symp. on Operating Systems Principles*, 1999.

[55] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in *Proc. of the 7th Intl. Symp. on High Performance Comp. Architecture*, 2001.

[56] C. J. Hughes, J. Srinivasan, and S. V. Adve, "Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications," in *Proc. of the 34th Annual Intl. Symp. on Microarchitecture*, 2001.

[57] D. H. Albonesi *et al.*, "Dynamically Tuning Processor Resources with Adaptive Processing," in *IEEE Computer*, 2003.

[58] J. Srinivasan and S. V. Adve, "The Importance of Accurate Heat-Sink Modeling and a Correction to "Predictive DTM for Multimedia Applications"," in *Workshop on Duplicating, Deconstructing, and Debunking (WDDD) at ISCA-05*, 2005.

[59] "The Intel Pentium M Processor: Microarchitecture and Performance." http://www.intel.com/ technology/itj/2003/volume07issue02/art03_pentiumm/p01_abstract.htm.

[60] V. S. Pai, P. Ranganathan, and S. V. Adve, "RSIM Reference Manual version 1.0," Tech. Rep. 9705, Department of Electrical and Computer Engineering, Rice University, August 1997.

[61] J. L. Hennessy and D. A. Patterson, *Computer Architecture, A Quantitative Approach*. Morgan Kaufmann, 2003.

[62] J. R. Black, "A brief survey of electromigration and some recent results," in *IEEE Transactions on Electron Devices*, 1969.

[63] W. Abadeer *et al.*, "Key Measurements of Ultrathin Gate Dielectric Reliability and In-Line Monitoring," in *IBM Journal of Research and Development*, 1999.

[64] K. P. Cheung, "Thin Gate-oxide Reliability - the current status," in *Keynote paper, Symposium on Nano Device Technology 2001*, 2001.

[65] E. Y. Wu *et al.*, "Cmos scaling beyond the 100-nm node with silicon-dioxide-based gate dielectrics," in *IBM Journal of Research and Development*, March/May, 2002.

[66] H. V. Nguyen *et al.*, "Fast temperature cycling stress-induced and electromigration-induced interlayer dielectric cracking failure in multilevel interconnection," 2001.

[67] K. Diefendorff, "Ibm paving the way to 0.10 micron," in *Microprocessor Report*, May 2000.

[68] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, "Interconnect Lifetime Prediction under Dynamic Stress for Reliability Aware Design," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2004.

[69] Z. Lu, J. Lach, M. Stan, and K. Skadron, "Banking Chip Lifetime: Opportunities and Implementation," in *Proceedings of the Workshop on High Performance Computing Reliability Issues (in conjunction with HPCA 2005)*, 2005.

[70] E. W. Czeck and D. Siewiorek, "Effects of transient gate-level faults on program behavior," in *Proceedings of the 1990 International Symposium on Fault-Tolerant Computing*, pp. 236–243, June 1990.

[71] J. Ohlsson, M. Rimin, and U. Gunneflo, "A Study of the Effects of Transient Fault Injection into a 32-bit RISC with Built-in Watchdog," in *Proceedings of the 1992 International Symposium on Fault-Tolerant Computing*, June 1992.

[72] H. Cha, E. M. Rudnick, J. H. Patel, R. K. Iyer, and G. S. Choi, "A gate-level simulation environment for alpha-particle-induced transient faults," *IEEE Transactions on Computers*, vol. 45, pp. 1248–1256, Nov. 1996.

[73] P. Folkesson, S. Svensson, and J. Karlsson, "A Comparison of Simulation Based and Scan Chain Implemented Fault Injection," in *Proceedings of the 1998 International Symposium on Fault-Tolerant Computing*, June 1998.

[74] Z. Kalbarczyk, R. K. Iyer, G. L. Ries, J. U. Patel, M. S. Lee, and Y. Xiao, "Hierarchical Simulation Approach to Accurate Fault Modeling for System Dependability Evaluation," *IEEE Transactions on Software Engineering*, vol. 25, pp. 619–632, Sept. 1998.

[75] C. R. Yount and D. P. Siewiorek, "A Methodology for the Rapid Injection of Transient Hardware Errors," *IEEE Transactions on Computers*, vol. 45, no. 8, pp. 881–891, 1996.

[76] M. Baze and S. Buchner, "Attenuation of single event induced pulses in CMOS combinational logic," *IEEE Transactions on Nuclear Science*, vol. 44, pp. 2217–2223, Dec. 1997.

[77] S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger, "Comparison of error rates in combinational and sequential logic," *IEEE Transactions on Nuclear Science*, vol. 44, pp. 2209–2216, Dec. 1997.

[78] D. Gil, R. Martínez, J. V. Busquets, J. C. Baraza, and P. J. Gil, "Fault Injection into VHDL Models: Experimental Validation of a Fault Tolerant Microcomputer System," in *Proceedings of the Third European Dependable Computing Conference*, Sept. 1999.

[79] P. Hazucha and C. Svensson, "Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate," *IEEE Transactions on Nuclear Science*, vol. 47, pp. 2586–2594, Dec. 2000.

[80] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walstra, and C. Dai, "Impact of CMOS Process Scaling and SOI on the Soft Error Rates of Logic Processes," in *Proceedings of the 2001 Symposium on VLSI Technology*, June 2001.

[81] N. Seifert, X. Zhu, D. Moyer, R. Mueller, R. Hokinson, N. Leland, M. Shade, and L. Massengill, "Frequency Dependence of Soft Error Rates for Sub-Micro CMOS Technologies," in *Proceedings of the 2001 IEEE International Electron Devices Meeting*, pp. 14.4.1 – 14.4.4., 2001.

[82] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pp. 389–398, June 2002.

[83] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings of the 36th International Symposium on Microarchitecture*, pp. 29–40, Dec. 2003.

[84] N. Wang, T. Rafacz, J. Quek, and S. J. Patel, "Characterizing the effects of transient faults on a modern high-performance processor pipeline," in *To Appear: Proceedings of the International Conference on Dependable Systems and Networks*, 2004.

[85] W. G. Magda, "Evaluating the impact of noise on microprocessor operation using an RTL model," Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[86] N. Wang, M. Fertig, and S. Patel, "Y-branches: When you come to a fork in the road, take it," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 56–66, 2003.

[87] D. Patterson *et al.*, "Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies," in *UC Berkeley CS Tech. Report UCB//SD-02-1175*, 2002.

[88] G. Hetheringon *et al.*, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," in *Proceedings of the International Test Conference*, 1999.

[89] F. Bower *et al.*, "Tolerating Hard Faults in Microprocessor Array Structures," in *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, 2004.

[90] I. Koren *et al.*, "Defect Tolerant VLSI Circuits: Techniques and Yield Analysis," in *Proceedings of the IEEE*, vol. 86, 1998.

[91] P. Shivakumar *et al.*, "Exploiting Microarchitectural Redundancy for Defect Tolerance," in *21st Intl. Conf. on Comp. Design*, 2003.

[92] K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," in *Proc. of the 1st Intl. Conf. on Mobile Computing and Networking*, 1995.

[93] D. Grunwald *et al.*, "Policies for Dynamic Clock Scheduling," in *Proc. of the 4th Symposium on Operating Systems Design and Implementation*, 2000.

[94] T. R. Halfhill, "Transmeta Breaks x86 Low-Power Barrier," *Microprocessor Report*, February 2000.

[95] Y.-H. Lee and C. Krishna, "Voltage-Clock Scaling for Low Power Energy Consumption in Real-Time Embedded Systems," in *Proc. of the 6th Intl. Conference on Real-Time Computing Systems and Applications*, 1999.

[96] M. Weiser *et al.*, "Scheduling for Reduced CPU Energy," in *Proc. of the 1st Symposium on Operating Systems Design and Implementation*, 1994.

[97] S. Manne, A. Klauser, and D. Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction," in *Proc. of the 25th Annual Intl. Symp. on Comp. Architecture*, 1998.

[98] A. Buyuktosunoglu *et al.*, "An Adaptive Issue Queue for Reduced Power at High Performance," in *Proc. of the Workshop on Power-Aware Computer Systems*, 2000.

[99] S. Dropsho *et al.*, "Integrating Adaptive On-Chip Storage Structures for Reduced Dynamic Power," in *Proc. of the Intl. Conf. on Parallel Architectures and Compilation Techniques*, 2002.

[100] D. Folegnani and A. González, "Energy-Efficient Issue Logic," in *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[101] A. Iyer and D. Marculescu, "Power Aware Microarchitecture Resource Scaling," in *Proc. of the Design, Automation and Test in Europe Conf.*, 2001.

[102] D. Ponomarev, G. Kuck, and K. Ghose, "Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources," in *Proc. of the 34th Annual Intl. Symp. on Microarchitecture*, 2001.

[103] R. I. Bahar and S. Manne, "Power and Energy Reduction Via Pipeline Balancing," in *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[104] R. Maro, Y. Bai, and R. Bahar, "Dynamically Reconfiguring Processor Resources to Reduce Power Consumption in High-Performance Processors," in *Proc. of the Workshop on Power-Aware Computer Systems*, 2000.

[105] A. Buyuktosunoglu, T. Karkhanis, D. H. Albonesi, and P. Bose, "Energy Efficient Co-Adaptive Instruction Fetch and Issue," in *Proc. of the 30th Annual Intl. Symp. on Comp. Architecture*, 2003.

[106] D. H. Albonesi, "Selective Cache Ways: On-Demand Cache Resource Allocation," in *Proc. of the 32nd Annual Intl. Symp. on Microarchitecture*, 1999.

[107] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," in *Proc. of the 8th Intl. Symp. on High Performance Comp. Architecture*, 2002.

Author's Biography

Jayanth Chandrasekar Srinivasan was born in Tanjore, India on January 24th, 1979. He attended high school at Vana Vani Matriculation in Madras, India. He received his Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Madras in 2000, and the Master of Science degree in Computer Science from the University of Illinois at Urbana-Champaign in 2003.