LIMITATIONS OF THE MTTF METRIC FOR ARCHITECTURE-LEVEL LIFETIME RELIABILITY ANALYSIS

BY

PRADEEP RAMACHANDRAN

B.Tech., Indian Institute of Technology, Madras, 2005

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

To amma, appa, Praveen and ammamma,

who support me from across the globe

ABSTRACT

This work concerns metrics for evaluating microarchitectural enhancements to improve processor lifetime reliability. The commonly used metric is mean time to failure (MTTF). However, MTTF does not provide information on the reliability characteristics during the relatively short operational life of a processor. We study nTTF, the time to failure of n% of the population. nTTF describes the failure process more accurately than MTTF, but determining it requires knowledge of the distribution of processor failure times which is generally hard to obtain.

This thesis, for the first time, evaluates the nTTF metric and quantifies its relation with MTTF for the lifetime reliability of a modern superscalar architecture. Specifically, we show through an in-depth analysis that the nTTF metric differs significantly from the MTTF metric and hence, designing for MTTF as a proxy for nTTF can lead to sub-optimal designs. Additionally, we present a straight-forward method of evaluating the nTTF metric with current architecture-level lifetime reliability analysis tools.

ACKNOWLEDGMENTS

An enumeration of all those who helped me complete this thesis would be run into a book by itself. I shall, however, attempt to thank all those who have helped me complete this first piece of work that paves way for my PhD.

Words alone cannot express my gratitude to my adviser, Sarita Adve. Her patience and pointed guidance has been an inspiration to pursue challenging problems in research. Her friendly nature and "never-say-die" attitude continually drives me to perform better. I look to grow further as she continues her guidance for my PhD.

My research this far (and hopefully well into the future) has been constantly mentored by Pradip Bose from IBM. His insights into reliability-related issues in processor design and lateral way of thinking have significantly impacted the direction that this research project has taken. I hope that these interactions continue as I march on into my PhD and help orient my research directions.

Much of this work has built up from work that Jayanth Srinivasan did previously. I would like to thank him for the fantastic work that lays the foundation for this thesis.

I would also like to acknowledge the following sources which have funded my work at Illinois - an IBM faculty partnership award, the MARCO/FCRP Gigascale Systems Research Center, the National Science Foundation under Grant No. NSF CCF-0541383, and an equipment donation from AMD.

Life in the middle of nowhere (aka Champaign) would've been a drag if not for the plethora of friends at Illinois. Naming all of them would be out of the question, but I would mention a couple of names here. Thanks to my room-mates, Raghu and Praveen, for putting up with all my theorems and jokes at home. Thanks to my office-mate Alex, for the long sessions of slacking in the office and the play-till-you-die badminton.

A constant force supports me every moment of every day where-ever I am in the world. The support from my friends and family back home has paved the path of my life this far and continues to do so everyday.

Amma, appa, Praveen, Ammamma, Suchi, DC, Amnik, and many others, I owe everything I have to them, my work, my life, my own.

As a symbol of this gratitude to my family, I would like to dedicate this first of two thesis that I will be writing at Illinois to them. Hope that my next dedication will not be too far in the distant future.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

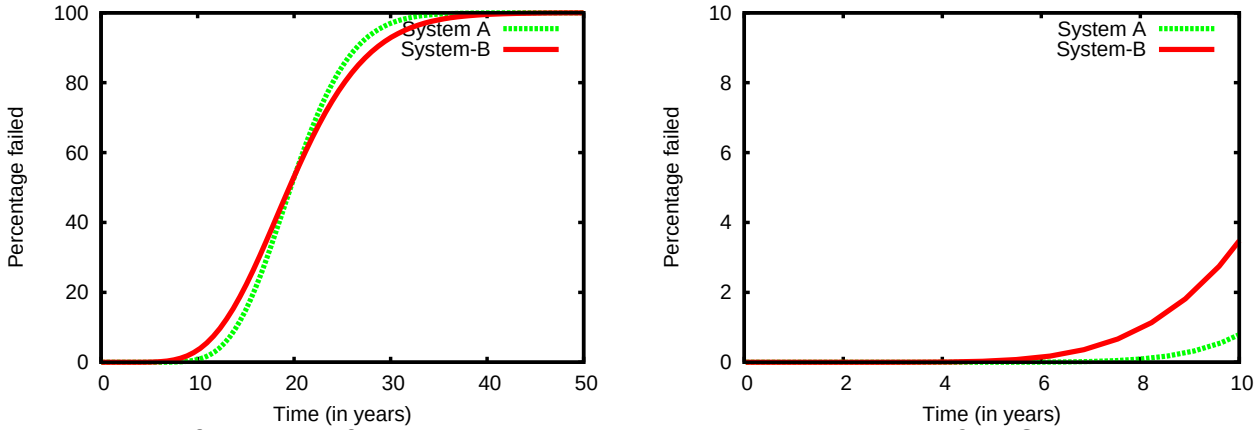
An important goal for processor designers is to ensure long-term or "lifetime" reliability against hard failures. Unfortunately, aggressive CMOS scaling coupled with increased on-chip temperatures is accelerating the onset of wear-out or aging related hard failures due to effects such as electromigration (EM), gate oxide breakdown (OBD), and negative bias temperature instability (NBTI) [2, 4, 25, 31].

Until recently, in the broad general-purpose processor market, lifetime reliability was largely addressed at the device level, without much help from microarchitects. Although such an approach tackles the problem at its root, it generally cannot exploit high-level application and system behavior. Recently, several academic and industry researchers have suggested exploring microarchitecture-level solutions for the lifetime reliability problem [4, 5, 6, 14, 24, 26, 31]. Such a solution can be application-aware and opens up new cost-performance reliability points for general-purpose processors, which were previously unavailable.

Appropriate metrics and models are essential to enable effective microarchitecture-level lifetime reliability research. The most commonly used metric for reliability is mean time to failure (MTTF), which is the expected lifetime of the given population of processors. Srinivasan et al. recently proposed RAMP, the first generation of microarchitecture level models to calculate workload-specific MTTF of a processor for various wear-out failure mechanisms [24, 26].

A key limitation of the MTTF metric is that it is a single summary statistic that does not provide insight on the failure behavior of the population, especially for failure mechanisms like wear-out, that do not have a constant failure rate [21, 28, 30]. The following example illustrates this limitation of MTTF.

Figure 1.1(a) shows the cumulative distribution function (CDF) of the lifetimes for two simulated product lines (the methodology for this simulation is explained in Section 4). The MTTF for both the product lines is 20 years, a typical MTTF target for processor designers [17]. However, the typical operational life



(a) Lifetime CDF for 50 years **(b) Closer view of the first 10 years**

Figure 1.1 MTTF can be misleading. Part (a) shows the cumulative distribution function (CDF) of the lifetimes for two product lines with MTTF of 20 years. Part (b) is a close view of the same graph for 10 years. The distributions are different although the MTTFs are identical. System A is preferable to system B for a <10 year usage as it sees fewer failures in its early life. MTTF fails to provide such information.

for a processor is much shorter, e.g., of the order of 5 to 7 years. Figure 1.1(b) shows a closer view of the CDF for the first 10 years, which is more relevant to the end user. The figure shows that the onset of failures for system A is far slower than that for system B in the first 10 years. For example, in 5 years, system A processors have seen practically no failures, but 0.03% of system B processors have already failed. Subsequently, in about 7 years, only about 0.02% of the processors in system A have failed, while about 0.4% of the processors in system B have failed. For users who expect to upgrade within 5-7 years, system A is clearly a better choice than system B. The MTTF metric, however, fails to make this distinction.

Designers typically target a much higher MTTF than the expected operational life of a processor so that most customers do not see failures during the operational life. However, the MTTF metric itself does not reveal the probability of failure over the operational life of the processor, as Figure 1.1 illustrates. This limitation of the MTTF metric has an impact on both the customer and the vendor.

From the customer's perspective, paying a premium for a high MTTF is not useful if it does not translate to a commensurately lower probability of failure during the expected operational life. In Figure 1.1, the customer would rather choose system A, but has no way to do so if the vendor only supplies the metric of MTTF. From the vendor's perspective, apart from the number of disgruntled customers, warranty and replacement costs also depend on the number of failures during the expected operational life which, in spite

of similar MTTFs, may be varied. Reliability models based purely on MTTF cannot distinguish between different designs from such perspectives and may thus result in incorrect systems.

1.1 The nTTF Metric

This thesis explores nTTF – an alternate metric that addresses the above limitations of MTTF. Mathematically, nTTF is the t at which F(t) = $\frac{n}{100}$, where F(t) is the CDF for the processor lifetimes. Informally, nTTF can be interpreted as the time to failure of n% of a huge processor population or the time t at which the probability of failure of a processor in a large population is $\frac{n}{100}$.

To emphasize the relationship with the commonly used MTTF, we denote this metric as $nTTF$ (n is a relatively small number for cases of interest). For example, 1-TTF = 5 years implies that 1% of the processor population will fail in 5 years; conversely, the probability of failure for a given processor in $\leq$5 years is 0.01.[1] nTTF, for various values of n, accurately represents the distribution of failures, overcoming the shortcoming of MTTF which loses such information to averaging.

From the vendor's point of view, an nTTF driven design must ensure that nTTF exceeds the anticipated useful operational life for some acceptable failure probability ($0.01 \times n$). Both n and the desirable nTTF depend on the market, including factors such as customer satisfaction, warranty costs, and anticipated useful operational life. For example, acceptable values of n (probability of failure) for the desktop market are likely to be higher than for the server market. Providing nTTF data for multiple values of n would allow the customer to consider the value most appropriate for their use.

From the customer's point of view, nTTF indicates that a given processor will fail with probability $0.01 \times n$ within nTTF years. This gives the customer more information than with MTTF for comparing products. For example, it may not be worthwhile for a desktop user to pay a cost premium to obtain 1-TTF $>$ 10 years, but it may be worthwhile to pay a cost premium to obtain 0.1-TTF $>$ 5 years.

[1] Note that for non-uniform distributions, like the exponential and the lognormal distributions, 50-TTF does not correspond to the MTTF. For the exponential distribution, 63-TTF corresponds to the MTTF and no closed form relationship is known for the lognormal distribution.

1.2 Contributions of this thesis

This thesis, for the first time, provides a quantitative evaluation of nTTF and its relationship to MTTF for lifetime reliability enhancing techniques for modern superscalar architectures. To do so, we show how to extend the state-of-the-art architecture-level RAMP 2.0 tool to measure nTTF in a straightforward way, making early-stage nTTF-driven processor design practical.

Specifically, we use our extended version of RAMP 2.0 to compare nTTF and MTTF for previous reliability enhancing techniques based on sparing and degradation of microarchitecture-level components, for a baseline out-of-order superscalar processor running various SPEC CPU2000 benchmarks.

Similar to MTTF, our results indicate that the nTTF metric is also application dependent, showing a large variation across different applications. Thus, even under the nTTF metric, a worst-case analysis based design may be overly conservative, incurring unnecessary performance and/or cost overheads. Techniques like Dynamic Reliability Management (DRM) continue to play an important role [24].

We also find that there is no obvious relationship between MTTF and nTTF for our systems. The key reason is that wear-out failures follow complex non-exponential lifetime distributions – RAMP 2.0 uses lognormal distributions for individual microarchitectural components, and uses Monte Carlo simulations with a MIN-MAX analysis to aggregate these lognormals into the overall lifetime of the processor. The resulting distribution of the processor lifetime is too complex to derive analytical (or empirical) relationships between MTTF and nTTF. Our results show that because of this complex relationship, designs that are optimal for MTTF may not be optimal for nTTF. In particular, if the designer is willing to incur an overhead in die area or performance, to achieve the reliability target, we see that designing to MTTF as a proxy for nTTF leads to systems that are generally over-designed, i.e., a design with a lower area overhead would achieve the required reliability target under nTTF, but using MTTF instead results in a design with a higher area overhead.

The unpredictable relation between nTTF and MTTF arises from the complexity of the underlying failure models. A lognormal failure distribution, with a Min-Max analysis makes this relationship intractable. A common industry practice is to assume exponential lifetime distributions which are widely used for their simplicity and analytical tractability (including in RAMP 1.0). For this distribution, it is known that the two metrics are proportional to each other for a given n (nTTF = $-MTTF \times ln(1 - \frac{n}{100})$) [15]. Thus, the common exponential assumption leads to a situation where the same designs are optimal both for MTTF and

nTTF for given n, and it is not useful to distinguish the two metrics. However, exponential distributions are known to be inaccurate for wear-out [20]. Hence, we quantify the errors in using the exponential distribution instead of more realistic lifetime distributions (specifically the lognormal distribution) for both nTTF and MTTF. We find that the absolute error in estimating nTTF or MTTF from an exponential distribution is in excess of 80% and hence, the exponential approximation is not valid for failures caused by wear-out.

Overall, our experimental results question the common industry practice of MTTF-driven designs for wear-out failures with complex lifetime distributions. This paper, for the first time, quantifies the impact of such a practice for modern superscalar architectures, and provides an architecture-level tool for the more meaningful nTTF-driven design methodology.

CHAPTER 2

RELATED WORK AND BACKGROUND

Section 2.1 describes related work on alternate metrics. Section 2.2 provides background on RAMP [24, 26], a state-of-the-art architecture level reliability model and tool we use in this study. Section 2.3 provides background on the two reliability-enhancing techniques we study here – Structural Duplication (SD) and Graceful Performance Degradation (GPD) [26].

2.1 Reliability Metrics

As mentioned in Chapter 1, MTTF (or MTBF – mean time between failures) is perhaps the most commonly used reliability metric in the electronics industry [30]. An alternate metric also widely used is based on failure rate, measured in units of FITs or failures per billion hours of operation. It is common practice to quote a constant FIT rate and use the relationship $FITs = \frac{10^9}{MTTF}$. However, the constant failure rate assumption and the reciprocal relationship between MTTF and failure rate only hold for failure mechanisms with exponential lifetime distributions [29]. It is widely accepted that wear-out based failures do not follow exponential lifetime distributions and have failure rates that change with time [29]. We assume the more realistic non-exponential model and so do not use FITs.

The nTTF metric explored here is widely used in the automotive and mechanical industries. In the automotive industry, it is commonly referred to as B-life and for mechanical devices like fans, it is referred to as L-life [30] (L_{10} is a commonly used metric [22]). In statistics and reliability theory, the nTTF function is referred to as the percent point function or the inverse distribution function (since it is the inverse of the cumulative distribution function of processor lifetime) [20]. Some papers from the computer industry also describe this metric [18, 28, 30] (sometimes referred to as t_n [18]), stating that it is more meaningful than MTTF but is not commonly used in the electronics industry due to its complexity [28, 30]. One concurrent

microarchitecture study has used 1% failure times as the metric to compare against which various reliability-aware designs [9].

However, none of these perform a quantitative comparison between the MTTF and the nTTF metric. Additionally, they do not quantify the impact of using MTTF-driven design instead of nTTF-driven design for processors. We present such an in-depth analysis of the nTTF metric. While experts of reliability theory may find it obvious that nTTF is a more natural metric than MTTF, the latter continues to be widely used in the microprocessor industry. Our results, for the first time, quantify the inaccuracies from using MTTF and motivate the use of the more appropriate nTTF while also demonstrating how it can be measured with current evaluation techniques.

An alternative metric to nTTF is its inverse – the probability of failure within a specified time (t). We refer to this metric as FIT-t (percentage failed in t years) to convey the symmetry with the more common failure rate based metric. Arguably, this metric may be more insightful than nTTF for customers who may be able to better anticipate their useful operational period (t) and wish to compare the probability of failure during this time. This metric is the same as the CDF of the lifetimes, which is also equivalent to 1– reliability. The methodology described in this paper and the key results are easily extensible to this metric as well, and hence we don't report these results.

Finally, there are various reliability based metrics derived from considerations other than failure; e.g., metrics related to repair time, replacement rate, maintenance, etc. These include part replacement rate, warranty claim rate, etc. Wood summarizes several such metrics [30]. Most of these metrics are, however, based on analysis of field data and is hence beyond the scope of this work.

2.2 RAMP

RAMP [24, 26] models the following wear-out failure mechanisms in processors: Electromigration (EM), Stress Migration (SM), Oxide Breakdown (OBD), Thermal Cycling (TC) and Negative Bias Temperature Instability (NBTI) [3]. It works in conjunction with a timing simulator, a power model, and a temperature model (in our case, Turandot [19], PowerTimer [8], and HotSpot [23], respectively).

RAMP starts with state-of-the-art device level analytical models for each of the above failure mechanisms. These models compute MTTF as a function of various operating parameters such as temperature, voltage, and utilization, assuming steady state operating conditions. RAMP abstracts these models at the

architecture level, incorporating workload-driven changes in the different parameters over time.

Much like previous power and temperature models [7, 23], RAMP divides the processor into a few discrete structures – functional units, register file, caches, etc. – and applies the analytic failure models to the structure as a whole. Every few cycles of the timing simulator, RAMP applies the analytic models to each structure to calculate the "instantaneous" MTTF for the *current* operating conditions derived using utilization and temperature information from the timing, power, and temperature simulators. RAMP then averages such "instantaneous" MTTFs, collected over the entire execution of the application, to give a net MTTF for a given structure and given failure mechanism for the specified workload.[1]

To obtain the overall reliability of the processor, RAMP needs to combine the MTTFs across different failure mechanisms and different structures. This requires knowledge of the lifetime distributions of the different failure mechanisms, which is generally difficult. RAMP 1.0 used a simple but commonly used and potentially inaccurate assumption while RAMP 2.0 used a potentially more accurate assumption. We describe both below since we use both in this paper.

2.2.1 Combining failures from different mechanisms and structures with RAMP 1.0.

RAMP 1.0 uses the industry standard Sum-of-Failure-Rates (SOFR) model [15, 29], which makes two assumptions: (1) the processor is a series failure system, i.e., the first instance of any structure failing due to any failure mechanism causes the entire processor to fail, and (2) each individual failure mechanism has a failure rate that stays constant in time, or equivalently, each failure mechanism has an exponentially distributed lifetime.

The above two assumptions imply [29]: (1) the lifetime distribution of the processor is also exponential (i.e., constant failure rate) and the failure rate of the processor is the sum of the failure rates of the individual structures due to individual failure mechanisms, and (2) the MTTF of the processor, $MTTF_p$, is the inverse of the total (constant) failure rate of the processor, λ_p (true for exponentially distributed lifetimes). Hence,

$$MTTF_p = \frac{1}{\lambda_p} = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{f}\lambda_{ij}} \tag{2.1}$$

where λ_{ij} is the failure rate of the i^{th} structure due to the j^{th} failure mechanism, n is the number of structures,

[1] The assumption underlying the averaging over time is analogous to the SOFR assumption described for RAMP 1.0 later in this section since SOFR averages over space.

and f is the number of failure mechanisms.

2.2.2 Combining failures from different mechanisms and structures with RAMP 2.0.

The SOFR model used by RAMP 1.0 assumes a constant failure rate for a given failure mechanism and structure. This is clearly inaccurate as a typical wear-out based failure mechanism starts with a low failure rate at the beginning of the component's lifetime and has an increasing failure rate as the component ages. Further, the series failure assumption of the SOFR model is also inaccurate if the processor supports redundant structures (as with the two reliability enhancing techniques studied here). In these cases, the failure of a single component does not imply the failure of the processor since the redundant component takes over.

RAMP 2.0 addresses the above two limitations of the SOFR model used in RAMP 1.0. First, instead of the exponential distribution, RAMP 2.0 assumes lognormal lifetime distributions for individual structures and failure mechanisms.[2] Lognormal distributions have shown to be more accurate for degradation processes common to semiconductor materials due to the multiplicative degeneration argument [20]. Since lognormal distributions are hard to deal with analytically, RAMP 2.0 uses the Monte Carlo simulation method to calculate the full processor MTTF from the lognormal lifetime distributions of individual structures and failure mechanisms. The mean of these individual distributions is provided by the timing simulator run similar to RAMP 1.0 and σ, the shape parameter for the lognormal distribution, is set at 0.5 (used for wear-out failures in prior work [1, 20]).

Second, RAMP 2.0 also does not require the series failure assumption. Series, parallel redundant, and/or standby redundant systems can be modeled using a MIN-MAX analysis on the individual component lifetimes in a given Monte Carlo trial.

2.3 Reliability-Enhancing Techniques

We examine two reliability-enhancing methods based on structure-level redundancy explored by Srinivasan et al. [26].

In the first method, **Structural Duplication (SD)**, certain redundant microarchitectural structures are added to the processor, and designated as *spares*. Spare structures can be turned on during the processor's

[2]The density function of the standard exponential distribution is given by $f(x) = e^{-x}, x \geq 0$. A variable x is said to follow a lognormal distribution if y = ln(x) follows a normal distribution. The density function of the standard lognormal distribution, with shape parameter $\sigma \geq 0$ is given by $f(x) = \frac{e^{-(lnx)^2/2\sigma^2}}{x\sigma\sqrt{2\pi}}$.

lifetime when the original structure fails. Hence, in a situation where the processor would have normally failed, the spare structure extends the processor's lifetime. With SD, the processor fails only when a structure with no spare fails, or if all available spares for a structure have also failed. The main function of the spares is to increase the reliability and not to enhance the performance; therefore, they are power gated and not used in the beginning of the processor's life.

The second method, **Graceful Performance Degradation (GPD)**, allows the processor to exploit existing microarchitectural redundancy for reliability. Modern processors have replicated structures to increase the performance for applications with instruction-level parallelism. The replicated structure, however, is not necessary for functional correctness. If the replicated structure fails in the course of a processor's lifetime, the processor can shut down the structure and can still maintain functionality, thereby increasing lifetime. With GPD, the processor fails only when a structure with no redundancy fails or when all redundant structures of a given type fail.

Both SD and GPD incur overheads while increasing processor reliability. In the case of SD, extra processor die area is required to introduce the spares incurring a cost overhead. In the case of GPD, a performance loss is incurred to improve the reliability. We will explore SD and GPD configurations with various overheads.

CHAPTER 3

INCORPORATING nTTF INTO RAMP

Mathematically, nTTF is defined as the time t at which

$$F(t) = \frac{n}{100} \tag{3.1}$$

where $F(t)$ is the cumulative distribution function (CDF) for the processor lifetime [29].

We calculate nTTF with RAMP 2.0 as follows. As mentioned in Section 2, RAMP 2.0 uses a Monte Carlo simulation to generate the processor MTTF from the MTTFs of the individual structures and failure mechanisms generated from the timing simulator run. Each Monte Carlo trial generates a value for the time to failure for each individual structure for each failure mechanism. This time is generated from the corresponding lognormal distribution, which is fully specified by the mean (MTTF) provided by the timing simulator run and shape parameter σ of 0.5 as mentioned earlier.

Using these lifetime values, RAMP 2.0 performs a MIN-MAX analysis across all structures to get the lifetime for the full processor. For example, the base processor with no redundancy is a series system. The lifetime of each structure is the minimum of the time to failure from each failure mechanism. The lifetime of the full processor is the minimum of the lifetimes for each structure. Details on the analysis for the SD and GPD systems are provided in [26].

Thus, each Monte Carlo trial generates the lifetime for one processor sample. RAMP 2.0 averages these lifetimes over 10^7 trials to provide the MTTF of the processor.

The processor lifetimes generated in the 10^7 Monte Carlo trials of RAMP 2.0 directly provide information on the cumulative distribution function for the processor lifetimes. To calculate nTTF, we therefore find the smallest processor lifetime value such that n% of the trials lie within that time. [1]

[1] We found 10^7 Monte Carlo trials to be sufficient to reach convergence for both MTTF and nTTF. For example, 10^8 trials

resulted in a difference of less than 0.1%.

CHAPTER 4

EXPERIMENTAL METHODOLOGY

We perform our evaluations using RAMP 2.0 (extended to calculate nTTF) coupled with performance, power, and temperature simulators and models, using a methodology similar to previous RAMP-based work [24, 25, 26].

4.1 Base Processor

The base processor we used for our simulation is a 65nm, out-of-order, 8-way superscalar processor. The microarchitecture is similar to that of a single core within a POWER4 chip [27]. The 65nm processor parameters were derived from scaling down parameters from the 180nm POWER4 processor [25]. Although we model the performance impact of the L2 cache, we don't model the reliability as its temperature is much lower than the processor core [16], resulting in very few L2 cache failures. Table 4.1 summarizes the base processor modeled.

4.2 Simulation Environment

For timing simulations, we use Turandot, a trace-driven research simulator developed at IBM's T.J.Watson Research Center [19]. Turandot was calibrated against a pre-RTL, detailed, latch-accurate processor model.

For the power model, we use PowerTimer, a tool-set built around Turandot [8]. The power values from PowerTimer are fed into HotSpot to evaluate the temperatures of various components on chip [23]. RAMP 2.0 uses the temperature estimates from HotSpot and utilization from Turandot every microsecond to calculate "instantaneous" MTTF of each structure due to each failure mechanism. As described in Section 2, RAMP 2.0 averages these MTTFs to compute the MTTF of each structure due to each failure mechanism

Technology Parameters	
Technology	65nm
V_{dd}	1.0V
Frequency	2.0GHz
Size (without L2)	3.6mm $\times$ 3.2mm
Leakage power density at 383K	0.60 W/mm^2
Base Processor Parameters	
Fetch/finish rate	8 per cycle
Retirement rate	1 dispatch group (=max of 5) per cycle
Functional units	2 Int, 2 FP, 2 Load-Store, 1 Branch, 1 LCR
Integer FU latencies	1 add, 7 multiply, 35 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	
Data L1	32KB
Instruction L1	32KB
L2 (Unified)	2MB

Table 4.1 Base processor simulated.

over the entire execution of the application. Then, using 10^7 Monte-Carlo trials with a Min-Max analysis, RAMP 2.0 computes the MTTF and nTTF values for the entire processor. For our experiments, the processor is divided into the following seven structures: instruction fetch unit, branch prediction unit, instruction decode unit, instruction scheduling unit, fixed point unit, floating point unit and the load store unit.

4.3 Workloads

We evaluate 16 SPEC2000 benchmarks (8 SpecInt + 8 SpecFP). The SPEC2000 trace repository used in this study was generated using the Aria trace facility in the MET toolkit [19] using a 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 [12].

4.4 Reliability Enhancements

As mentioned earlier, we study two reliability enhancements to the base processor - structural duplication (SD) and graceful performance degradation (GPD). Our methodology is identical to that used in [26].

For SD, the seven structures of the processor (Section 4.2) are grouped into five groups, each of which can be duplicated for standby or spare redundancy. The cost overhead of this area increase is evaluated using the Hennessey-Patterson die cost model [10], and reported as the ratio of the cost of the SD and base processors. For GPD, the structures are grouped into four groups, each of which is allowed to degrade to half performance without the entire processor failing. The net performance overhead of a GPD configuration is reported as the slowdown incurred by the fully degraded version of that configuration for the entire application (relative to the base configuration).[1] SD with a cost overhead of X and GPD with a performance overhead of Y are denoted by SD-X and GPD-Y respectively. In each case, we use the system configuration that gives the most benefit in MTTF for the specified overhead for the given application.

4.5 Metrics

In this paper, we study the relationship between MTTF and nTTF for n values of 0.1%, 0.5%, 1% and 5%. The specific failure rates targeted by the industry are, in general, confidential and hard to obtain. We found documentation for one scenario that appears to target 1% failure over the operational lifetime for Intel LXT971A Fast Ethernet Transceiver [11]. Another document mentions that 0.1% and 1% failure times are more relevant to processors than MTTF but fails to provide any additional information [18]. In addition to these failure rates, we study nTTF for a 0.5% failure rate to fill the space between 0.1% and 1%. We use 5% failure rate as an upper bound for the target failure rate of a population of processors as a 5% failure rate corresponds to a failure probability of ≤ 0.05, which is a high failure probability for the microprocessor industry.

[1] The performance of the fully degraded version of a given GPD configuration is the guaranteed performance. In reality, the early part of the lifetime will see better performance. Srinivasan et al. report both the guaranteed and the actual performance [26]. We chose to report only the former since the method of counting this overhead is orthogonal to the point of this paper.

CHAPTER 5

RESULTS

5.1 Application Impact on nTTF

Solutions that tackle the reliability problem at the microarchitecture level have an advantage over solutions that work at the circuit or device level by being application aware. Since applications utilize various parts of the processor differently, the reliability characteristics of different applications are significantly different. Previous work has shown that this variation in application characteristics results in significantly different MTTFs for different applications [24]. Thus, designs that assume worst-case operation are too conservative and may incur unnecessary performance and/or cost overheads This motivates techniques like Dynamic Reliability Management (DRM) that adapt the operational point dynamically by increasing performance when the application is operating well within the reliability targets [13, 24]. In this section, we investigate if DRM type solutions are applicable with the nTTF metric.

Figure 5.1 shows the relative values of nTTF and MTTF for 8 SpecFP and 8 SpecInt applications in the base system. The values are relative to the average MTTF across all applications in the base system. The applications are ordered in increasing order of MTTF. The figure shows that there is a large variation in both MTTF and nTTF, for all n, across the different applications (30% standard deviation as a percentage of the mean for both MTTF and nTTF).

These results show that even under the nTTF metric, the reliability characteristics are application specific. The large application specific variation shows that designing to the worst-case may result in incurring unnecessary overhead in terms of performance and/or cost to ensure that the target reliability is always attained. For example, based on worst-case analysis, one would design the system with the reliability characteristics of the application *gap* as the target reliability would be achieved even when the system runs any

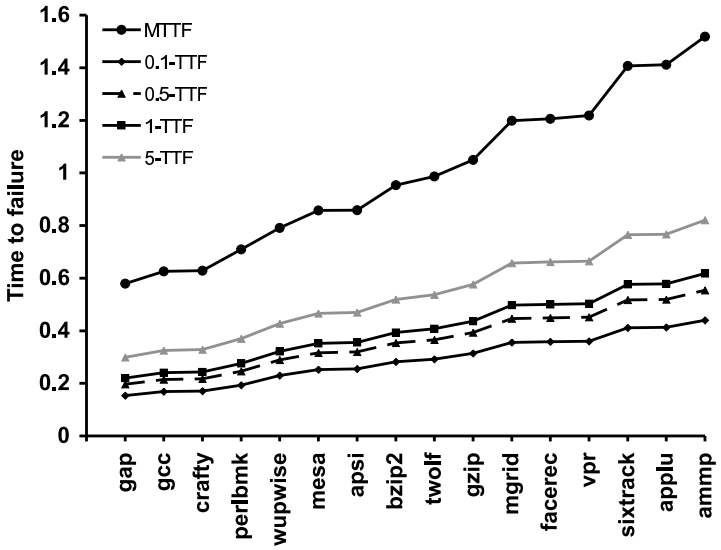


Figure 5.1 Relative MTTF and nTTF values for various applications running in the base system. All values are relative to the average MTTF across all applications. Similar to MTTF, nTTF, for all n, shows a significant variation across applications motivating application-aware dynamic reliability management.

other application. However, this design is an over-kill when an application like *ammp* runs on this system as the system would be well above the target reliability, incurring unnecessary performance and/or cost penalties.

This difference motivates application-aware reliability management even when using nTTF as a metric, instead of MTTF. The benefits achieved by implementing methods like Dynamic Reliability Management to gain extra performance or power savings are also applicable when using nTTF as the metric.

The similarity in the variation of nTTF and MTTF for the base system raises the question of whether the nTTF of the system can be estimated by using the MTTF of the base system. The subsequent sections probe the relationship between these metrics further.

5.2 Relationship Between MTTF and nTTF

For the lognormal distribution, we find empirically that the ratio of nTTF to MTTF is a function of only n and the shape parameter (σ). Our base system is a series failure system, i.e., the system fails when the first component in the system fails, where the underlying structure lifetimes are lognormally distributed

(for a given failure mechanism). We investigate if a proportionality relationship similar to the lognormal distribution exists even for such a system. Systems that employ redundancy to enhance reliability are no longer series failure systems like the base system, but are series-parallel failure systems. We investigate if the relationship of nTTF of these systems can be predicted from the nTTF-MTTF relationship of the base system.

If such prediction could be made accurately, then MTTF would be a reasonable proxy for nTTF; e.g., when comparing the benefits of two reliability-enhancing techniques, the improvement in MTTF would be representative of the improvement in nTTF.

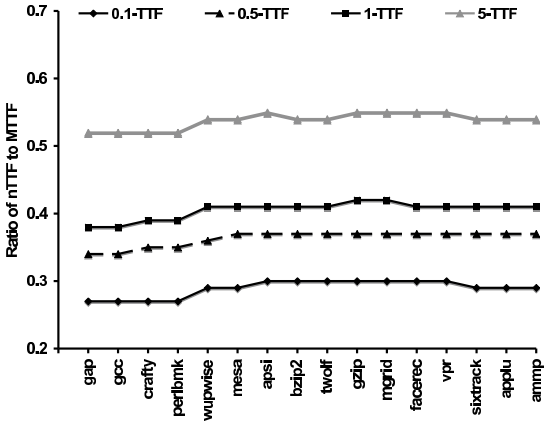
Figure 5.2 shows the ratio of nTTF to MTTF for all applications for n=0.1%, 0.5%, 1% and 5%, in the (a) Base, (b) GPD-1.11X, (c) GPD-2X, (d) SD-1.75X and, (e) SD-2.25X systems. These systems show a representative mix of varying overheads in performance and area to enhance the system reliability. Additionally, the inferences drawn from these GPD and SD systems are similar to those drawn from the other systems. Hence, results for only these systems are shown.

The figure shows that the ratio of nTTF to MTTF, for a given n, is both system and application dependent.

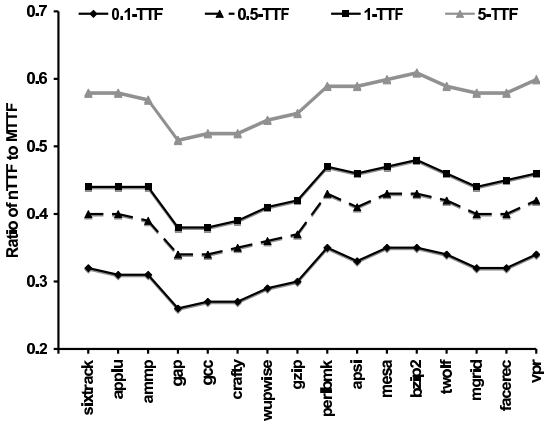
The base system is a series failure system, with each components having lognormally distributed failure times for a given failure mechanism. The lifetime of the entire system in each Monte Carlo trial is computed as a min of the lifetimes of each component. Hence, the resulting distribution of the failure times of the system is a min of many lognormals, which, in general, will not be a lognormal distribution. Thus, the ratio of nTTF to MTTF for the base system is not the same as that of the lognormal distribution.

The base system is a series failure system. In order to enhance the reliability of the system, GPD allows for some components to work in degraded mode, i.e., in half configuration. SD on the other hand employs spares which are switched on when the main component fails. The nature of the resulting distribution, and hence the ratio of nTTF to MTTF, is governed by the composition of such a system and the order in which the components fail. Since the utilization of the different components in the system is application specific, the component that fails first is application specific, resulting in the distribution of failures different for different applications even within the same system. Hence, the ratio of nTTF to MTTF for various systems are different from each other and this ratio also shows a variation across different applications.

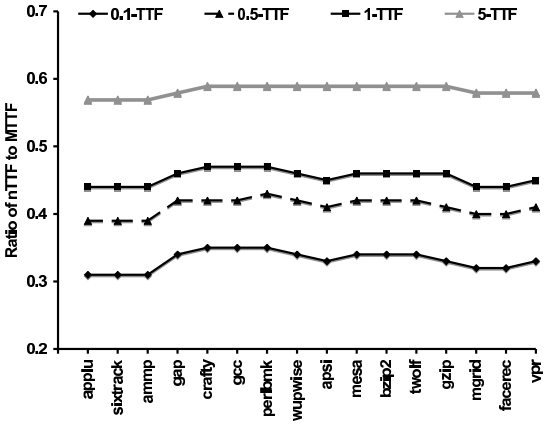
Oblivious to such a difference across different systems, one may attempt to estimate the nTTF of a particular system with the knowledge of its MTTF, assuming that the ratio is similar to that of the base



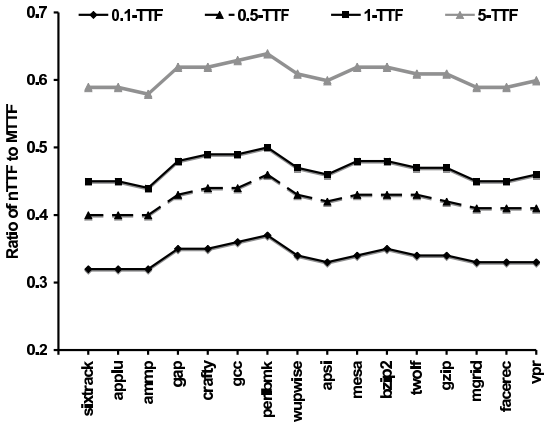
(a) Base



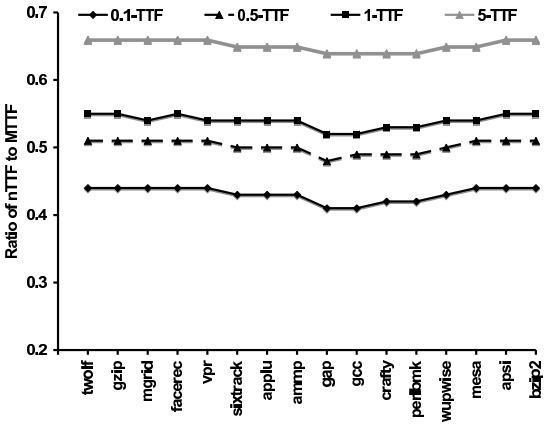
(b) GPD-1.11X



(c) GPD-2X



(d) SD-1.75X



(e) SD-2.25X

Figure 5.2 Ratio of nTTF to MTTF for (a) Base, (b) GPD-1.11X, (c) GPD-2X, (d) SD-1.75X, and (e) SD-2.25X systems. The ratio of nTTF to MTTF, for various n, is dependent on both the application and the system making nTTF hard to predict from MTTF.

System	Error in estimation				MTTF benefit over base
	0.1-TTF	0.5-TTF	1-TTF	5-TTF	
GPD-1.05X	-6.72%	-6.14%	-5.45%	-4.49%	1.32
GPD-1.11X	-7.28%	-7.09%	-6.84%	-5.24%	1.40
GPD-1.25X	-9.84%	-9.32%	-8.99%	-6.57%	1.53
GPD-1.43X	-11.38%	-10.56%	-10.09%	-7.24%	1.58
GPD-1.67X	-12.89%	-11.66%	-10.96%	-8.01%	1.63
GPD-2X	-12.75%	-11.54%	-10.85%	-7.82%	1.63
SD-1.25X	0.03%	0.17%	0.15%	0.00%	1.00
SD-1.5X	2.90%	3.01%	3.18%	2.62%	1.03
SD-1.75X	-14.69%	-13.99%	-13.42%	-11.38%	1.66
SD-2X	-25.79%	-23.73%	-22.09%	-16.97%	1.88
SD-2.25X	-32.96%	-27.60%	-24.86%	-17.42%	2.04

Table 5.1 Average percentage error in estimating nTTF from ratio of nTTF to MTTF for the base system. The error in estimation can be significant and may lead to over-designed systems.

system. This could lead to significant errors as the ratios are both application and system specific.

Table 5.1 gives the percentage error in estimating the nTTF assuming that the nTTF for an application on that system is k$\times$MTTF for that system, where k is the ratio of nTTF to MTTF of that application for that n in the base system. The percentage error for each system averaged across all applications is given.

We notice that the error in estimating the nTTF based on the ratio of nTTF to MTTF for the base system mostly results in under-estimating the values of nTTF. However, there do exist some cases where the average error (and even the application specific errors) are positive.

This error in the estimation can be explained based on the way in which the SD and GPD systems improve the reliability of the system. The improvement primarily comes from allowing the structure with the lowest MTTF to operate in half-degraded mode (in the case of GPD) or to have a spare (in SD). In such a case, the time at which this component in the system fails is improved because of degradation or sparing. Thus, the time at which the first component fails improves, improving the nTTF at a rate higher than the MTTF. However, when the structure with the lowest MTTF is not degraded or spared, this structure fails first a larger number of times. However, the nTTF of the system does improve since the failure times of the component that is degraded or spared increases. This results in only a marginal improvement in nTTF, making the benefit in nTTF lower than the benefit in MTTF, resulting in a positive error in the estimation.

Hence, we see that, the errors in estimating the nTTF using the base system's nTTF to MTTF ratio results in significant errors for both SD and GPD systems. Thus, when the lifetimes of a system are computed

based on lognormal failure distributions and the system employs redundancy, predicting the nTTF from MTTF based on ratios from the base system can result in significant errors (upto 33% in our experiments). Furthermore, this error is different for different systems and for differing values of n.

5.3 Improvement in MTTF versus nTTF

The previous section showed that predicting the value of nTTF from MTTF requires additional information about the configuration of the system and the application as the ratio is is dependent on both these parameters. However, designers and users are often concerned with relative comparisons among systems; therefore, we next ask the question whether the *improvement* in nTTF from a reliability-enhancing technique can be predicted from the improvement in MTTF. We use the techniques of SD and GPD for this purpose.

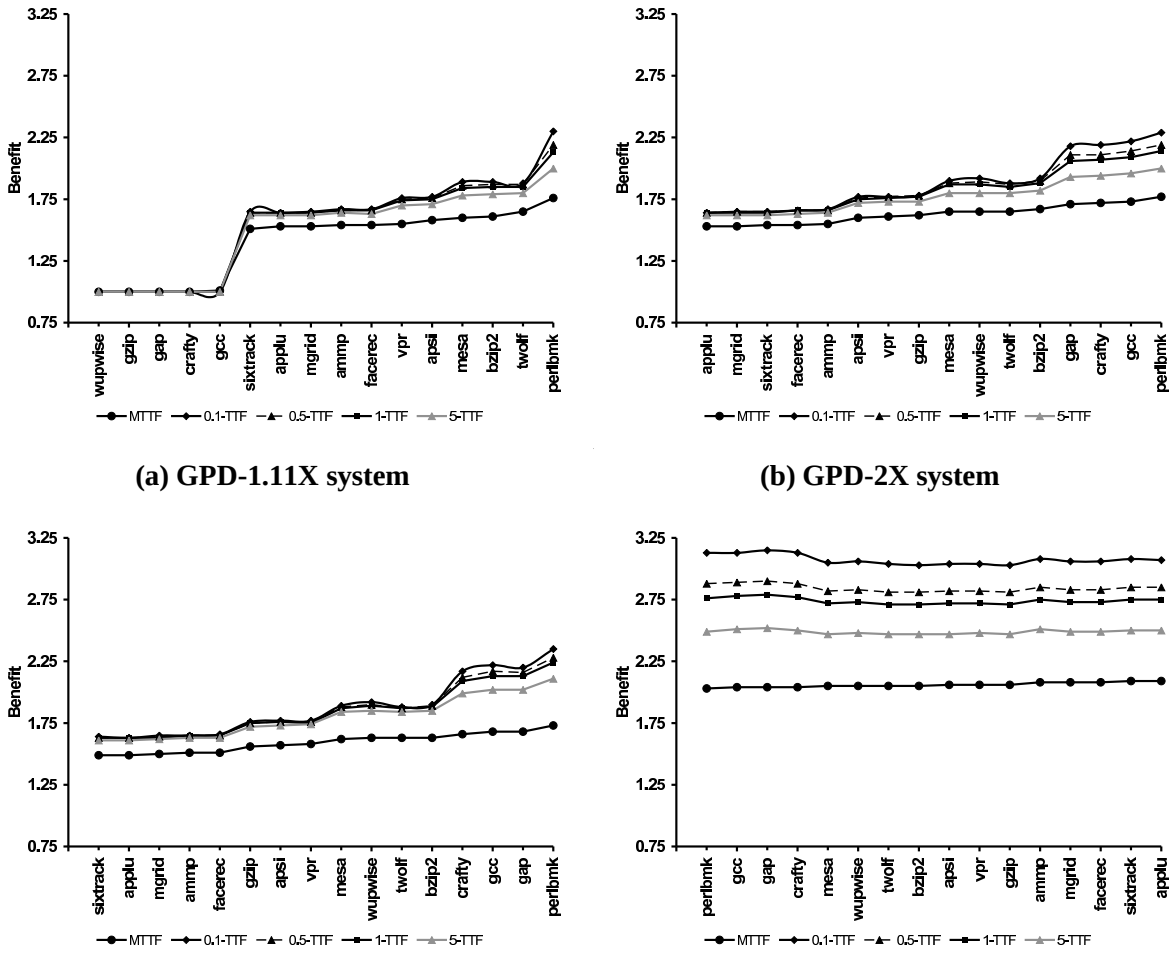
Figure 5.3 shows the benefit in MTTF and nTTF for (a) GPD-1.11X, (b) GPD-2X, (c) SD-1.75X and (d) SD-2.25X systems for different applications. The benefit in nTTF (or MTTF) is computed as the ratio of the nTTF (or MTTF) of the GPD or SD system over that of the base system. For each system, the applications are ordered in increasing order of improvement of MTTF over the base system.

The figures show that the benefit in MTTF does not correspond to an identical benefit in the nTTF. This benefit in the nTTF is dependent on both the application and the system, with the benefits in nTTF being higher than the benefits in MTTF in most cases.

For example, consider the GPD-1.11X system shown in Figure 5.3(a). Under this performance overhead, the application *gap* shows no benefit in nTTF or MTTF while the application *mgrid* shows a 1.5X benefit in MTTF and a 1.6X benefit in nTTF, for all n shown. The application *perlbmk* shows a 1.7X improvement in MTTF and the benefits in nTTF are higher than 2X, with varying benefits for different n. Thus, even for a given performance overhead, we see that the benefits in MTTF and nTTF are application specific.

This benefit is also markedly varied across different systems. For example, in the SD-1.75X system, the application *apsi* shows a 1.5X improvement in MTTF and a 1.7X improvement in nTTF for various n. However, the same application, in the SD-2.25X system shows a 2X improvement in MTTF and a benefit of larger than 2.5X for all n shown. This shows that the benefit in nTTF, in addition to being application specific, is also dependent on the configuration of the system.

The nature of the graphs in Figure 5.3 are a result of the composition of each system and the order in which the various components in the system fail. We note that the LSU, being the largest structure in the



(a) GPD-1.11X system **(b) GPD-2X system**

(c) SD-1.75X system **(d) SD-2.25X system**

Figure 5.3 Benefit in MTTF and nTTF for various n, for (a) GPD-1.11X, (b) GPD-2X, (c) SD-1.75X, and (d) SD-2.25X. For each system, the applications are ordered in increasing order of benefit in MTTF. The benefit in MTTF and nTTF are application specific, and for the SD systems, benefits in MTTF cannot be used to predict benefits in nTTF.

processor, has the lowest MTTF among all components and hence fails first in over 80% of the Monte Carlo trials for all applications in the base system. The benefit in nTTF and MTTF for the various systems then have a strong correlation with whether the LSU is allowed to degrade (in GPD systems) or be replicated (in SD systems) under the given performance/area overhead. For example, in the GPD-1.11X system, the applications that show no benefit in the MTTF or nTTF do not degrade the LSU, while the systems that degrade the LSU (like applu, mgrid, etc.) show a benefit in both nTTF and MTTF. Additionally, the various perturbations in the figures stem from a difference in the composition of components that fail first in the Monte Carlo trials. The SD-2.25X system (show in Figure 5.3(d)) is inherently different from the other systems shown. In the SD-2.25X system, all components are allowed to have spares. Thus, the system's lifetime in each Monte Carlo trial is computed as a min of the sum of lognormals, which is inherently a different distribution from that of the other systems. Hence, the SD-2.25X system shows a significantly different behavior for the nTTF benefits.

Thus, in addition to the inability to predict the values of nTTF from MTTF (Section 5.2), one cannot reliably predict the benefits in nTTF from the benefits in MTTF. Further, these benefits are application specific.

From this, we can infer that designing systems with MTTF as a proxy for nTTF can potentially lead to poor design choices. The next section provides experimental data to show such scenarios.

5.4 Implications for Reliability-Aware Design

The previous sections have shown that when the underlying distributions of failures is assumed to be a log-normal, and the lifetime of the system is computed as a series or series-parallel system, the relation between nTTF and MTTF is complex and not easily predictable. This may have a marked impact on Reliability-Aware design as the use of MTTF as a proxy to nTTF may result in poor design choices. This section illustrates the impact of using MTTF as a proxy for nTTF when considering alternative reliability-enhanced designs. Again, we use SD and GPD as our example reliability enhancements.

Figure 5.4 shows the benefit under different metrics (MTTF, 0.1-TTF, 0.5-TTF, 1-TTF, 5-TTF) for SD and GPD with various overheads for various applications. Figure 5.5 shows the average benefits, with Figure 5.5(a) showing the benefits averaged across the SpecFP applications, and Figure 5.5(b) showing the averages for SpecInt applications. The solid and dotted lines respectively represent SD and GPD systems.

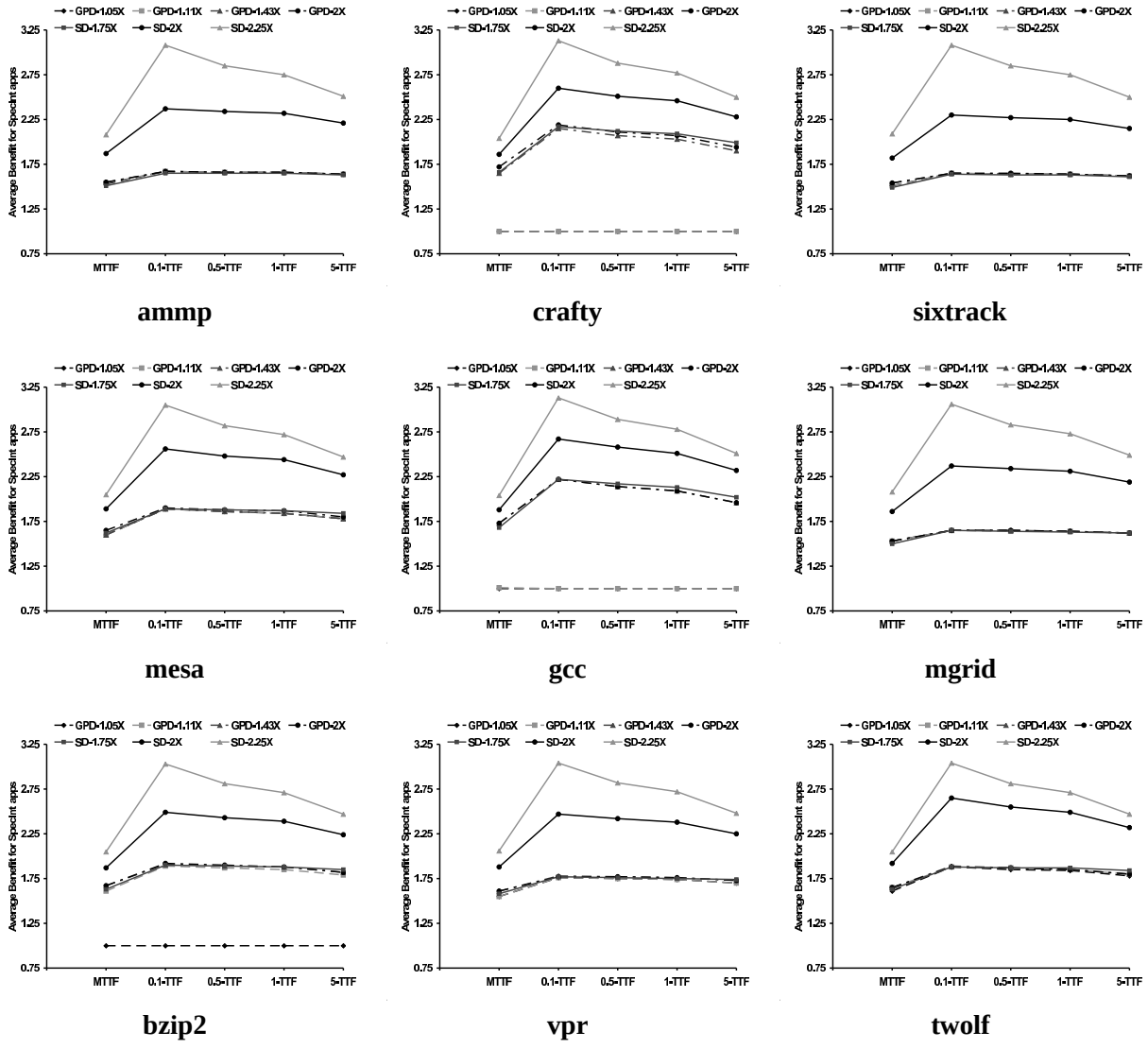
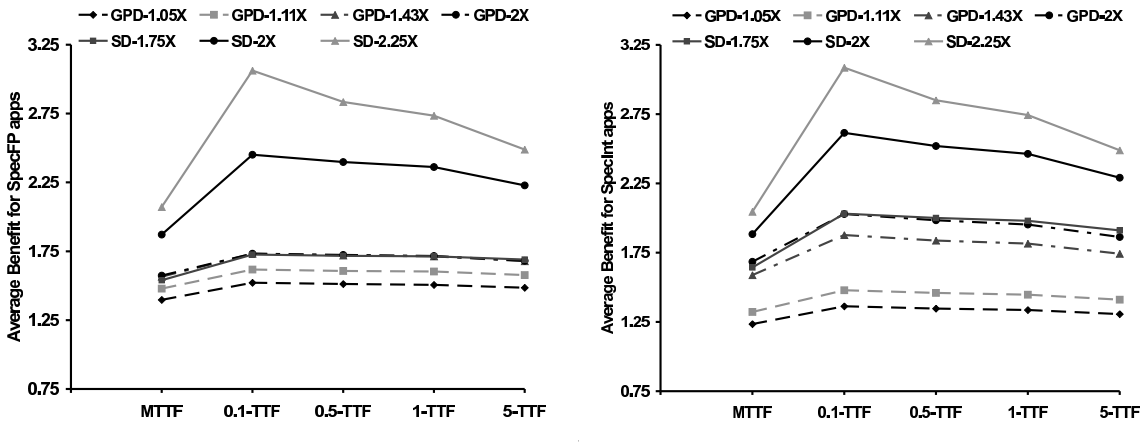


Figure 5.4 Impact of using MTTF instead of nTTF on application specific reliability-aware design. Each figure shows the benefit under different metrics (MTTF, 0.1-TTF, 1-TTF, 5-TTF and 10-TTF) for reliability-aware systems with different overheads. The solid and dotted lines respectively represent SD and GPD systems. The system with the lowest cost to achieve the target reliability is, in general, different for different metrics. This may lead to poor design choices.



(a) SpecFP average benefit **(b) SpecInt average benefit**

Figure 5.5 Impact of using MTTF instead of nTTF on reliability-aware design for average (a) SpecFP and (b) SpecInt applications. Each figure shows the benefit under different metrics (MTTF, 0.1-TTF, 0.5-TTF, 1-TTF and 5-TTF) for reliability-aware systems with different overheads. The solid and dotted lines respectively represent SD and GPD systems. The system with the lowest cost to achieve the target reliability is, in general, different for different metrics. This may lead to poor design choices.

The optimal system that achieves the required reliability target, under any given metric, is that system that has the lowest cost (for a performance bound design) or the highest performance (for a cost bound design) or both. The figure shows that the system with the lowest cost that achieves a target reliability benefit is markedly different for different metrics.

For example, consider the case of a performance bound designer targeting a 2X improvement in 0.1-TTF on a system that runs mostly integer workloads. If the designer wants to use MTTF as a proxy for 0.1-TTF (since computing MTTF is simpler), the designer would try to achieve a 2X improvement in MTTF with the lowest possible cost. From Figure 5.5(b), we see that the only system that achieves this target reliability benefit in MTTF is the SD-2.25X system. However, if the designer had designed to 0.1-TTF, SD-2X would be the chosen design for this benefit in 0.1-TTF as it is the lowest cost solution.

Thus, an incorrect projection of a 2X benefit in 0.1-TTF to a 2X benefit in MTTF results in an over-designed system with a higher cost. Similar cases can also be seen for other metrics in both the average SpecInt and SpecFP cases.

A cost bound designer, on the other hand, attempts to achieve the required reliability targets while incurring the lowest performance overhead possible. Consider one such designer who is targeting a 2X improvement in 0.1-TTF. If this target is inferred as a 2X improvement in MTTF, Figure 5.5(b) shows that

for systems running integer benchmarks, no GPD system gives this MTTF improvement. Thus, the designer would mark this reliability target as unachievable with any GPD or SD system. However, in reality, the GPD-2X system achieves this target under 0.1-TTF.

Thus, using MTTF as a proxy for nTTF to perform reliability-aware system design can lead to incorrect design decisions for reliability-aware designs. Systems with an unnecessarily high overhead may be chosen to achieve the desired target while a system with a significantly lower overhead would have sufficed. Thus, designing with MTTF as a proxy for nTTF can result in significant design discrepancies.

5.5 Lognormal vs. Exponential Distributions

Our results so far show that using MTTF instead of nTTF can have a significant implication for design. The reason for the difference between MTTF and nTTF based designs is because we use lognormal lifetime distributions for the underlying components of the system. Combining component-wise metrics to get a system-wide metric in this case is analytically hard and results in a complex relationship between nTTF and MTTF that does not have a closed form [20]. However, as discussed earlier, this is not the case for exponential component-wide distributions, which result in exponential system-wide distributions and a closed form between MTTF and nTTF [20]. Although exponential distributions are known to not adequately represent wear-out behavior, much work in industry makes the assumption of exponential distributions due to its simplicity [15].

We therefore next analyze the complexity vs. accuracy trade-off between the exponential and lognormal distributions. We estimate the exponential nTTF using (1) the SOFR-derived MTTF from the simpler RAMP 1.0 and (2) the closed form relationship between MTTF and nTTF for exponential distributions (Section 4). We then report the error in this nTTF relative to the nTTF derived from the lognormal assumption with RAMP 2.0.

Figure 5.6 shows this percentage absolute error averaged across all applications for each of 0.1-TTF, 0.5-TTF, 1-TTF, 5-TTF, and MTTF for different systems. (In order to perform a fair comparison, the MTTF values were normalized such that the average MTTF in the base across all applications is 30 years for both RAMP 1.0 and for RAMP 2.0.)

The average errors in nTTF and MTTF when assuming an exponential distribution is significant in all systems, for all n. Thus, although the exponential approximation largely simplifies the computation of nTTF,

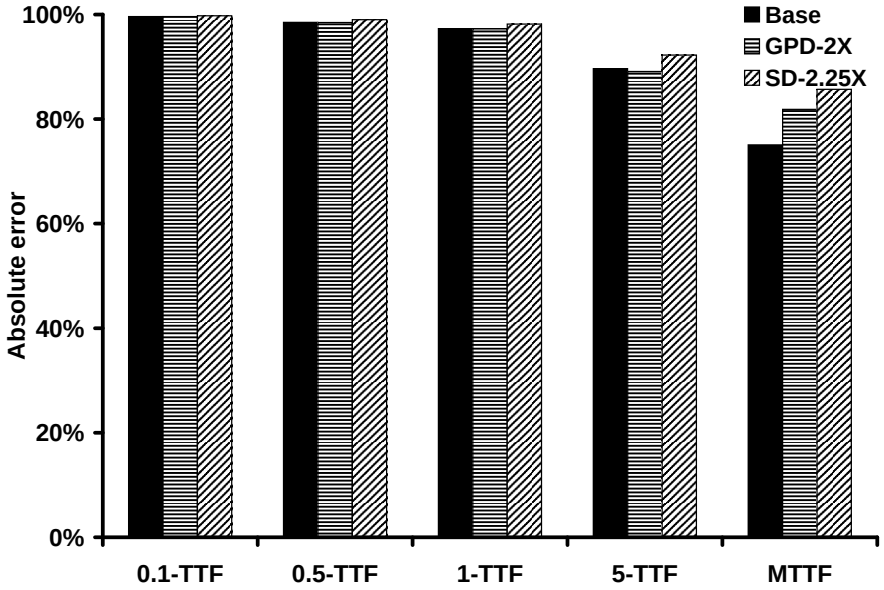


Figure 5.6 Average absolute error in MTTF and nTTF, for different n and different systems, when the underlying failure distribution is assumed to be exponential. The significant error margin in the estimation suggests that the exponential approximation is not valid even when using nTTF as a design metric.

the simplicity comes at a large cost in accuracy.

The errors in nTTF are consistent with our previous data. The exponential methodology computes nTTF as a constant function of MTTF. As seen in Section 5.2, this approximation is not valid. In addition, the constant used for a particular n is $-ln(1 - \frac{n}{100})$ which is significantly smaller than the ratio of nTTF to MTTF, for small n, as seen from Figure 5.2. Hence, the error in the estimation is significant. Additionally, the exponential approximation models a constant failure rate, unlike the potentially more accurate lognormal distribution. Hence, the error in the MTTF estimated using the exponential distribution is also significant.

Thus, our results show that nTTF for different systems has to be computed using first principles – a constant factor approximation applied to MTTF and a simpler failure distribution assumption lead to erroneous results.

CHAPTER 6

CONCLUSIONS

Aggressive CMOS scaling has lead to the onset of wear-out based failures at a rate not seen in the past. While high-end systems in niche high-reliability markets have always been concerned about lifetime reliability, recently this concern has spread to the broader general-purpose market as well. Concerned microarchitects have begun to propose reliability-aware microarchitectures to tackle problems caused by wear-out failures. However, these designs are largely analyzed based on the benefits achieved in MTTF. MTTF, being an average, does not capture sufficient information about the useful operational life of the processor, which is typically much smaller than the mean life. Thus, MTTF can be misleading to both customers and designers as systems with similar MTTFs can have different failure distributions and reliability characteristics for the most useful part of the product's life. We studied nTTF, denoting the time at which the probability of failure is $\leq$ n/100, as an alternative to MTTF.

This thesis, for the first time to our knowledge, analyzes the implication of designing reliability-aware microarchitectures for MTTF as a proxy for potentially more accurate metrics like nTTF. Distribution dependent metrics such as nTTF are inherently significantly more complex to compute, but this work presents a straightforward way of computing such a metric through the use of the state-of-the-art architecture level lifetime reliability tool RAMP 2.0.

Our analysis of the nTTF metric for reliability-aware designs leads us to the following conclusions

- Similar to the MTTF metric, the nTTF metric also motivates application-aware reliability management as the reliability characteristics of different systems are different for different applications.
- The relative value of the nTTF to MTTF is system dependent and cannot be predicted using the nTTF to MTTF ratio for the base system, owing to the complexity involved in computing them. Hence, for a

given system, the nTTF values will have to be calculated from first principles and cannot be predicted from the MTTF.

- The system with the lowest cost overhead (in terms of performance or area) chosen to achieve the target reliability is markedly different for different systems. Hence, the use of incorrect metrics may result in non-optimal designs, making it necessary to understand and target the right reliability metrics for reliability-aware design.

Admittedly, current architecture level tools for lifetime reliability are still preliminary and make significant assumptions. Nevertheless, we take an important step in understanding the right metrics that should be targeted by such tools. It quantifies the impact of using the widely used MTTF metric, instead of more accurate metrics, for reliability-aware processor design.

Additionally, nTTF may not be the only metric that designers are concerned about. Other metrics such as mean time to repair, etc. are also important. Although this work does not analyze all such metrics, it establishes an first important step to move beyond the widely used but potentially misleading MTTF for architects.

REFERENCES

[1] Methods for Calculating Failure Rates in Units of FITs. Technical Report JESD85, JEDEC Publication, 2001.

[2] Critical Reliability Challenges for The International Technology Roadmap for Semiconductors. Technical Report 03024377A-TR, International Sematech Tech. Transfer, 2003.

[3] Failure Mechanisms and Models for Semiconductor Devices. Technical Report JEP122-C, JEDEC Publications, March 2006.

[4] S. Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[5] F. A. Bower, P. G. Shealy, S. Ozev, and D. J. Sorin. Tolerating hard faults in microprocessor array structures. In *Proceedings of International Conf. on Dependable Systems and Networks*, 2004.

[6] F. A. Bower, D. J. Sorin, and S. Ozev. A mechanism for online diagnosis of hard faults in microprocessors. In *Proceedings of the 38th Annual International Symposium on Microarchitecture*, 2005.

[7] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, 2000.

[8] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6), 2000.

[9] B. Greskamp, S. Sarangi, and J. Torrellas. Threshold Voltage Variation Effects on Aging-Related Hard Failure Rates. In *International Symposium on Circuits and Systems (ISCAS), Special Session: Circuit Design in the Presence of Device Variability*, May 2007.

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

[11] Intel. LXT971A Fast Ethernet Transceiver. `http://www.intel.com/design/network/products/lan/phys/lxt971a-972a.htm`.

[12] V. S. Iyengar, L. H. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proceedings of the 2nd IEEE Symposium on High-Performance Computer Architecture*, 1996.

[13] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Reliability modeling and management in dynamic microprocessor-based systems. In *Proceedings of the 43rd annual conference on Design automation*, 2006.

[14] Z. Lu, J. Lach, M. R. Stan, and K. Skadron. Temperature-Aware Modeling and Banking of IC Lifetime Reliability. *IEEE Micro special issue on Reliability-Aware Microarchitectures*, 25(6):40–49, Nov./Dec. 2005.

[15] W. Q. Meekar and L. A. Escobar. *Statistical Methods for Reliability Data*. John Wiley and Sons, 1998.

[16] C. Moore. The POWER4 System Microarchitecture. In *Microprocessor Forum*, 2000.

[17] Reliability in CMOS IC Design: Physical Failure Mechanisms and their Modeling. MOSIS Technical Notes, http://www.mosis.org/Faqs/tech_cmos_rel.pdf.

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

[19] M. Moudgill, P. Bose, and J. Moreno. Validation of Turandot, a Fast Processor Model for Microarchitecture Evaluation. In *Proceedings of International Performance, Computing and Communications Conf.*, 1999.

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

[21] ReliaSoft. Limitations of the exponential distribution for reliability analysis. In *Reliasoft Newsletter – http://www.reliasoft.com/newsletter/4q2001/exponential.htm*, 2001.

[22] Sidharth and S. Sundaram. A methodology to assess microprocessor fan reliability. In *The 9th Intersociety conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2004.

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

[24] J. Srinivasan, S. Adve, P. Bose, and J. A. Rivers. The Case for Microarchitectural Awareness of Lifetime Reliability. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*, 2004.

[25] J. Srinivasan, S. Adve, P. Bose, and J. A. Rivers. The Impact of Scaling on Processor Lifetime Reliability. In *Proceedings of the International Conf. on Dependable Systems and Networks*, 2004.

[26] J. Srinivasan, S. Adve, P. Bose, and J. A. Rivers. Exploiting Structural Duplication for Lifetime Reliability Enhancement. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, 2005.

[27] J. M. Tendler, J. S. Dodson, J. S. F. Jr., and B. S. H. Le. POWER4 system microarchitecture. *IBM Journal of Research and Development*, 46(1), 2002.

[28] D. C. Trindade and S. Nathan. Analysis of field data for repairable systems. In *Proceedings of the Annual Reliability and Maintainability Symposium*, 2006.

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

[30] A. Wood. Reliability-metric varieties and their relationships. In *Proceedings of Annual Reliability and Maintainability Symposium*, 2001.

[31] D. Yen. Chip multithreading processors enable reliable high throughput computing. In *International Reliability Physics Symposium (Key-note)*, 2005.