Metrics for Architecture-Level Lifetime Reliability Analysis

Pradeep Ramachandran[†], Sarita V. Adve [†], Pradip Bose[‡], and Jude A. Rivers[‡]

[†]Department of Computer Science
University of Illinois at Urbana-Champaign
{pramach2, sadve, srinivsn}@uiuc.edu

[‡]IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{jarivers, pbose}@us.ibm.com

***Abstract*— This work concerns metrics for evaluating microarchitectural enhancements to improve processor lifetime reliability. A commonly reported reliability metric is mean time to failure (MTTF). Although the MTTF metric is simpler to evaluate, it does not provide information on the reliability characteristics during the relatively short operational life of commodity processors. An alternate metric is nTTF, which represents the time to failure of n% of the processor population. nTTF is a more informative metric for the (short) portion of the lifetime that is relevant to the end-user, but determining it requires knowledge of the distribution of processor failure times which is generally hard to obtain.**

The goals of this paper are (1) to determine if the choice of metric has a quantitative impact on architecture-level reliability analysis and modern superscalar processor designs and (2) to build a fundamental understanding of why and when MTTF- and nTTF-driven analysis result in different designs. We show through an in-depth analysis that, in general, the nTTF metric differs significantly from the MTTF metric, and using MTTF as a proxy for nTTF leads to sub-optimal designs. Additionally, our analysis introduces the concept of relative vulnerability factor (RVF) for different processor components to guide reliability-aware design. We show that the difference between nTTF- and MTTF-driven design largely occurs because the relative vulnerabilities of the processor components change over the processor lifetime, making the optimal design choice dependent on the amount of time the processor is expected to be used.

I. 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) [3], [26], [33].

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 [3], [5], [4], [11], [20], [27], [25], [33]. 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. A commonly reported 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 [25], [28].

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 [17], [30], [32]. The following example illustrates this limitation of MTTF.

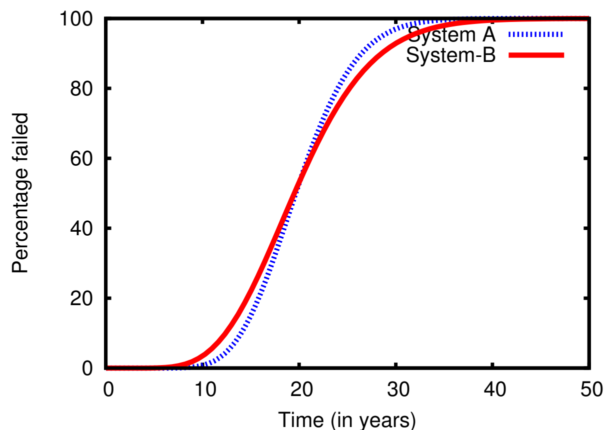
Figure 1(a) shows the cumulative distribution function (CDF) of the lifetimes for two simulated product lines (the methodology for this simulation is explained in Section IV). The MTTF for both the product lines is 20 years, a typical MTTF target for processor designers [13]. However, the typical operational life for a processor is much shorter, e.g., of the order of 5 to 7 years. Figure 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.

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, 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 sub-optimally designed systems.

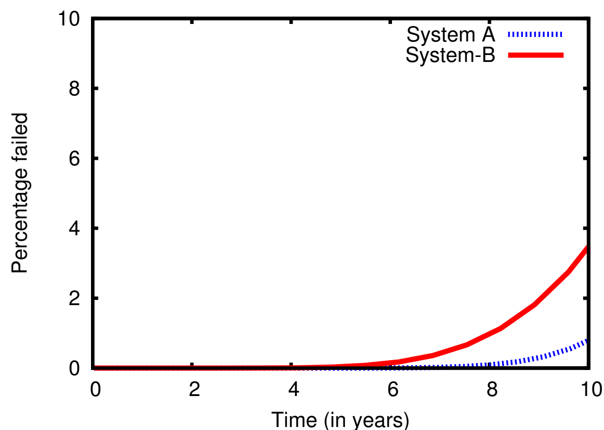
A. The nTTF Metric

This paper explores nTTF – an alternate metric for reliability-aware designs, and studies its relation to the commonly used MTTF. Mathematically, nTTF is the time 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 large processor population. Alternately, by time t, the probability of failure of a processor is $\frac{n}{100}$.

This work is supported in part by an IBM faculty partnership award, the Gigascale Systems Research Center (funded under FCRP, an SRC program), the National Science Foundation under Grant NSF CCF 05-41383 and an equipment donation from AMD.



(a) Lifetime CDF for 50 years



(b) Closer view of the first 10 years

Fig. 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 5 to 10 year usage as it sees fewer failures in its early life. MTTF fails to provide such information.**

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 ≤ 5 years is 0.01.[1] nTTF, for various values of n, accurately represents the distribution of failure times, 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.

B. Contributions of this paper

This paper provides a quantitative evaluation of nTTF and its relationship to MTTF for lifetime reliability enhancing techniques for modern superscalar architectures. It shows (1) that the choice of metric has a quantitatively significant impact on architecture-level reliability analysis and modern superscalar processor design and (2) a fundamental reason for the difference between MTTF- and nTTF-driven designs is the change in relative vulnerabilities of different microarchitectural components over time.

[1]Note that for non-uniform distributions, like the exponential and the lognormal distributions, 50-TTF does not correspond to the MTTF.

To achieve our goals, we 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. We then 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.

We find that, in general, 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 complex and different for various applications running on the same system and across different systems.

Our results also show that designs that are optimal for MTTF may not be optimal for nTTF, and vice-versa. In particular, 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.

To explain the difference in design decisions under the different metrics, we introduce the notion of the *relative vulnerability factor (RVF)* for the different processor components. The RVF of a component indicates its vulnerability to wear-out, relative to other components in the system. Hence, the highest reliability benefit is achieved by enhancing the reliability of the components with the highest RVF (e.g., introducing spares for such components). However, due to the complexity of the lifetime distributions of the systems we study, we find the RVFs change depending on how long the system is expected to be used. Thus, depending on the metric used (which effectively determines the system lifetime that is considered for making design decisions), we see that the components with the highest RVF are different

for different metrics, resulting in different design choices when considering alternate metrics.

Overall, our analysis of the different metrics shows that the choice of metric has a significant impact on the architecture-level design decisions for lifetime reliability and quantifies this impact for modern superscalar processors.

II. Related Work and Background

Section II-A describes related work on alternate metrics. Section II-B provides background on RAMP [25], [28], a state-of-the-art architecture level reliability model and tool we use in this study. Section II-C provides background on the two reliability-enhancing techniques we study here – Structural Duplication (SD) and Graceful Performance Degradation (GPD) [27].

A. Reliability Metrics

MTTF (or MTBF – mean time between failures) is a widely used reliability metric in both marketing literature for the computing industry and academic papers evaluating different wear-out tolerant architectures [2], [7], [18], [19], [24], [28], [32].

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 [31]. It is widely accepted that wear-out based failures do not follow exponential lifetime distributions and have failure rates that change with time [31] – typical wear-out failure mechanisms start with a low failure rate at the beginning of the component's lifetime and have an increasing failure rate as the component ages. We assume the more realistic non-exponential model and so do not use FITs. [2]

The nTTF metric explored here is widely used in the automotive industry, where it is commonly referred to as B-life, and for mechanical devices like fans, where it is referred to as L-life [32] (L_{10} is a commonly used metric [21]). 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) [15]. The metric is also directly related to the Mission Time Function, which gives the time at which the system reliability falls below a given level [22]. Some papers from the computer industry also describe this metric [13], [30], [32] (sometimes referred to as t_n [13]), stating that it is more meaningful than MTTF for short operational life, but [30], [32] state that it is not commonly used.

However, none of the above provide a quantification of the impact of using nTTF vs. MTTF on the design of modern superscalar processors. We present such a quantification and an in-depth analysis of these metrics. While reliability experts are already aware of the differences between nTTF and MTTF, we believe that this work, for the first time, quantitatively demonstrates the impact of these differences on the design of modern processors, and builds a fundamental understanding for architects performing early-stage reliability analysis. It additionally demonstrates how nTTF can be measured at the architecture-level with current evaluation tools.

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 [32]. Most of these metrics are, however, based on analysis of field data and is beyond the scope of this work.

B. RAMP

RAMP [25], [28] 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) [1]. It works in conjunction with a timing simulator, a power model, and a temperature model (in our case, Turandot [14], PowerTimer [6], and HotSpot [23], respectively).

RAMP starts with state-of-the-art device level analytical models for each of the above failure mechanisms. These models were derived from a deep understanding of the physics behind each of the failure mechanisms. The original 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 level of microarchitectural structures (e.g., ALU, etc.), and incorporates workload-driven changes in the different parameters over time (obtained from the timing simulator). At the end of the timing simulation, RAMP reports the MTTF of individual microarchitectural structures for each failure mechanism.

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. We use the state-of-the-art RAMP 2.0, which uses lognormal distributions [27]. As discussed earlier, an exponential distribution with a constant failure rate is often used for its analytical simplicity, but is widely known to be inaccurate. Lognormal distributions have been shown to be more accurate for wear-out processes common to semiconductor materials due to the multiplicative degeneration argument [15], [27]. 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 individual distributions are fully specified by their means (provided by the previous step using the timing simulator) and σ, the shape parameter for the lognormal

[2] For a quantification of errors incurred from exponential distribution relative to a more realistic distribution, see [16].

distribution, which is set at 0.5 (used for wear-out failures in prior work [1], [15]).

The use of Monte Carlo simulation also allows RAMP 2.0 to calculate the MTTF of a full system that incorporates redundancy at the level of microarchitectural structures for reliability enhancement (Section II-C). Specifically, RAMP 2.0 can model series failure systems (i.e., no redundancy) and series-parallel redundant and/or standby redundant systems, using a MIN-MAX analysis on the individual component lifetimes in a given Monte Carlo trial [27].

C. Reliability-Enhancing Techniques

We examine two reliability-enhancing methods based on structure-level redundancy explored by Srinivasan et al. [27].

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 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 overhead is incurred to improve the reliability. We will explore SD and GPD configurations with various overheads.

III. INCORPORATING nTTF INTO RAMP

Mathematically, nTTF is defined as the time t at which

$$F(t) = \frac{n}{100} \quad (1)$$

where $F(t)$ is the cumulative distribution function (CDF) for the processor lifetime [31].

We calculate nTTF with RAMP 2.0 as follows. As mentioned in Section II, 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.

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 Memory Hierarchy Parameters	
Data L1	32KB
Instruction L1	32KB
L2 (Unified)	2MB
Base Processor Parameters	
Fetch/finish rate	8 per cycle
Retirement rate	1 dispatch grp (=max of 5)/cycle
Functional units	2 Int, 2 FP, 2 Ld-St, 1 Branch, 1 LCR
Integer FU latencies	1 add, 7 mul, 35 div (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

TABLE I
Base processor simulated.

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 [27].

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. [3]

IV. 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 [25], [26], [27].

A. 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 [29]. The 65nm processor parameters were derived from scaling down parameters from the 180nm POWER4 processor [26]. 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, resulting in very few L2 cache failures. Table I summarizes the base processor modeled.

B. Simulation Environment

For timing simulations, we use Turandot, a trace-driven research simulator developed at IBM's T.J.Watson Research Center [14]. Turandot was calibrated against a pre-RTL, detailed,

[3] 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% in each value.

latch-accurate processor model. For the power model, we use PowerTimer, a tool-set built around Turandot [6]. The power values from PowerTimer are fed into HotSpot to evaluate the temperatures of various components on chip [23]. These tools are interfaced with RAMP 2.0 as described earlier. As mentioned, RAMP 2.0 divides the processor into microarchitectural structures and operates at this structure granularity. In our work, we divide the processor into the following seven structures: instruction fetch unit (IFU), branch prediction unit (BXU), instruction decode unit (IDU), instruction scheduling unit (ISU), fixed point unit (FXU), floating point unit (FPU) and the load store unit (LSU) (similar to [27]).

C. 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 [14] 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 [10].

D. 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 [27].

For SD, the seven structures of the processor (Section IV-B) 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 [8], 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).[4] 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.

E. Metrics

We study the relationship between MTTF and nTTF for n values of 0.1, 0.5, 1 and 5. The specific failure probabilities 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 [9]. Another document mentions that 0.1% and 1% failure times are more relevant to processors than MTTF but fails to provide any additional information [13]. In addition to these documented values for n, we study n=0.5 to fill the space between 0.1% and 1%. We use n=5 as an upper bound for the target failure rate of a population of processors – a failure probability of 0.05 is a high failure probability for the microprocessor industry.

[4]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 [27]. We chose to report only the former since the method of counting this overhead is orthogonal to the point of this paper.

V. RESULTS

A. Can We Predict nTTF from MTTF?

We first explore if MTTF can be used to predict nTTF. It can be analytically shown that for both the exponential and lognormal distributions, for a given n, the nTTF is proportional to the MTTF (Appendix A). Unfortunately, the systems of interest to us have lifetimes that are complex functions of lognormal distributions. The base system is series-failure (it fails when any component fails) and so its lifetime is a minimum of several lognormals (one distribution for each failure mechanism applied to each component). GPD and SD employ redundancy and so their lifetimes are complex functions involving minimums, maximums, and sums of lognormals of the underlying components and failure mechanisms [27]. To our knowledge, there are no known analytical solutions to determine the distributions for such complex functions [12]. We therefore report empirical results.

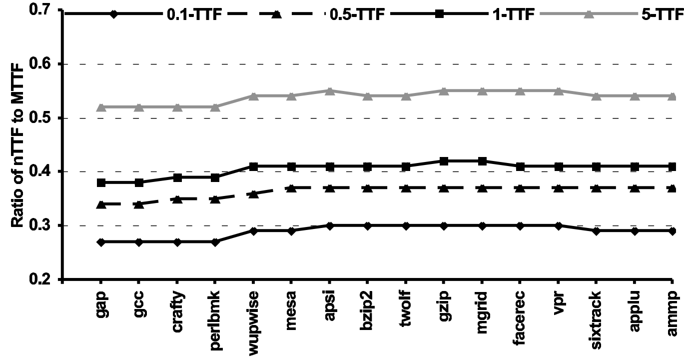
We first explore if there is a proportionality relationship between the MTTF and nTTF for our systems. Such a relationship would enable the use of MTTF as a proxy for nTTF for reliability analysis. Figure 2 shows the ratio of nTTF to MTTF for all applications for n=0.1%, 0.5%, 1%, and 5%, in the (a) Base, (b) GPD-1.11X, (c) GPD-2X, (d) SD-1.75X, and (e) SD-2.25X systems. The applications are ordered in increasing order of MTTFs. The chosen GPD and SD systems represent a mix of different performance and cost overheads to enhance system reliability, and the inferences drawn from these systems are similar to those from other systems.

The figure shows that the ratio of nTTF to MTTF, for a given n, is, in general, both system and application dependent. Tables II(a) and (b) respectively quantify these dependences further. For each n, each table assumes a constant nTTF/MTTF ratio, say k_n, and predicts nTTF as $k_n \times MTTF_{measured}$. Table II(a) assumes a system-oblivious (but application-dependent) k_n – we use the measured nTTF/MTTF for the base system for the given application. Table II(b) assumes an application-oblivious (but system-dependent) k_n – we use the measured nTTF/MTTF averaged across all applications for the given system. For each system, the tables report the error in the estimated nTTF (absolute error averaged across all applications) as:

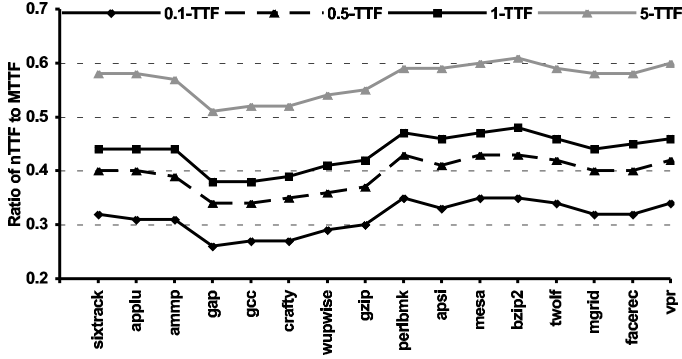
$$\frac{Error}{100} = \frac{1}{16}\sum_{n=1}^{16}\left(\left|\frac{k_n \times MTTF_{measured} - nTTF_{measured}}{nTTF_{measured}}\right|\right)$$

Table II(a) shows significant errors, clearly showing that the ratio of nTTF to MTTF is system dependent. Thus, information from the base system is insufficient to predict the nTTF from the MTTF of a reliability-enhanced system. The table shows that the errors in making such estimations may be as high as 33%.

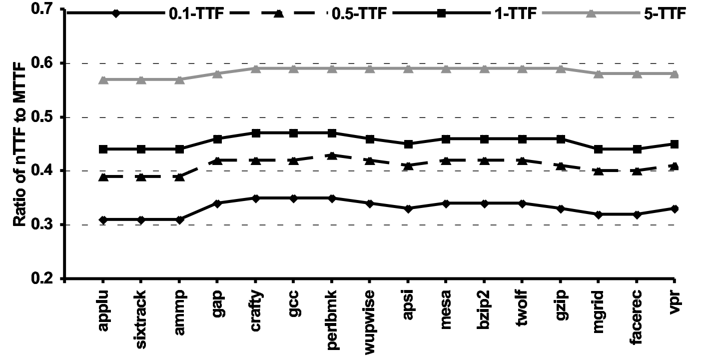
Table II(b) also shows errors, but these are much smaller than Table II(a). That is, for a given system, the nTTF/MTTF ratio shows smaller variation across our applications. However, given the system dependence in the previous table, it is hard to determine this ratio a priori and use it to predict nTTF from MTTF.



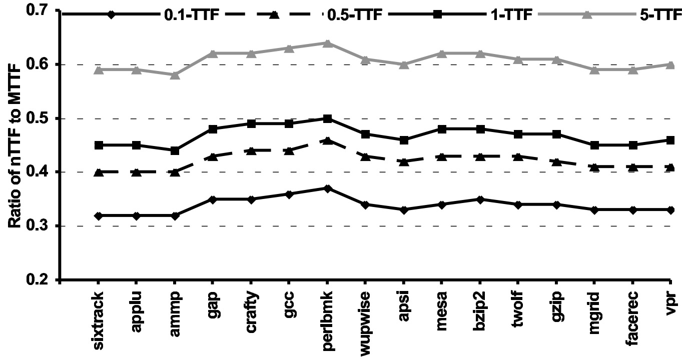
(a) Base



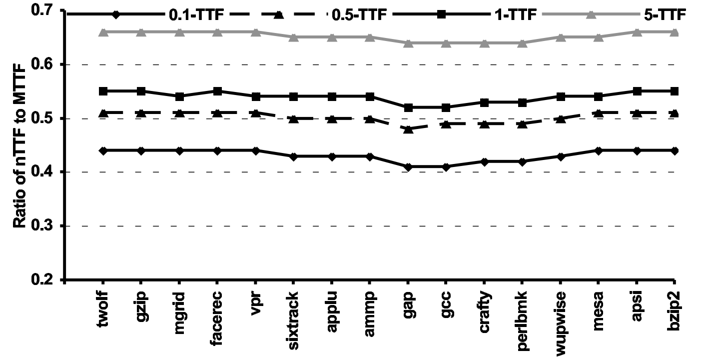
(b) GPD-1.11X



(c) GPD-2X



(d) SD-1.75X



(e) SD-2.25X

Fig. 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. For each system, the applications are ordered in increasing order of MTTFs. 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.**

B. Can We Predict nTTF Improvement from MTTF Improvement?

The previous section showed that, in general, the nTTF cannot be directly predicted from the MTTF. However, designers and users are often concerned with relative comparisons among systems; therefore, we next explore 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.

The benefit in nTTF (or MTTF) for a system is the ratio of the nTTF (or MTTF) of the GPD or SD system over that of the base system when running the same application. Figure 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. For each system, the applications are ordered in increasing order of MTTF benefit.

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

For example, consider the GPD-1.11X system shown in Figure 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

System	Error in estimation			
	0.1-TTF	0.5-TTF	1-TTF	5-TTF
Base	N/A	N/A	N/A	N/A
GPD-1.05X	6.81%	6.25%	5.84%	4.76%
GPD-1.11X	7.85%	7.20%	6.93%	5.52%
GPD-1.25X	10.25%	9.67%	8.99%	6.83%
GPD-1.43X	11.68%	10.50%	10.04%	7.46%
GPD-1.67X	12.72%	11.56%	10.88%	7.98%
GPD-2X	12.58%	11.44%	10.77%	7.79%
SD-1.25X	0.45%	0.17%	0.15%	0.00%
SD-1.5X	2.90%	3.01%	3.18%	2.62%
SD-1.75X	14.69%	13.99%	13.42%	11.38%
SD-2X	25.79%	23.73%	22.09%	16.97%
SD-2.25X	32.96%	27.60%	24.86%	17.42%

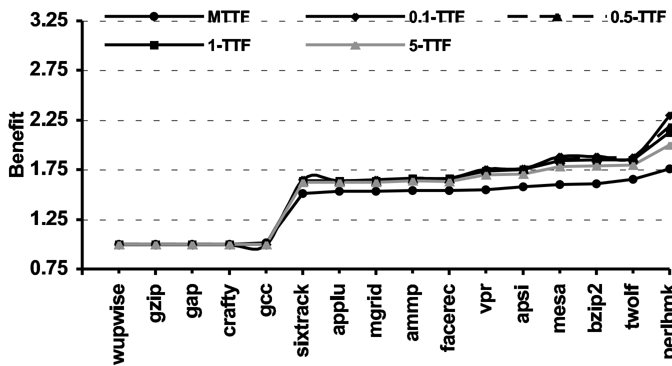
(a) Average error when system oblivious

System	Error in estimation				MTTF benefit
	0.1-TTF	0.5-TTF	1-TTF	5-TTF	
Base	3.41%	2.65%	2.51%	1.70%	0.00
GPD-1.05X	7.63%	7.34%	6.59%	5.03%	1.32
GPD-1.11X	7.84%	6.83%	6.03%	4.65%	1.40
GPD-1.25X	6.44%	5.01%	4.33%	2.96%	1.53
GPD-1.43X	5.41%	4.19%	3.71%	2.33%	1.58
GPD-1.67X	3.79%	2.76%	2.34%	1.29%	1.63
GPD-1X	3.61%	2.61%	2.22%	1.21%	1.63
SD-1.25X	3.77%	2.63%	2.44%	1.70%	1.00
SD-1.5X	2.97%	2.42%	2.17%	1.45%	1.03
SD-1.75X	3.35%	3.25%	3.12%	2.32%	1.66
SD-2X	3.37%	2.10%	1.92%	0.94%	1.88
SD-2.25X	2.05%	1.63%	1.34%	1.09%	2.04

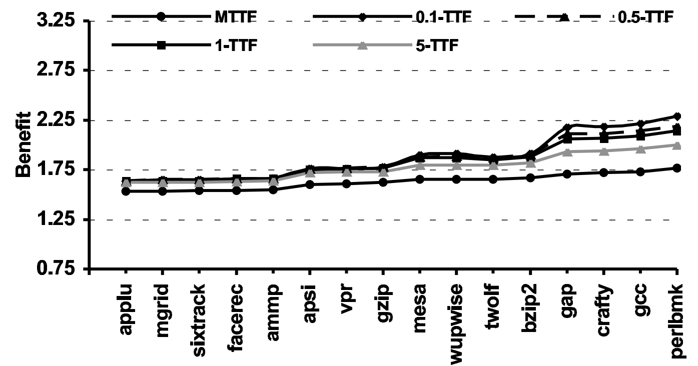
(b) Average error when application oblivious

TABLE II

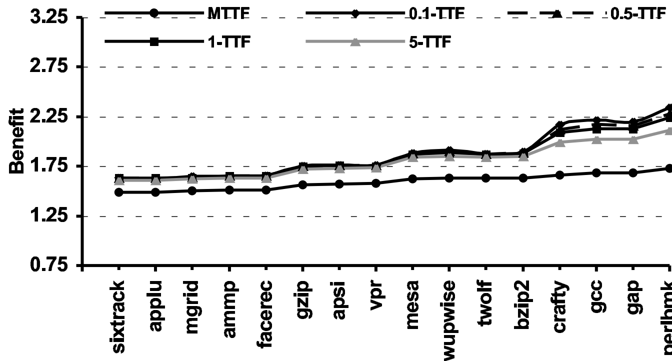
Average absolute error in estimating nTTF from the ratio of nTTF to MTTF in a (a) system-oblivious fashion (assuming the nTTF to MTTF ratio for the given system is the same as for the base system for the given application) and (b) application-oblivious fashion (assuming the nTTF to MTTF ratio for each application in the given system is the same and set to be the average ratio). The system-oblivious estimations show large errors in several cases, indicating a strong system dependence of the nTTF to MTTF ratio.



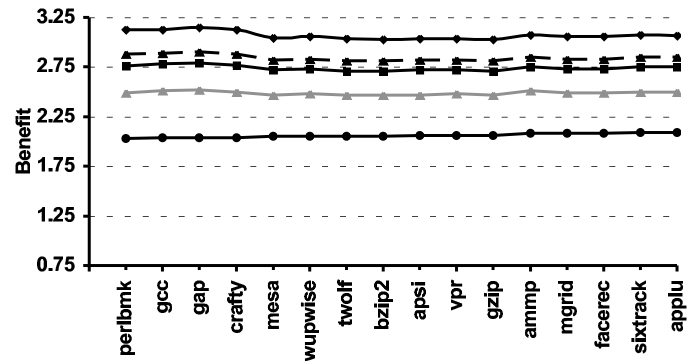
(a) GPD-1.11X system



(b) GPD-2X system



(c) SD-1.75X system



(d) SD-2.25X system

Fig. 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. In general, the benefit in MTTF cannot be used to predict the benefit in nTTF.**

benefits for different n. Thus, even for a given performance overhead, the absolute and relative benefits in MTTF and nTTF are application specific.

The absolute and relative benefits in nTTF and MTTF are also varied across different systems. For example, in the SD-1.75X system, *apsi* shows a 1.5X benefit in MTTF and a 1.7X benefit in nTTF for various n. However, the same application in the SD-2.25X system shows a 2X benefit in MTTF and an nTTF benefit larger than 2.5X for all n shown.

C. Does nTTF vs. MTTF Affect Design Decisions?

From a pragmatic viewpoint, we mainly care about whether our tools and metrics lead to the right design, even if they do not make the correct quantitative predictions. This section explores the impact of using MTTF instead of nTTF on the final design choice. We pose the following question. Suppose a designer had to improve the reliability of the base system by a certain factor, say K, by using either SD or GPD. How would the optimal design choice based on improving the MTTF by factor K compare with that based on improving nTTF by K? By optimal,

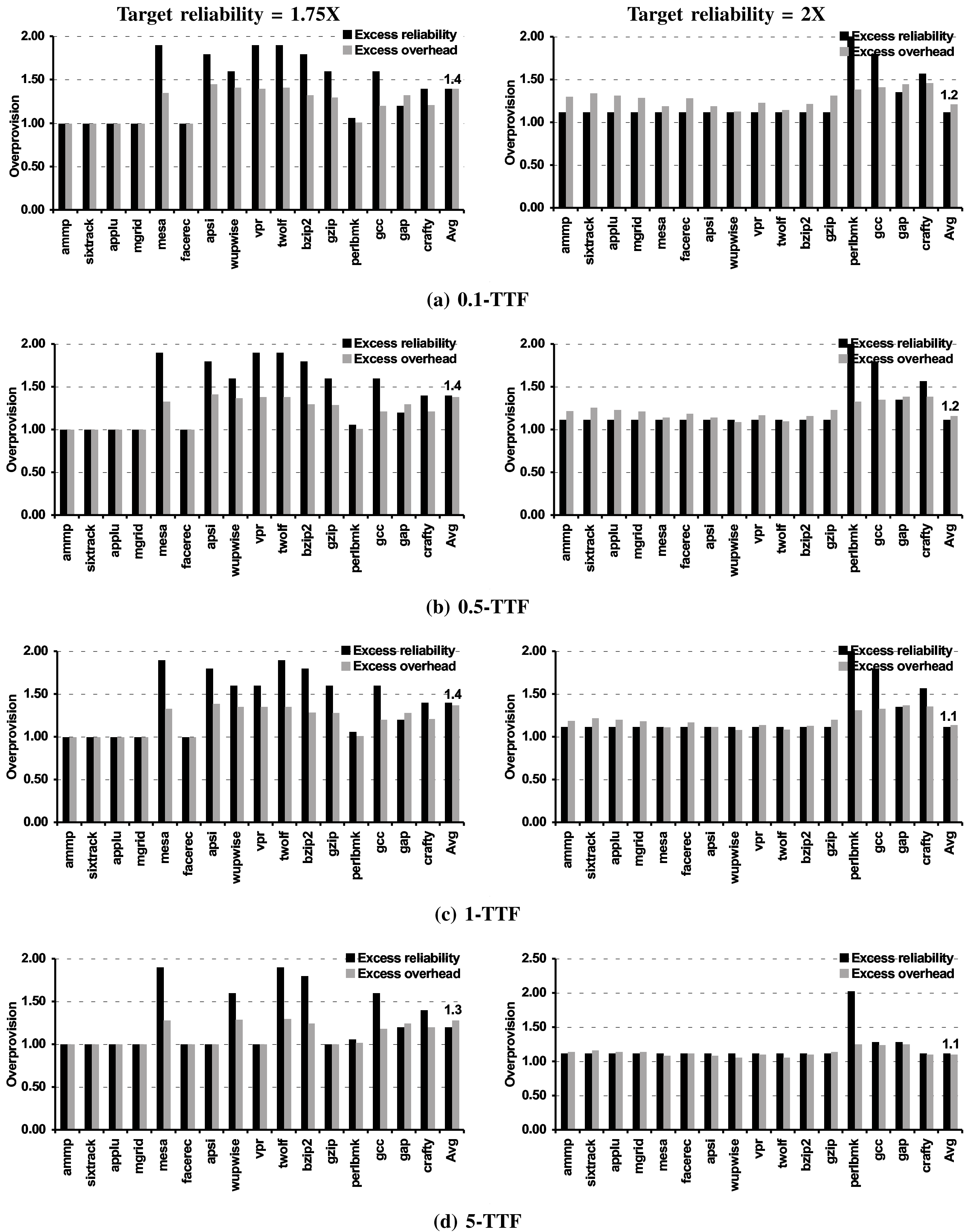


Fig. 4. **Sub-optimality in design decisions for reliability benefit targets of 1.75X and 2X when using MTTF vs. nTTF. The graphs show the over-provisioned reliability and excess overhead incurred when MTTF is used as a proxy for nTTF. The number on top of the average bars give the excess overhead for the given target. This incorrect projection of a target in nTTF as a target in MTTF often leads to poor design choices, incurring unnecessary overhead.**

we mean the design choice that achieves the target reliability improvement at the lowest cost or performance overhead.

Since the benefits in MTTF are mostly lower than the benefits in nTTF, designing with MTTF as a proxy for nTTF can lead to design choices that achieve an nTTF benefit that is well above the targeted reliability benefit, potentially resulting in an unnecessarily higher overhead system. Conversely, for cases where the benefits in MTTF are higher than the benefits in nTTF, an MTTF-optimal design may not achieve the desired reliability target in nTTF.

Figure 4 shows the impact of the differences in design choices for two target reliability benefits (over base) of 1.75X (left column graphs) and 2X (right column graphs). For a given target reliability, the figure shows two bars for each application. These bars respectively show the excess reliability and excess overhead incurred by an MTTF-driven design relative to a nTTF-driven design that achieves the target reliability benefit. A unit value on the graph denotes that the nTTF- and MTTF-driven designs both chose the same system (and so there is no excess reliability or overhead for the MTTF-driven design). Values higher than unity on both bars denote that the MTTF-driven design chose a system with an unnecessarily higher overhead (and an unnecessarily higher nTTF) than the nTTF-driven design. The (last) average bars in each set of graphs show the excess reliability and excess overhead when targeting the given reliability benefit on average across all applications.

These graphs show that when the target reliability improvement is projected as an MTTF improvement, the system chosen often has a much higher overhead than when the corresponding improvement is targeted for nTTF. This overhead provides higher reliability in nTTF, but this reliability is overkill. For example, when the target reliability is an improvement of 1.75X for 1-TTF but is projected incorrectly as a 1.75X improvement in MTTF, on an average, a system with excess overhead of 40% is chosen. The more expensive MTTF-driven system achieves an nTTF benefit of 1.4X more than the nTTF-driven system, but this is excess reliability that is not required, making the system over-designed. Individual applications and other systems show much larger over-provisioning, owing to varying benefits in various metrics across different systems.

Thus, the design decisions made for reliability-aware system design is impacted significantly by the choice of the metric. Incorrect design metrics may result in systems with an unnecessarily high overhead to be chosen to achieve the desired target when a system with a significantly lower overhead would have sufficed.

D. Analysis and Implications for Design: the Relative Vulnerability Factor

Our results so far clearly show that the nTTF metric has different implications than MTTF for reliability-aware design. This section seeks to understand the reasons for these differences, and describes implications of this analysis for the design methodology for reliability-aware systems.

In order to understand the nature of the benefits curves shown in Figure 3, we probe further into the lifetime distributions of the various components within a processor.

The main difference between nTTF and MTTF is that nTTF is focused on the first few years while MTTF is concerned with a longer time. Thus, clearly the failure processes see a change over these time scales. A key to understanding this difference is to get data on the relative probability of failure of a given component when we look at failure data over a long time scale versus when we look at failure data only over the time scale of nTTF years.

Table III presents this data for the SpecInt application *vpr* running on the Base, SD-1.75X, and SD-2X systems. A star (*) next to a component indicates that the component has a spare in the given system. We chose vpr for this analysis as other applications on other systems also show similar behavior, and the insights gained from the analysis of this benchmark are readily extensible to other scenarios. For a given system, the row labeled MTTF gives the percentage of the 10^7 Monte Carlo trials where the given component was responsible for system failure (i.e., the component failed first). The row labeled 0.1-TTF provides the same statistic, but for the trials that failed within the first 0.1-TTF years (i.e., for the 10^4 trials that resulted in the earliest failures, it gives the percentage of these trials where the given component was responsible for failure).

We refer to the data shown as the relative vulnerability factor or RVF of the component – this factor is critical in determining the reliability properties of the system. Clearly, the component with the highest RVF is the most likely to fail and must be made redundant for effective design. If there are multiple components with similar RVFs, all would require redundancy to make a difference. Now if the RVF values of the different components change for different metrics, then design decisions based on one metric will not necessarily be good for another. This is what we see in the previous sections. In cases where the MTTF improvement is not predictive of nTTF improvement and where the overhead-optimal designs are different, we find that the RVFs of the components are markedly different when using MTTF or nTTF as the metric.

For example, in Table III, in the base system, the LSU has the highest RVF (99%) for the 0.1-TTF metric (it fails first in over 99% of the trials that fail in the first 0.1TTF years). For the MTTF metric, the LSU still has the highest RVF value, but it is a lower 88% (it fails first in 88% of all 10^7 trials). Thus, when a spare LSU is added (as in the SD-1.75X system) 0.1-TTF improves by a higher factor of 1.8X compared to the 1.5X improvement in MTTF. Similarly, in other cases where the benefits in the various metrics differ, we see that the RVFs of the different components are different under the various metrics.

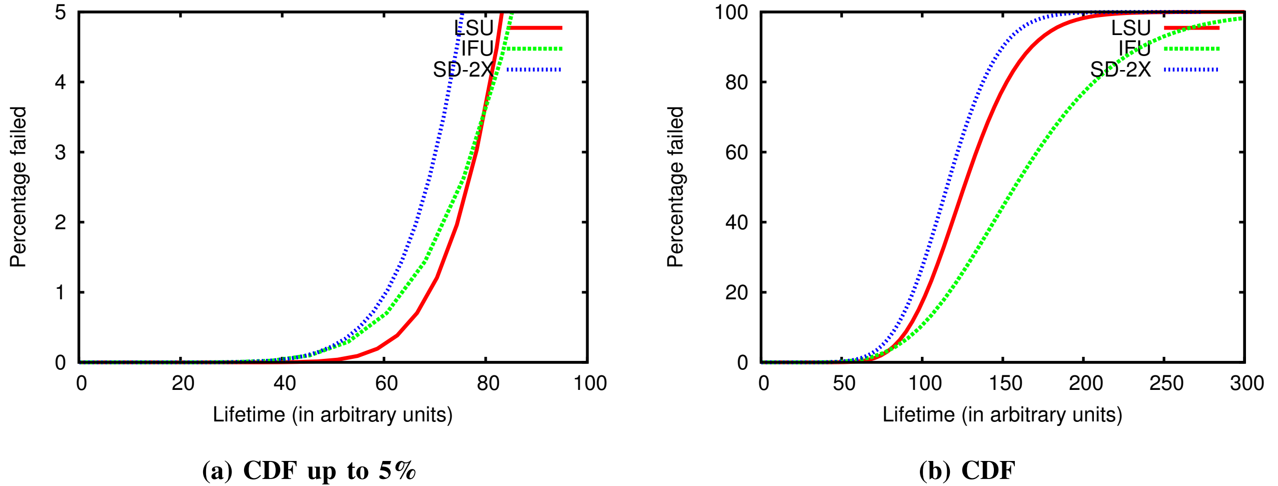
These differences in the vulnerability arise from the underlying lifetime distributions of the components. The vulnerability for nTTF (for small n) depends on the nature of the lifetime distribution curve in the initial years, while those of the MTTF depend on the long-term lifetime curve. Figure 5 shows the CDFs of the LSU and the IFU lifetimes when the vpr application is run on the SD-2X system (the CDF of the full system lifetime is also shown). Figure 5(a) shows the CDFs up to 5%, while Figure 5(b) shows the full curves.

We see from Figure 5(a) that for small lifetimes, t, the IFU has a higher failure probability than the LSU (the CDF curve for the IFU is higher than that of the LSU) while for larger t, Figure 5 shows that the LSU has a higher failure probability. Hence, the IFU has a higher RVF for 0.1-TTF (which considers

System	Metric	IFU	BXU	IDU	ISU	FXU	FPU	LSU
Base	MTTF	1.97%	0.00%	0.01%	10.13%	0.13%	0.08%	87.68%
Base	0.1-TTF	0.04%	0.00%	0.00%	0.82%	0.00%	0.00%	99.14%
SD-1.75X	MTTF	14.17%	0.03%	0.14%	52.74%	1.52%	1.33%	30.06*%
SD-1.75X	0.1-TTF	5.94%	0.00%	0.00%	93.87%	0.07%	0.08%	0.04*%
SD-2X	MTTF	26.80%	0.09%	0.00*%	1.85*%	3.53%	3.02%	64.71*%
SD-2X	0.1-TTF	90.05%	0.00%	0.00%	0.00*%	1.45%	1.36%	7.14*%

TABLE III

Relative Vulnerability Factor (RVF) of various components for MTTF and 0.1-TTF with the application *vpr* running on the Base, SD-1.75X, and SD-2X systems. A star (*) next to a component indicates that the component has a spare in the given system. The RVF of the components is different under the two metrics, resulting in varying benefits from having spares for these components.



(a) CDF up to 5% **(b) CDF**

Fig. 5. **Cumulative distribution functions of the LSU and the IFU lifetimes when the application vpr runs in the SD-2X system. The CDF of the system lifetime is also shown. (a) shows the CDF up to 5%, while (b) shows the entire CDF. The curves show that initially the IFU is more likely to fail while in later years the LSU is more vulnerable. Thus, when designing for nTTF for small n, redundancy should be added to the IFU while MTTF-driven design would add further redundancy to the LSU.**

failures for a small t) while the LSU has a higher RVF for the MTTF. Thus, using these RVF values, designs based on MTTF will add a spare LSU while designs based on 0.1-TTF will add a spare IFU. This clearly shows that the RVF directly impacts the design choices made when designing for different metrics, resulting in different designs under different metrics.

Thus, the knowledge of the RVF for different components in the system under different metrics can help guide the right design choices that incur the lowest overhead while achieving the highest reliability benefit.

VI. CONCLUSIONS

Aggressive CMOS scaling is expected to accelerate wear-out based failures. 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.

Comparing these reliability-aware microarchitectures requires a thorough understanding of the metrics used for such a comparison. MTTF (or FIT, computed as $\frac{1}{MTTF}$) is a widely reported reliability metric in academic and marketing literature. 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. An alternate metric is nTTF, which represents the time to failure of n% of the processor population. However, it is not clear if the choice of metrics makes a quantitative difference to modern processor designs.

This paper quantitatively analyzes the implication of designing reliability-aware microarchitectures for MTTF versus designing for nTTF. Distribution dependent metrics such as nTTF are inherently more complex to compute, but this work presents a straightforward way to compute such a metric using a state-of-the-art architecture level lifetime reliability tool RAMP 2.0.

Our analysis shows that the choice of metrics significantly impacts the design decisions for reliability-aware designs. We showed examples where MTTF-driven designs incurred an excess overhead of 40% or more for unnecessary reliability. The difference in the optimal design choice under the two metrics results from a change in the relative vulnerability factor (RVF) of the different processor components over time, due to the complex lifetime distribution curves of our systems. Since nTTF is concerned with the first few years while MTTF is a mean over the entire distribution curve, the change in RVFs results in different design choices.

Admittedly, current architecture level tools for lifetime reliability are still preliminary and our work also makes significant assumptions. Nevertheless, this paper takes an important step in understanding the right metrics that should be targeted by such tools and quantifies the impact of using different 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 important step in understanding the differences between two such metrics – MTTF and nTTF.

ACKNOWLEDGMENTS

We would like to thank Jayanth Srinivasan for laying the foundations on which this work was built. We would also like to thank Marc Snir for his suggestion to use the Monte Carlo simulations for computing alternate metrics, and Shubu Mukherjee for discussions that helped reinforce, for us, the importance of alternatives to MTTF.

REFERENCES

[1] "Methods for Calculating Failure Rates in Units of FITs," JEDEC Publication, Tech. Rep. JESD85, 2001.
[2] J. Blome *et al.*, "Self-calibrating Online Wearout Detection," in *Proceedings of International Symposium on Microarchitecture*, 2007.
[3] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, vol. 25, no. 6, 2005.
[4] F. Bower *et al.*, "A Mechanism for Online Diagnosis of Hard Faults in Microprocessors," in *Proceedings of International Symposium on Microarchitecture*, 2005.
[5] F. A. Bower *et al.*, "Tolerating Hard Faults in Microprocessor Array Structures," in *Proceedings of International Conference on Dependable Systems and Networks*, 2004.
[6] D. Brooks *et al.*, "Power-aware microarchitecture: Design and modeling challenges for next-gen microprocessors," *IEEE Micro*, vol. 20, no. 6, 2000.
[7] Gateway, "Press Release," 2006, download.intel.com/personal/news/GatewayPressRelease.pdf.
[8] J. L. Hennessy and D. A. Patterson, *Computer Architecture, A Quantitative Approach*. Morgan Kaufmann, 2003.
[9] Intel, "LXT971A Fast Ethernet Transceiver," http://www.intel.com/design/network/products/lan/phys/lxt971a-972a.htm.
[10] V. S. Iyengar, L. H. Trevillyan, and P. Bose, "Representative traces for processor models with infinite cache," in *Proceedings of International Symposium on High Performance Computer Architecture*, 1996.
[11] 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*, vol. 25, no. 6, Nov./Dec. 2005.
[12] M. Milevsky and S. Posner, "Asian Options, The Sum of Lognormals and the Reciprocal Gamma Distribution," *The Journal of Financial and Quantitative Analysis*, vol. 33, no. 3, pp. 409–422, September 1998.
[13] "Reliability in CMOS IC Design: Physical Failure Mechanisms and their Modeling," MOSIS Technical Notes – http://www.mosis.org/support/technical-notes.html.
[14] M. Moudgill, P. Bose, and J. Moreno, "Validation of Turandot, a Fast Processor Model for MicroArch. Evaluation," in *Proceedings of International Conference on Performance, Computing and Communication*, 1999.
[15] NIST, "Assessing Product Reliability, NIST/SEMATECH e-Handbook of Statistical Methods," http://www.itl.nist.gov/div898/handbook/.
[16] P. Ramachandran, "Limitations of using the MTTF metric for architecture-level lifetime reliability analysis," MS Thesis, Department of Computer Science, University of Illinois at Urbana Champaign, 2007.
[17] Reliasoft, "The Limitations of Using the MTTF as a Reliability Specification," 2001, http://www.reliasoft.com/newsletter/4q2001/exponential.htm.
[18] M. Russel, "Quality Busters: Reducing application complexity, IBM," 2005, http://www.ibm.com/developerworks/library/wa-qualbust12/.
[19] J. Shin, V. Zyuban, Z. Hu, J. A. Rivers, and P. Bose, "A Framework for Architecture-Level Lifetime Reliability Modeling," in *Proceedings of International Conference on Dependable Systems and Networks*, 2007.
[20] S. Shyam *et al.*, "Ultra Low-Cost Defect Protection for Microprocessor Pipelines," in *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, 2006.
[21] Sidharth and S. Sundaram, "A methodology to assess microprocessor fan reliability," in *The 9th Intersociery Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2004.
[22] D. Siewiorek and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*. Digital Press, 1992.
[23] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarch." in *Proceedings of International Symposium on Computer Architecture*, 2003.
[24] 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.
[25] J. Srinivasan *et al.*, "The Case for Microarchitectural Awareness of Lifetime Reliability," in *Proceedings of International Symposium on Computer Architecture*, 2004.
[26] ——, "The Impact of Scaling on Processor Lifetime Reliability," in *Proceedings of International Conference on Dependable Systems and Networks*, 2004.
[27] ——, "Exploiting Structural Duplication for Lifetime Reliability Enhancement," in *Proceedings of International Symposium on Computer Architecture*, 2005.
[28] ——, "Lifetime Reliability: Toward an Architectural Solution," *IEEE Micro*, vol. 25, no. 3, 2005.
[29] J. Tendler *et al.*, "POWER4 system microarchitecture," *IBM Journal of Research and Development*, vol. 46, no. 1, 2002.
[30] D. Trindade and S. Nathan, "Analysis of field data for repairable systems," in *Proceedings of Reliability and Maintainability Symposium*, 2006.
[31] K. Trivedi, *Probability and Statistics with Reliability, Queueing, and Comp. Science Applications*. Prentice Hall, 1982.
[32] A. Wood, "Reliability-metric varieties and their relationships," in *Proceedings of Reliability and Maintainability Symposium*, 2001.
[33] D. Yen, "Chip Multithreading Processors Enable Reliable High Throughput Computing," in *Proceedings of International Reliability Physics Symposium*, 2005, keynote Address.

APPENDIX

We show here that the ratio of nTTF to MTTF is a constant for the exponential and lognormal distributions. Note that this result does not apply to the lifetime distributions of the full systems we study since those are complex functions (minimum/maximum/sum) of lognormals.

The following derivations use standard definitions and results from probability theory and are presented here for completeness [15]. Let G(0,1)(n) denote the nTTF function (or the percent point function) of a standard distribution with location parameter 0 and scale parameter 1. When this distribution is scaled by a parameter b, the percent point function scales according to

$$G(0,b)(n) = b \times G(0,1)(n) \quad (2)$$

For the exponential distribution, the scale parameter is the Mean. Thus, for an exponential distribution

$$G(0,b)(n) = Mean \times G(0,1)(n) \quad (3)$$

$$\Rightarrow \frac{G(0,b)(n)}{Mean} = G(0,1)(n) \quad (4)$$

Thus, for the exponential distribution, the ratio of nTTF to MTTF is a constant, independent of the mean.

For a lognormal distribution, the scale parameter is the median (T_{50}) where the mean and median are related by

$$T_{50} = Mean \times e^{\frac{\sigma^2}{2}} \quad (5)$$

where σ is the shape for the lognormal distribution.

Substituting for the scale parameter from 5 into 2, we get

$$G(0,b)(n) = \left(Mean \times e^{\frac{\sigma^2}{2}}\right) \times G(0,1)(n) \quad (6)$$

$$\Rightarrow \frac{G(0,b)(n)}{Mean} = e^{\frac{\sigma^2}{2}} \times G(0,1)(n) \quad (7)$$

Thus, for a lognormal distribution, the ratio of the nTTF to MTTF is also a constant (for a given shape parameter σ).