In the Proceedings of *The 32nd International Symposium on Computer Architecture (ISCA'05)*, June 2005[1].

Exploiting Structural Duplication for Lifetime Reliability Enhancement *

Jayanth Srinivasan, Sarita V. Adve
University of Illinois at Urbana-Champaign
Department of Computer Science
{srinivsn,sadve}@cs.uiuc.edu,

Pradip Bose, Jude A. Rivers
IBM T.J. Watson Research Center
Yorktown Heights,NY
{pbose,jarivers}@us.ibm.com

Abstract

Increased power densities (and resultant temperatures) and other effects of device scaling are predicted to cause significant lifetime reliability problems in the near future. In this paper, we study two techniques that leverage microarchitectural structural redundancy for lifetime reliability enhancement. First, in **structural duplication** *(SD), redundant microarchitectural structures are added to the processor and designated as spares. Spare structures can be turned on when the original structure fails, increasing the processor's lifetime. Second,* **graceful performance degradation** *(GPD) is a technique that exploits existing microarchitectural redundancy for reliability. Redundant structures that fail are shut down while still maintaining functionality, thereby increasing the processor's lifetime, but at a lower performance.*

Our analysis shows that exploiting structural redundancy can provide significant reliability benefits, and we present guidelines for efficient usage of these techniques by identifying situations where each is more beneficial. We show that GPD is the superior technique when only limited performance or cost resources can be sacrificed for reliability. Specifically, on average for our systems and applications, GPD increased processor reliability to 1.42 *times the base value for less than a* 5% *loss in performance. On the other hand, for systems where reliability is more important than performance or cost, SD is more beneficial. SD increases reliability to* 2.53 *times the base value for* 2.25 *times the base cost, for our applications. Finally, a combination of the two techniques (SD+GPD) provides the highest reliability benefit.* [1]

1 Introduction

Lifetime reliability due to wear-out related hard errors in processor components is emerging as a critical challenge in modern microprocessors. The steady processor performance increases seen over the last twenty years have been driven by aggressive scaling of CMOS devices. At the same time, scaling leads to higher temperatures and reduced device feature sizes which results in lower processor lifetime reliability [23]. Device, manufacturing, and fabrication researchers have been aware of the lifetime reliability problem for many years and there exists a large body of research at the device level. On the other hand, there is a dearth of architectural lifetime reliability research as microarchitects have traditionally not viewed the subject as a problem.

As a first step towards addressing this issue, in [22], we proposed RAMP, a microarchitecture-level model that dynamically tracks processor lifetime reliability, accounting for the behavior of the executing application. In [23], we integrated device scaling models in RAMP and quantified the impact of technology scaling on reliability, showing that scaling has a significant and increasing effect on processor hard failure rates. For a contemporary superscalar processor running Spec2000 applications, our results in [23] show an average increase of 316% in processor failure rates when scaling from 180nm to 65nm. In such a reliability-constrained environment, some performance and/or die area (and resultant cost) will have to be sacrificed for reliability. In this paper, we examine efficient usage of these performance and cost budgets through structural redundancy for lifetime enhancement.

*This work is supported in part by an equipment donation from AMD and the National Science Foundation under Grant No. EIA-0224453. Jayanth Srinivasan is supported by an IBM Ph.D. Fellowship, and a large part of the work was performed while he was a co-op at IBM T. J. Watson Research Center.

[1]This version of the paper differs slightly from the original ISCA'05 version due to a correction in the lognormal equation (discussed in Section 3.2). Although there is a small change in some of the quantitative results, there is no change in the qualitative results or conclusions of the paper.

1.1 Exploiting Structural Redundancy for Lifetime Reliability

Redundancy is a commonly used technique for reliability enhancement. However, most previous work for lifetime reliability focused on redundancy at the processor granularity. Due to the large area overheads involved in duplicating entire processors, such redundancy does not provide a cost-effective reliability solution. Structural redundancy addresses some of these shortcomings of processor redundancy by incurring less area overhead and allowing runtime processor reconfiguration for reliability.

We examine two methods by which structural redundancy can be used for reliability enhancement. In the first case, referred to as **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 a processor would have normally failed, the spare structure extends the processor's lifetime. With SD, the processor fails only in the case where a structure without a spare fails, or all available spares have been used. It should be noted that the main function of the spare units is to increase reliability, and not performance. As a result, the spare structures are power gated and not used at the beginning of the processor's life (a power gated structure would suffer almost no hard errors since there would be no gate-oxide breakdown or interconnect wear-out).

Next, we examine **graceful performance degradation** (GPD) which allows the processor to exploit existing microarchitectural redundancy for reliability. Modern processors have replicated structures that are used for increasing performance for some high parallelism applications. However, the replicated structures are not required for functional correctness. If a replicated structure fails in the course of a processor's lifetime, the processor can shut down the structure and still maintain functionality, thereby increasing lifetime. Hence, rather than fail when the first structure on chip fails, a processor with GPD would fail only when all redundant structures of a type fail. We also examine architectures that use a combination of SD and GPD.

Both SD and GPD incur overheads while increasing reliability. In the case of SD, extra processor die area is required due to the introduction of spare structures. This area overhead translates into a cost overhead. However, SD results in no performance loss relative to the base processor. Conversely, GPD results in a processor's performance degrading during its lifetime when replicated structures fail. However, since no extra structures are added to the processor, this technique comes with no area overhead.

Given a reliability-constrained design situation, some performance and/or cost will have to be sacrificed for reliability. Our analysis shows that structural redundancy can use this performance or cost tradeoff for significant reliability benefit. In addition, we provide guidelines for intelligent reliability decisions by identifying the superior design technique for a given performance or cost trade-off. For our systems and applications, we show that GPD is a superior technique when only limited performance or area resources can be sacrificed for reliability. On average, GPD increases processor reliability to 1.42 times the base value for less than a 5% loss in performance. On the other hand, for systems where reliability is more important than performance or cost, SD is more beneficial. SD increases reliability to 2.53 times the base value for 2.25 times the base cost for our systems and applications. Finally, a combination of SD and GPD increases reliability to as much as 4.16 times the base value.

1.2 Enhancements to the Reliability Model

Our reliability modeling methodology is based on RAMP [22], which represents the current state-of-the-art. However, to use RAMP to evaluate SD and GPD, we had to enhance some parts of the model. Currently, RAMP assumes all processors are series failure systems [22]; i.e., the first failure anywhere on chip will cause the entire processor to fail. However, processors that use redundancy for SD or GPD are series-parallel failure systems. Also, RAMP assumes all failure mechanisms have an exponential distribution, which implies that they have a constant failure rate throughout the processor lifetime [22]. This is inaccurate – a typical wear-out failure mechanism will have a low failure rate at the beginning of the component's lifetime and the value will grow as the component ages. We address this limitation in RAMP by modeling failure mechanisms with lognormal distributions. Lognormal distributions better model failure mechanisms than exponential distributions [1], and allow us to model the dependence on time of the failure mechanisms. We then use Monte-Carlo simulation methods in RAMP to calculate total processor reliability for series-parallel systems with lognormal distributions.

Finally, we incorporate a model for a new failure mechanism, negative bias temperature instability (NBTI), into RAMP. Currently RAMP models four critical mechanisms – electromigration, stress migration, time dependent dielectric breakdown, and thermal cycling. NBTI has recently emerged as a critical failure mode, and is expected to grow in importance with scaling [26].

2 Related Work

Redundancy has been a commonly used technique for lifetime reliability enhancement in processor design, and there exists a large body of work on the subject [2, 21]. However, this work has primarily focused on redundancy at the processor granularity for systems. In particular, much has been done on systems that require manual "hot-swapping" of a new processor when a processor fails [21].

Structural redundancy addresses some of the shortcomings of processor redundancy by providing a more cost and performance effective solution.

There are some systems that duplicate at a structural granularity within a processor for soft error detection and tolerance. Prominent among such systems is the IBM S/390 System [21] and the Compaq NonStop Himalaya Systems [2]. However, in both systems, all replicated processor units are concurrently utilized, and the replication is not intended for hard error tolerance.

Redundancy is also used in microprocessor yield enhancement techniques [13, 19]. These are not run-time techniques and are instead used during processor testing. They are based on detecting and disabling faulty processor resources like cache lines [13]. Shivakumar et al. extend this concept and propose disabling defective redundant microarchitectural structures during testing to improve yield [19], resulting in gracefully degraded processors. They also suggest that this redundancy can be exploited to increase useful processor lifetime.

Finally, redundancy is also utilized in array structures for lifetime enhancement. Many current memory systems utilize built-in self test (BIST) and built-in self repair (BISR) to detect and disable faulty memory elements. Redundant spares are then swapped in [10]. Recently, Bower et al. proposed self-repairing array structures (SRAS), a technique to mask hard faults in array structures like the reorder buffer and branch history table [4]. These techniques are limited to array structures and replicate at the granularity of individual array entries.

3 Enhancements to RAMP

3.1 RAMP Overview

As mentioned in Section 1, our reliability modeling methodology is based on RAMP [22]. RAMP uses industrial strength analytic models for four failure mechanisms, electromigration, stress migration, time-dependent dielectric breakdown, and thermal cycling, and provides lifetime estimates based on the executing application. Much like previous power and temperature models [6, 20], RAMP divides the processor into discrete structures like the functional units and caches, and applies the analytic failure models to the structure as a whole.

The failure models in RAMP provide reliability estimates in terms of mean time to failure (MTTF). RAMP combines the MTTFs due to each failure mechanism across all the structures to provide a total processor MTTF for the given application. This is done using the industry-standard sum-of-failure-rates (SOFR) model. The SOFR model makes two assumptions [25]: (1) The processor is a series failure system – in other words, the first failure of any structure due to any failure mechanism would cause the entire processor to fail; and (2) each individual failure mechanism has a constant failure rate (equivalently, every failure mechanism has an exponential lifetime distribution). A constant failure rate implies that the probability of failure of a processor does not vary with its age. Both assumptions limit RAMP's applicability. First, many redundant structures on chip can fail without the entire processor failing. Hence, the ability to model series-parallel failure systems in addition to series failure systems is required. Second, wear-out failure mechanisms do not exhibit constant failure rates. Instead, wear-out mechanisms have low failure rates at the beginning of the processor's lifetime and the value will grow as the processor ages (the probability that a processor will fail will increase, the older the processor gets).

In order to use RAMP to evaluate structural duplication and graceful performance degradation, we address the above two limitations of the SOFR model. We use lognormal distributions (instead of exponential) for the failure mechanisms, and we use a Monte-Carlo simulation method to model series-parallel systems with lognormal distributions. In Section 3.2, we describe lognormal distributions, and we explain our Monte-Carlo simulation methodology for series-parallel systems in Section 3.3. Finally, we add a model for an emerging critical failure mechanism, NBTI, to the existing four failure mechanisms in RAMP. This is discussed in Section 3.4.

3.2 Lognormal Distributions

The lognormal distribution has been found to be a better model for failure degradation processes common to semiconductor failure mechanisms than the exponential distribution [1, 15, 14]. In most cases, this can be shown using the multiplicative degradation argument [1], briefly explained below. For a structure undergoing wear-out due to some failure mechanism, let $x_1, x_2, ...x_n$ be the amount of degradation seen at successive discrete time intervals. Let us assume that the amount of degradation seen in a time interval tends to depend on the total amount of degradation already present. This is known as multiplicative degradation [1]. In other words, the amount of degradation experienced in the n^{th} time interval, $(x_n - x_{n-1})$, will be some multiple of the total degradation already present at the end of the $(n-1)^{th}$ time interval, x_{n-1}. Hence,

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

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

$$x_n = [\prod_{i=1}^{n} (1 + \alpha_i)] x_0 \qquad (2)$$

where x_0 is the degradation at time 0, and is a constant, and

α_i are small random values. Taking the natural logarithm of both sides,

$$\ln x_n = \sum_{i=1}^{n} \ln(1+\alpha_i) + \ln x_0 \approx \sum_{i=1}^{n} \alpha_i + \ln x_0 \quad (3)$$

since $\ln(1+x) \approx x$ for small values of x. Since α_i are random, equidistant, independent values, the Central Limit Theorem [25] implies that $\ln x_n$ has a normal distribution. Hence, x_n has a lognormal distribution for any n (or any time t). Since failure occurs when the amount of degradation reaches a critical point, time of failure will be modeled successfully by a lognormal for this type of process. The multiplicative degradation model has been shown to be a good fit for chemical reactions, diffusion of ions, and crack growth and propagation. Most semiconductor failure models are caused by one of these three degradation processes [1]. Hence, the lognormal distribution is a good fit for wear-out mechanisms.

The probability density function for the lognormal distribution is given by [12]:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \quad (4)$$

μ and σ are the mean and standard deviation of the underlying norrmal distribution [12]. μ is related to the MTTF of the lognormal distribution, $MTTF$, as $MTTF = e^{\mu + \frac{\sigma^2}{2}}$ [12]. As suggested in [3], we use $\sigma = 0.5$ which has been found to model wear-out based failure mechanisms well.

3.3 Monte Carlo Simulation for Reliability

To obtain the lifetime distribution and MTTF for the processor as a whole, we need to combine the effects of the individual lognormal distributions across all the mechanisms and structures. Due to the complexity of the lognormal distribution, and the large cross-product of structures and mechanisms, calculating processor reliability analytically is exceedingly difficult. [2] To address this problem, we use a Monte Carlo simulation method to calculate total processor reliability. A Monte Carlo method is an algorithm that solves a problem by generating suitable random numbers and observing the fraction of the numbers that obey some property or properties. The method is useful for obtaining numerical solutions to problems that are too complicated to solve analytically [18].

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

3.3.1 Generating Lognormal Distributions

The Box-Muller transform can be used to generate a lognormal distribution from a uniform distribution [7]. As discussed previously, the mean of the underlying normal distribution, μ, is related to the MTTF of the lognormal distribution, $MTTF$, by

$$MTTF = e^{\mu + \frac{\sigma^2}{2}} \quad (5)$$

Hence,

$$\mu = ln(MTTF) - \frac{\sigma^2}{2} \quad (6)$$

Also, as described earlier, for a wear-out based failure mechanism, $\sigma = 0.5$.
If $rand1$ and $rand2$ are two independent uniformly distributed random numbers, a normally distributed random number, $rand_{normal}$, with mean 0 and standard deviation 1, is given by [7]

$$rand_{normal} = \sqrt{-2ln(rand1)} \times sin(2\pi rand2) \quad (7)$$

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

$$rand_{scaled-normal} = \mu + rand_{normal} \times \sigma \quad (8)$$

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

$$rand_{lognormal} = e^{rand_{scaled-normal}} \quad (9)$$

Substituting,

$$rand_{lognormal} = e^{ln(MTTF) - \frac{\sigma^2}{2} + \sigma(\sqrt{-2ln(rand1)}sin(2\pi rand2)} \quad (10)$$

Hence, with Equation 10, two random uniform variables, $rand1$ and $rand2$, can be used to generate a lognormal distribution with parameters $MTTF$ and σ.

3.3.2 Modeling Systems with the MIN-MAX Method

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

If A and B are in series, failure would occur at $MIN(t_A, t_B)$ because the first structure to fail will cause the processor to fail. On the other hand, if A and B are in parallel, failure would occur at $MAX(t_A, t_B)$ because both structures have to fail for the processor to fail. If a structure, C, with lifetime, t_C, is added in series to A and B in parallel, the new lifetime of the processor would be $MIN(MAX(t_A, t_B), t_C)$. This simple concept can be extended to any processor represented in a series or series-parallel fashion to obtain total MTTF.

In any single iteration of the Monte-Carlo experiment, we use Equation 10 to generate a random lifetime for each failure mechanism and structure on chip. A MIN-MAX analysis of these lifetimes based on the processor's configuration would give the lifetime of the entire processor for that iteration. The MTTF of the processor can now be calculated by repeating this process over many iterations and averaging the processor lifetimes obtained. As in any other Monte-Carlo experiment, the accuracy of the analysis increases with the number of iterations performed.

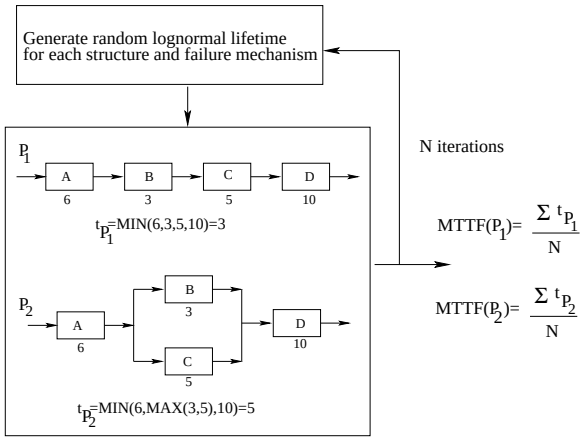


Figure 1. Monte Carlo simulation of MTTF of two processors, P_1 and P_2. The MIN-MAX method to determine processor lifetime is illustrated for sample lifetime values for both processors.

Figure 1 illustrates this method. Consider two processors, P_1 and P_2. Both processors have four structures, A, B, C, and D. P_1 is a series failure system while P_2 is a series-parallel failure system. For any single iteration of the Monte-Carlo algorithm, the lifetime of P_1 is $t_{P_1} = MIN(t_A, t_B, t_C, t_D)$, while the lifetime of P_2 is $t_{P_2} = MIN(t_A, MAX(t_B, t_C), t_D)$, where t_A, t_B, t_C, and t_D are the randomly generated lifetimes of each structure. If N iterations are performed, the MTTF of processor P_1 is $MTTF_{P_1} = \sum \frac{t_{P_1}}{N}$, and the MTTF of processor P_2 is $MTTF_{P_2} = \sum \frac{t_{P_2}}{N}$. In our experiments, we use a value of $N = 10^7$.

3.4 Negative Bias Temperature Instability (NBTI)

Currently, RAMP models four critical failure mechanisms – electromigration, stress migration, time dependent dielectric breakdown, and thermal cycling. We add a model for another emerging critical failure mechanism, NBTI, which is an electro-chemical reaction that takes place in PFETs when the gate is biased negative with respect to the source and drain. This typically occurs when the input to a gate is low while the output is high, resulting in an accumulation of positive charges in the gate oxide. This accumulation causes the threshold voltage of the transistor to increase. Higher threshold voltages result in gate overdrive (supply voltage - threshold voltage) decreasing, which slows down the performance of the gate. This eventually leads to processor failure due to timing constraints [26].

NBTI has a strong positive temperature and field dependence. As a result, the higher temperatures seen on chip due to scaling exacerbate this problem. Similarly, thinning of the gate oxide due to scaling also increases NBTI reliability concerns [26].

The NBTI model we use is based on recent work by Zafar et al. at IBM, and is a physics-based model verified using new and published NBTI failure data [26]. The model shows that MTTF due to NBTI has a large dependence on temperature. The MTTF due to NBTI at a temperature, T, is given by:

$$MTTF \propto [(ln(\frac{A}{1+2e^{\frac{B}{kT}}}) - ln(\frac{A}{1+2e^{\frac{B}{kT}}} - C)) \times \frac{T}{e^{\frac{-D}{kT}}}]^{\frac{1}{\beta}} \quad (11)$$

where A, B, C, D, and β are fitting parameters, and k is Boltzmann's constant. Based on the data in [26], the values we use are $A = 1.6328$, $B = 0.07377$, $C = 0.01$, $D = -0.06852$, and $\beta = 0.3$.

4 Structural Redundancy for Lifetime Reliability

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

Structural Duplication (SD): In SD, extra structural redundancy is added over and above the required base processor resources during microarchitectural specification. The extra structures that are added are designated as *spares*, and are power gated and not used at the beginning of the processor's lifetime. During the course of the processor's life, if a structure with an available spare fails, the processor reconfigures and uses the spare structure. This ex-

tends the processor's life beyond the point when it would have normally failed, and instead, processor failure occurs only when a structure without a spare, or all available spares fail. It is important to note that spare structures are added over and above the required processor resources for optimal performance. Most modern high-performance processors have enough redundancy to exploit all the available parallelism in common applications, resulting in very little performance benefit from the spares. As a result, the spares would be power gated to prevent any unnecessary wear-out, and would be powered on only when the original structure fails.

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

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

Unlike SD, GPD does not add an area overhead to the base processor as no extra units are added. However, disabling redundant structures that fail lowers the processor's performance for the latter part of the processor's lifetime. Hence, the *guaranteed* performance of a processor with GPD is its performance in the fully degraded state. We report GPD results for both guaranteed and actual performance in Section 6.2.

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

Figure 2 illustrates the differences between the three techniques. Consider a base processor with two structures, A and B. Now, if the lifetimes of structures A and B for a random instance of the base processor are t_A and t_B, the base processor's lifetime in that instance is $MIN(t_A, t_B)$, as the first structure to fail would cause the processor to fail. Next, consider the base processor with SD, where another structure C is added as a spare

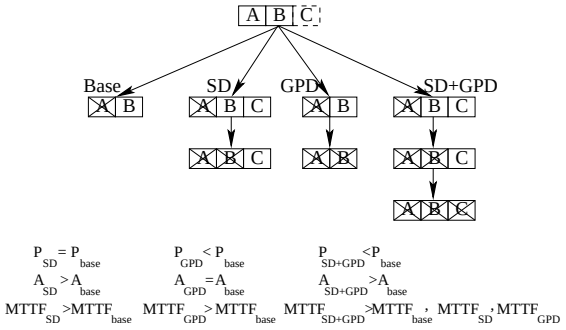


Figure 2. Steps to failure for a base processor, base processor with SD, with GPD, and with SD+GPD. The relationship between the performance (P), area (A), and MTTF of each of the processors is also given.

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

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

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

4.1 Design Issues

A key requirement for SD, GPD, and SD+GPD is the ability of the processor to detect and disable structures that have failed during normal processor operation. Detecting errors is a critical issue for hard and soft error tolerance, and there is significant ongoing work on detection techniques. However, much work still has to be done on the subject – currently, efficient detection techniques with high coverage for processor logic do not exist, and a detailed discussion of such functionality is beyond the scope of this paper. However, we expect detection and coverage issues to impact SD and GPD similarly, allowing a relative comparison of the techniques.

Also, both SD and GPD require additional hardware for detection and disabling/enabling of failed units. This extra

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

Table 1. Base 65 nm POWER4-like processor. Some of the buffer and cache sizes are different from those in the actual POWER4 processor.

hardware and resultant wiring will adversely affect processor power and performance (due to the larger communication distance between critical units). Accounting for these effects requires a detailed design for these techniques which is beyond the scope of this paper. Therefore, we do not account for these overheads in the results in this paper.

5 Experimental Methodology

5.1 Base Processor and Performance Simulation

The base processor we use for our simulations is a 65nm, out-of-order, 8-way superscalar processor, conceptually similar to a single core POWER4-like processor [24]. The 65 nm processor parameters were derived by scaling down parameters from the 180nm POWER4 processor [23]. Although we model the performance impact of the L2 cache, we do not model its reliability as its temperature is much lower than the processor core [24] resulting in very few L2 cache hard failures. Table 1 summarizes the base 65nm processor modeled.

Our architectures are modeled using Turandot, a trace-driven research simulator developed at IBM's T.J. Watson Research Center [16]. As described in [17], Turandot was calibrated against a pre-RTL, detailed, latch-accurate processor model. Despite the trace-driven nature of Turandot, the extensive validation methodology provides high confidence in its results.

5.2 Power, Temperature, and Reliability Models

To estimate processor power dissipation, we use the PowerTimer toolset developed at IBM's T.J. Watson Research Center [5]. This toolset, in its default form, is built around the Turandot cycle-accurate performance simulator referred to in the previous section. The power models that are built into the Turandot-based PowerTimer are based on circuit accurate power estimations from the 180nm POWER4 processor [24]. For our simulations, we use realistic clock gating assumptions in PowerTimer, in tune with actual data available from current generation microprocessors.

For temperature simulation, we use the HotSpot tool [20]. HotSpot models temperature at a structural level (using power information from PowerTimer). The large time constant of the processor heat sink prevents significant heat sink changes from occurring during simulations [20]. As a result, HotSpot has to be initialized with an accurate heat sink temperature for every simulation. For this purpose, we run everything twice – the first run is used to obtain the average power consumption of the processor which can be used to initialize the temperature of the heat sink. Once the heat sink is initialized, the second run produces accurate temperature results.

We use an area based leakage power model, with a leakage power density of 0.60 W/mm^2 at 383K. This value is a rough estimate, based on leakage trends for 65nm processors of the type and complexity of the POWER4, and assumes standard leakage power control techniques like the use of high-threshold devices in non-critical logic paths and arrays. We also model the impact of temperature on leakage power using the technique in [9]. At a temperature T, the leakage power, $P_{leakage(T)}$, is given by $P_{leakage(T)} = P_{leakage(383K)} \times e^{\beta(T-383)}$ where β is a curve fitting constant with a value of 0.017 [9].

As discussed previously, we use an enhanced version of RAMP [22] for reliability measurements. For a simulated application, based on temperature estimates from HotSpot and power estimates from PowerTimer sampled at a granularity of 1 μsecond, RAMP calculates an MTTF estimate for each structure and failure mechanism on the processor. The Monte-Carlo simulation method is then used to determine the MTTF of the processor.

5.3 Die Cost Model

In order to evaluate the cost impact of area increases imposed by structural duplication, we use the Hennessy-Patterson die cost model [8]. The cost, C, of a die of area, A is:

$$C \propto \frac{1}{(\frac{\pi r_{wafer}^2}{A} - \frac{2\pi r_{wafer}}{\sqrt{2A}})} \times (1 + \frac{DA}{\alpha})^{\alpha} \qquad (12)$$

where r_{wafer} is the wafer radius, D is the defects per unit area during manufacture of the wafer, and α is a parameter that corresponds inversely to the number of masking levels. We assume a $300mm$ wafer process, $D = 0.6$ per square centimeter, and $\alpha = 4.0$ [8]. In our experiments, we normalize our base processor cost to 1.0 (for a base area of $11.52mm^2$).

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

Table 2. Maximum temperature seen for Spec 2000 benchmarks

5.4 Workload Description

Our experimental results are based on an evaluation of 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 [16], and was generated using the full reference input set. Sampling was used to limit the trace length to 100 million instructions per program. The sampled traces have been validated with the original full traces for accuracy and correct representation [11].

5.5 Processor Configurations Evaluated

The base 65nm POWER4-like processor evaluated has a total area of $11.52mm^2$. The chip is divided into 7 distinct structures: floating point unit (FPU), fixed point unit (FXU), instruction decode unit (IDU), instruction scheduling unit (ISU), load store unit (LSU), instruction fetch unit (IFU), and branch prediction unit (BXU).

5.5.1 SD Configurations

To limit our configuration space, we do not allow all the structures on chip to be replicated individually for SD. Instead, we clubbed the processor's structures into 5 logical groups that can be replicated for spares – FPU, FXU, BXU+IFU, LSU, IDU+ISU. Table 3 summarizes these groups and the area overhead imposed on the processor by replicating each group. With these 5 groups, based on whether a group is replicated or not in the processor, we create 32 (2^5) SD configurations. If more than one group is replicated, the area overhead for that processor is the sum of the areas of the replicated groups.

5.5.2 GPD Configurations

Like SD, we limit our configuration space in GPD by not allowing every structure to degrade individually. Instead, the structures are grouped into 4 logical groups that can degrade – FPU, FXU, BXU+IFU, LSU. Unlike structural duplication, we do not allow the IDU+ISU to degrade. Each group can be in one of two states, full size or degraded to half size. That is, the group can be fully functional, or if a failure occurs in a structure, the half of the group that contains the failure would be shut down (although many structures like the caches can degrade to levels other than half size, we do not study them to limit the configuration space). With these 4 groups, based on whether a group is allowed to degrade to half size or not, 16 (2^4) configurations including the base can be created. Table 3 shows the configuration of the groups before and after degradation.

5.5.3 SD+GPD configurations

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

6 Results

6.1 SD Results

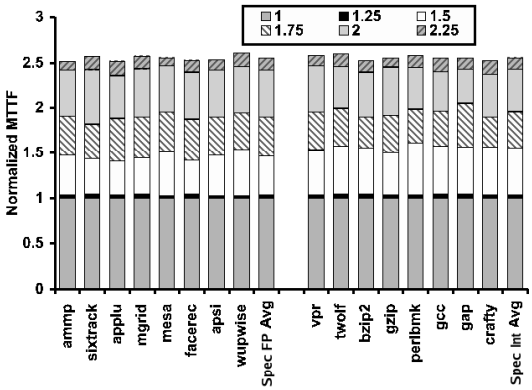


Figure 3. Reliability benefit from SD for different costs. The vertical axis shows normalized MTTF, with the MTTF of the application on the base processor normalized to 1.0 (the bottom segment of each bar). Each additional segment in the bars represents the normalized gain in MTTF from moving to higher costs.

Figure 3 shows the SD reliability benefit for various cost points for each of our applications, and also the average for all SpecFP and SpecInt applications. The vertical axis shows normalized MTTF. The results are presented in a stacked-bar format. The MTTF of each application on the base processor (which has a cost of 1.0), is the lowest segment in each bar, and is normalized to 1.0. Each additional segment in the bars represents the incremental normalized MTTF benefit obtained from moving to higher costs. For each segment, we selected the SD configuration that had

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

Table 3. Groups replicated in SD and allowed to degrade in GPD. The IDU+ISU is not allowed to degrade. The areas of each group for SD and the structures in the original and degraded group for GPD area also given.

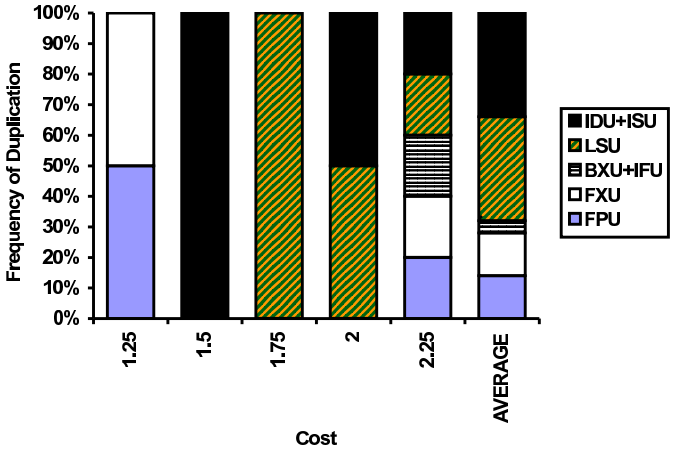
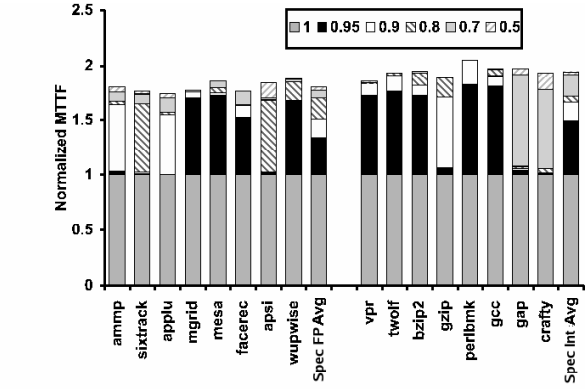


Figure 4. Fraction of applications for which different groups of structures are chosen for duplication with SD, for different costs. The average frequency across all costs is also given.

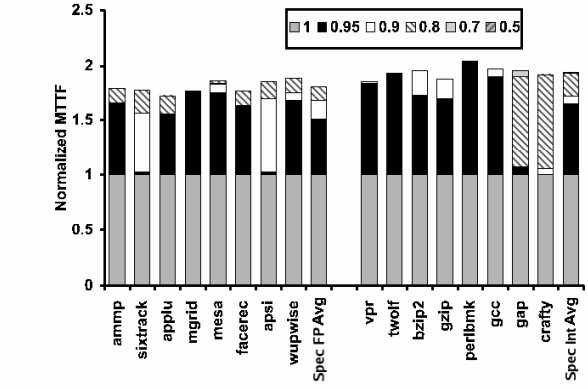
the highest MTTF among the configurations that satisfied the cost requirement. Figure 4 shows the fraction of applications for which different groups of structures are chosen for duplication with SD, for different costs. In addition, the average frequency across all costs is also shown.

As seen in Figure 3, SD provides significant reliability benefit, particularly for higher cost values. At a cost of 2.25 times the base cost, SD provides an average MTTF 2.53 times better than base MTTF. However, at a cost of 1.25 times the base cost, the MTTF is only 4% greater than base MTTF. These results can be understood with Figure 4 – for costs less than 1.5 times the base cost, only the FPU and FXU are chosen for duplication. Although the FPU and FXU do not provide large reliability benefit, they are the only structures that have areas small enough to satisfy the cost limit at 1.25 times the base cost (Table 3). As we move to higher cost points (left to right in Figure 4), larger structures which have higher failure rates can be duplicated, resulting in significant impact on reliability. At 1.5 times the base cost, the IDU+ISU can be duplicated, and at 1.75 times the base cost, the LSU can be duplicated. For points beyond 1.75 times the base cost, combinations of structures are used in SD. Finally, from the average bar in Figure 4, we can see that the FPU and FXU are chosen equally often. This is due to our equal mix of SpecFP and SpecInt applications.

6.2 GPD Results



(a) Guaranteed Performance



(b) Actual Performance

Figure 5. Reliability benefit from GPD for different (a) *guaranteed* **and (b)** *actual* **performance levels. The vertical axis shows normalized MTTF, with the lowest segment in each bar representing the normalized base MTTF of the application (performance of 1.0). Each additional segment shows the incremental MTTF benefit from moving to lower performance values.**

Figures 5(a) and (b) show the GPD reliability benefit for various performance levels for each of our applications, and also the average for all SpecFP and SpecInt applica-

tions. Like Figure 3, the vertical axis represents normalized MTTF. The MTTF of each application on the base processor (which has a performance of 1.0), is the lowest segment in each bar, and is normalized to 1.0. Each additional segment shows the incremental benefit from moving to lower performance. Figure 5(a) shows *guaranteed* performance values, while Figure 5(b) shows *actual* performance values. Unlike SD, where the cost overhead of a configuration applies for the entire lifetime of the processor, the performance degradation in GPD is not seen for the entire lifetime of the processor. At the beginning of the processor's lifetime, it will run at full performance. The degraded performance level is encountered only after one or more structures on chip fail. Due to the statistical nature of wear-out failures, for a given processor, no performance greater than the degraded value can be *guaranteed* (in a random batch of processors, some might have structures failing immediately). Figure 5(a) presents GPD results for this lowest guaranteed performance level. However, most processors will have a higher *actual* performance (which is the time-weighted average of all the IPCs seen during the lifetime of the processor). These actual performance values are reported in Figure 5(b). For each performance value (guaranteed or actual), we identified the GPD configuration which had the highest MTTF among the configurations which satisfied the performance requirement.

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

The results in Figure 5 show that processor resources in current high performance microprocessors likely exceed the requirements for performance and functionality of many applications. Most applications do not regularly use all the extra replicated units. As a result, when a failure occurs in one of these relatively unused structures, the processor can degrade to half the structure's size without a significant loss in performance, but with large reliability benefit. Once all the structures that are not used have degraded, further performance reductions result in much smaller reliability benefit.

As in Figure 4, Figure 6 shows the fraction of applications for which different groups of structures are chosen for degradation with GPD, for different performance levels. The average frequency across all performance levels is also given. Unlike SD where different structures were chosen

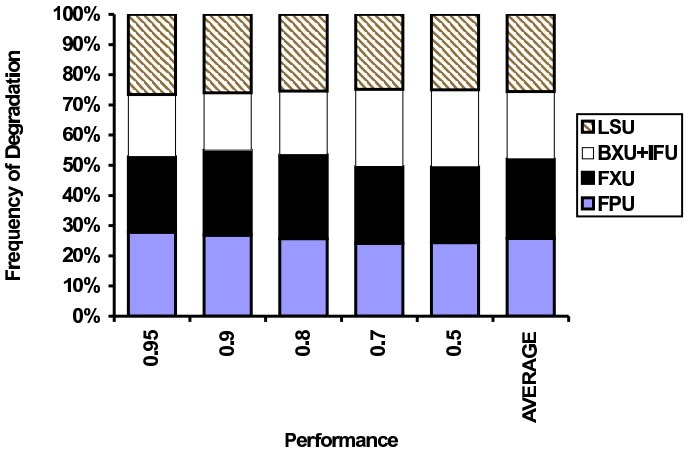


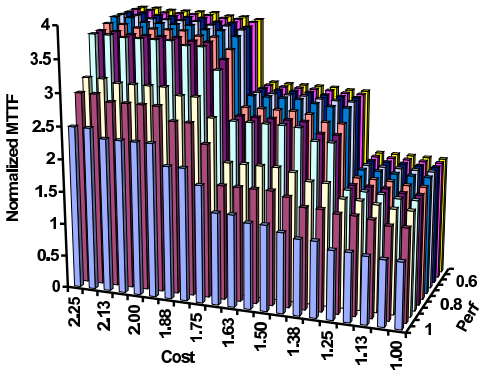
Figure 6. Fraction of applications for which different groups of structures are chosen for degradation with GPD, for different performance levels. The average frequency across all performance levels is also given.

for duplication at different costs, all structures are chosen with nearly the same frequency for degradation in GPD. For higher performance values (left side of Figure 6), the frequencies are similar because different applications in our workload rely on different processor structures for performance. This shows that no structure in the fully functional state is performance critical for all applications. For lower performance values (right side of Figure 6), the frequencies are similar because most applications have reached the fully degraded state, shutting down half of every structure.

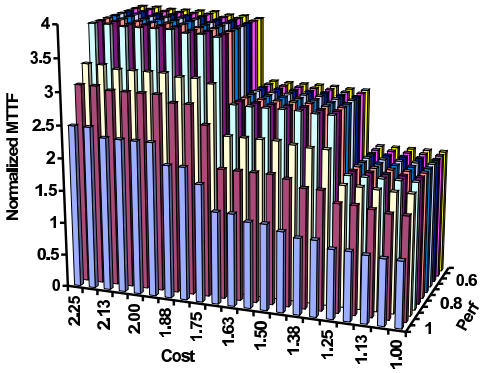
6.3 SD+GPD Results

Figures 7(a) and (b) show the reliability benefit from combining SD and GPD. The figures show the highest MTTF possible for each cost and performance constraint, averaged across all applications. That is, for each point with cost=C and performance=P, we report the highest MTTF (averaged across all applications) among all the SD+GPD configurations with cost $\leq$ C and performance $\geq$ P. Each MTTF value (represented by the height of the bars) is the average normalized MTTF across all applications, where the average MTTF at a performance of 1.0 and a cost of 1.0 (no SD or GPD) is normalized to 1.0. In the figure, when performance is 1.0, the values show average MTTF using only SD. When cost is 1.0, the values show average MTTF using only GPD. Every other point in the figures shows average MTTF for some degraded performance level and cost value (SD+GPD). Like Figures 5(a) and (b), Figures 7(a) and (b) represent *guaranteed* and *actual* performance levels, respectively.

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



(a) Guaranteed Performance



(b) Actual Performance

Figure 7. Highest SD+GPD MTTF (averaged across all our applications) possible for each cost and performance constraint. Each MTTF value (represented by the height of the bars) is the average normalized MTTF across all applications, where the average MTTF at a performance of 1.0 and a cost of 1.0 is normalized to 1.0

point, a guaranteed loss of 50% or an actual loss of 15% in performance (performance value of 0.5 in Figure 7(a) and 0.85 in Figure 7(b)), coupled with a cost 2.25 times the base cost, provides an average MTTF 3.89 times better than base MTTF. As discussed in Section 6.1, SD provides low average reliability benefit at very low cost values, but large benefits at higher cost values, for any given performance level. Similarly, as discussed in Section 6.2, GPD provides a larger incremental reliability benefit for smaller performance degradations (larger performance values), for any given cost. Also, the overall increase from SD is higher than that for GPD. Finally, as expected, much smaller decreases in actual performance provide the same reliability benefit as larger decreases in guaranteed performance. As explained earlier, this is due to the processor running at full performance at the beginning of its lifetime.

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

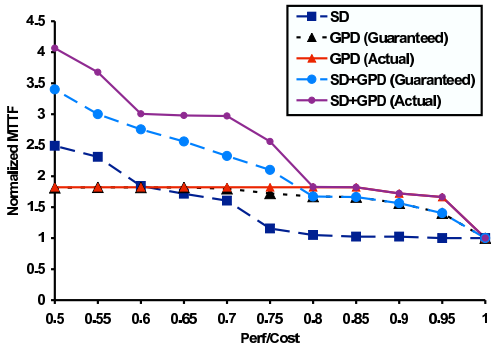


Figure 8. Average normalized MTTF benefit versus $\frac{P}{C}$ for SD, GPD, and SD+GPD across all applications. For GPD and SD+GPD, both *guaranteed* and *actual* performance values are given.

In order to understand performance and cost tradeoffs simultaneously, we use the ratio of performance and cost ($\frac{P}{C}$), a standard industrial metric, to evaluate SD, GPD, and SD+GPD. The normalized $\frac{P}{C}$ for all our applications on the base processor is 1.0. In SD, cost will increase, leading to $\frac{P}{C}$ values lower than 1.0. In GPD, performance will decrease, leading to $\frac{P}{C}$ values lower than 1.0, and in SD+GPD, both increases in cost and decreases in performance lower the value of $\frac{P}{C}$. Figure 8 shows the average MTTF benefit across all our applications from each of the three techniques for a range of $\frac{P}{C}$ values. The vertical axis represents normalized MTTF. The horizontal axis represents different $\frac{P}{C}$ design points. For both GPD and SD+GPD, both guaranteed and actual performance levels are given.

The results in Figure 8 clearly reflect the trends seen in Figures 3, 5, and 7. At high $\frac{P}{C}$ values (low performance or cost overhead), GPD provides much more benefit than SD. However, the benefit from GPD tapers off as we move to lower values of $\frac{P}{C}$. On the other hand, SD provides much more MTTF benefit at lower $\frac{P}{C}$ values, and overtakes GPD. The combination of both techniques always provides the highest MTTF benefit. This is intuitive because SD+GPD can choose any configuration SD or GPD can choose, in addition to the cross product of the two. However, SD+GPD chooses the same configurations as GPD chooses at high

values of $\frac{P}{C}$. Finally, since processors run at full performance at the beginning of their lifetime, the same MTTF benefit for GPD (Actual) and SD+GPD (Actual) comes at higher $\frac{P}{C}$ values than GPD (Guaranteed) and SD+GPD (Guaranteed).

6.5 Discussion

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

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

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

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

7 Conclusions

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

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

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

We also enhance the RAMP reliability model by addressing some of its limitations. In particular, we incorporate time dependence in RAMP's failure mechanisms by modeling them as lognormal distributions, and use Monte-Carlo methods to calculate processor lifetimes. We also add a failure model for a critical emerging failure mechanism, NBTI.

This paper has focused on an analysis of the benefits of structural redundancy for reliability. For such techniques to be used in practice, several design issues need to be addressed. Specifically, techniques to efficiently detect and disable/enable failed structures need to be developed. Given that detection techniques are unlikely to offer 100% coverage, our model must incorporate the incomplete coverage.

8 Acknowledgments

We would like to thank Sufi Zafar and Zhigang Hu of IBM T.J. Watson Research Center for their help with the NBTI model and Turandot simulator respectively.

References

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

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

[3] Methods for Calculating Failure Rates in Units of FITs. In *JEDEC Publication JESD85*, 2001.

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

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

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

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

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

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

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

[11] V. Iyengar, L. H. Trevillyan, and P. Bose. Representative Traces for Processor Models with Infinite Cache. In *Proc. of the 2nd Intl. Symp. on High-Perf. Comp. Architecture*, 1996.
[12] R. A. Johnson. *Probability and Statistics for Engineers*. Prentice Hall of India, 2002.
[13] I. Koren et al. Defect Tolerant VLSI Circuits: Techniques and Yield Analysis. In *Proceedings of the IEEE*, 1998.
[14] H. Masuda et al. Assessment of a 90nm PMOS NBTI in the Form of Products Failure Rate. In *Proceedings of the IEEE 2005 International Conference on Microelectronic Test Structures*, 2005.
[15] J. McPherson et al. Comparison of E and 1/E TDDB Models for Si02 under long-term/low-field test conditions. In *Technical Digest IEEE International Electron Devices Meeting (IEDM)*, 1998.
[16] M. Moudgill et al. Environment for PowerPC microarchitectural exploration. In *IEEE Micro*, 1999.
[17] M. Moudgill et al. Validation of turandot, a fast processor model for microarchitectural exploration. In *IEEE Intl Perf., Computing, and Communications Conf.*, 1999.
[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, 1992.
[19] P. Shivakumar et al. Exploiting Microarchitectural Redundancy for Defect Tolerance. In *21st Intl. Conf. on Comp. Design*, 2003.
[20] K. Skadron et al. Temperature-Aware Microarchitecture. In *Proc. of the 30th Annual Intl. Symp. on Comp. Arch.*, 2003.
[21] 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.
[22] J. Srinivasan et al. The Case for Lifetime Reliability-Aware Microprocessors. In *Proc. of the 31st Annual Intl. Symp. on Comp. Architecture*, 2004.
[23] J. Srinivasan et al. The Impact of Technology Scaling on Lifetime Reliability. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, 2004.
[24] J. M. Tendler et al. POWER4 System Microarchitecture. In *IBM Journal of Research and Development*, 2002.
[25] K. Trivedi. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice Hall, 1982.
[26] S. Zafar et al. A Model for Negative Bias Temperature Instability (NBTI) in Oxide and High-K pFETs. In *2004 Symposia on VLSI Technology and Circuits*, June, 2004.