Appears in the 21^{st} International Conference on Computer Design

Exploiting Microarchitectural Redundancy For Defect Tolerance

Premkishore Shivakumar Stephen W. Keckler Charles R. Moore Doug Burger
Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin
cart@cs.utexas.edu - www.cs.utexas.edu/users/cart

Abstract

The continued increase in microprocessor clock frequency that has come from advancements in fabrication technology and reductions in feature size, creates challenges in maintaining both manufacturing yield rates and long-term reliability of devices. Methods based on defect detection and reduction may not offer a scalable solution due to cost of eliminating contaminants in the manufacturing process and increasing chip complexity. This paper proposes to use the inherent redundancy available in existing and future chip microarchitectures to improve yield and enable graceful performance degradation in fail-in-place systems. We introduce a new yield metric called performance averaged yield (Y_{PAV}) *which accounts both for fully functional chips and those that exhibit some performance degradation. Our results indicate that at 250nm we are able to increase the Y_{PAV} of a uniprocessor with only redundant rows in its caches from a base value of 85% to 98% using microarchitectural redundancy. Given constant chip area, shrinking feature sizes increases fault susceptibility and reduces the base Y_{PAV} to 60% at 50nm, which exploiting microarchitectural redundancy then increases to 99.6%.*

1 Introduction

The bulk of the performance improvement in microprocessors in recent years has come from increases in clock frequency predominantly achieved by aggressive reductions in technology features sizes from $10\mu m$ to $0.13\mu m$, and on-chip transistor counts that have soared from 2,300 to over 100 million. While technology trends suggest chips with clock frequencies in the multigigahertz range containing over a billion transistors by the end of the decade, two substantial challenges must be addressed to enable practical deployment of such systems. First, shrinking lithography, new materials and process technologies, and lower design tolerances make integrated circuits more susceptible to manufacturing defects, requiring substantial investments to maintain chip yield at acceptable levels. Second, some manufacturing defects are latent and manifest themselves only after the chips have been deployed and run for some period of time. As larger commercial and scientific systems are constructed from hundreds or thousands of processors, the probability and frequency of latent failures increase.

In this paper we examine the redundancy already available within modern microprocessors that can be used to improve chip yield and enhance the graceful degradation of fail-in-place systems. While modern chips are typically declared *functional* only if *all* of the components are fully functional (taking into account redundant rows to increase yield in caches), we propose that chips with some non-functioning components are still useful and can contribute to both overall yield and gracefully degraded components in a fail-in-place system. Today, it has also become common for manufacturers to separate chips that are for sale into speed bins based on their operating frequency, and recently some have made use of a more general *performance binning* strategy that separates parts into bins of guaranteed performance levels rather than bins based solely on operating frequency [11]. We propose that designs that include replicated or non-essential functions in support of increased performance be enhanced with the capability to disable some of these structures in face of defects detected within the circuitry. Chips of different end-performance, corresponding to different degraded configurations, can be offered at different prices, extending the current manufacturers use of speed binning. We formalize this notion of performance binning, and propose a new yield metric called *performance averaged yield* (Y_{PAV}) in which the total yield is a function of the performance range of each bin and the number of chips in the bins. Our results indicate that the Y_{PAV} for a uniprocessor can be improved from 85% to 98% with certain assumptions about defect density and defect size. For chip-multiprocessor architectures at future technologies, we show that microarchitectural redundancy provides substantial benefits achieving Y_{PAV} of up to 99.6%.

The remainder of this paper is organized as follows. Section 2 provides background on yield loss and some related work in yield enhancement. Section 3 identifies and classifies the types of redundancy in modern microprocessors and describes the mechanisms required to exploit it for yield and fail-in-place enhancement. Section 4 describes the details of our yield, area, and performance models. Section 5 presents results showing the yield benefits of microarchitectural redundancy as a function of technology, defect characteristics, and architecture. Section 6 summarizes our findings and describes the synergy between this work and other design trends.

2 Background and Related Work

Yield loss over time can be divided into an initial phase of technology deployment dominated by systematic failures, with an eventual crossover to a more mature phase dominated by random defects [17]. Future technology advancements are expected to involve continued shrinking of feature sizes, and the introduction of new process steps and materials increasing the yield sensitivity to design features, introducing new sources of systematic defects, and requiring a feature-based methodology to quantify yield loss [12]. Though we focus primarily on the yield loss due to random defects, we recognize that many of the techniques discussed here will also help in identifying more usable chips during the initial technology learning phase. In this paper we push the traditional techniques of yield enhancement, based on detecting and either disabling or reconfiguring the faulty resources [10], inside the boundaries of a single processor by identifying and exploiting redundancies at the microarchitectural level. Due to the enormous expense of suitable testers and test time with growing chip complexity, compounded by the fine-grained nature of redundancy proposed in this paper, we envision the need for more advanced BIST controllers that build on the capability that exists for array repair to include support for other types of fault tolerance mechanisms [6].

There are two classes of work related to the *performance averaged yield* concept. In [20], yield evaluation is done for memory chips with redundancy that allows the chip to be partitioned so that the fault-free sections can operate independently. The *equivalent yield* concept proposed in that paper accounts for partially good chips by scaling the yield by the memory capacity of the degraded chip. In this paper, we extend the argument to processor chips and propose a performance based metric that is a better measure of the effect of chip degradation at the system level. Performability [14] was proposed as a refined measure of availability by accounting for the degraded performance dynamically. *Performance averaged yield* adapts this dynamic concept to static chip yield evaluation, while recognizing the fact that it directly applies to fail-in-place systems with runtime BIST and repair.

3 On-Chip Redundancy Model

This section describes each redundancy model and our implementation of the redundancy models in the different processor components. In the future, many chips will likely contain multiple processors, when we can imagine a set of *intra-processor* redundancies as well as *inter-processor* redundancy at the next level of hierarchy. As a basis for analysing the effects of the different redundancy types, we have defined a processor model (Table 1) that is similar to the Alpha 21264 [9]. Both the integer and floating point clusters are symmetric and each have 2 functional units within them. The processor also has an on-chip L2 cache of 1MB. The *spare entries* provided in the components are used only in the face of defects and do not contribute to additional performance. We identify three primary types of redundancy as a basis for our redundancy model (Figure 1).

Component Level Redundancy (CLR): In CLR, the component is typically replicated to provide additional performance through parallelism, but only a subset is actually required for correct functionality. Each component that has CLR has a *resource* line associated with it, and the component's BIST module sets the *resource* line to be permanently BUSY in the event the component is disabled due to internal faults. The parent control logic already contains mechanisms that restricts its use each cycle to only those components whose resource lines are FREE. For instance, the instruction scheduling logic is implemented using wakeup arrays that contain *RESOURCE AVAILABLE* lines indicating which resources are FREE in the given cycle [4]. The execution clusters and the internal ALUs of the processor are covered by the CLR model. The hierarchical nature of the CLR for the clusters and ALUs provides coverage over the control logic of the individual clusters.

Array Redundancy (AR): When defects are detected in rows or columns of bit cells in the main body of the array, the AR mechanisms can be configured to effectively steer the decode towards the redundant entry rather than towards the bad row or column. From a yield perspective, AR is attractive because a relatively small investment in area can offer excellent defect tolerance for the entire structure. The set associative L1 and L2 caches are way-interleaved allowing both CLR (disabling one set) and AR (redundant row steering within one of the operational sets). The CLR for the caches provides coverage over the peripheral logic of the individual cache banks also. Consistent with the accepted design practice [8], the redundant rows and columns are about 2.5% of the base cache capacity. The TLBs are also covered by the AR model.

Dynamic Queue Redundancy (DQR): A *valid bit* is added to each queue entry that has DQR. If a particular queue entry has defects, it can be permanently disabled by clearing the *valid bit*, thus decreasing the number of *available* entries. The existing protocols that add queue entries are modified to stall the machine when the *available* queue entries are full. Downstream queue access logic is also augmented so that the queue entries marked *invalid* are never processed. In highly pipelined designs, as well as designs that support dynamic reordering of operations, many structures such as the reorder buffer, the issue window, the register remappers, the load and store buffers are implemented as queues. In our implementation of DQR, we include spare queue entries to provide some defect tolerance without losing any performance, similar to AR. Nevertheless, our ex-

Processor Redundancy Configuration			
Microarchitecture Resource	Base capacity / Spare entries	Redundancy Model	Minimum operational size
INT, FP Instruction Window	20 / 1	DQR	20
INT, FP Register File	80 / 2	DQR	80
INT, FP Map Table	32 / 1	DQR	32
Execution units per cluster (INT Alu, FP Alu, INT Mult, FP Mult)	2 / 0	CLR	1
INT, FP Clusters	2 / 0	CLR	1
Reorder Buffer	80 / 2	DQR	80
Load / Store queue	32 / 1	DQR	32
TLBs (Fully associative)	128 / 2	AR	128
L1 I, D cache (2-way associative)	64KB / 1.5KB	AR, CLR	32KB
L2 cache on-chip	1MB / 24KB	AR, CLR	0MB

Table 1. Processor redundancy configuration

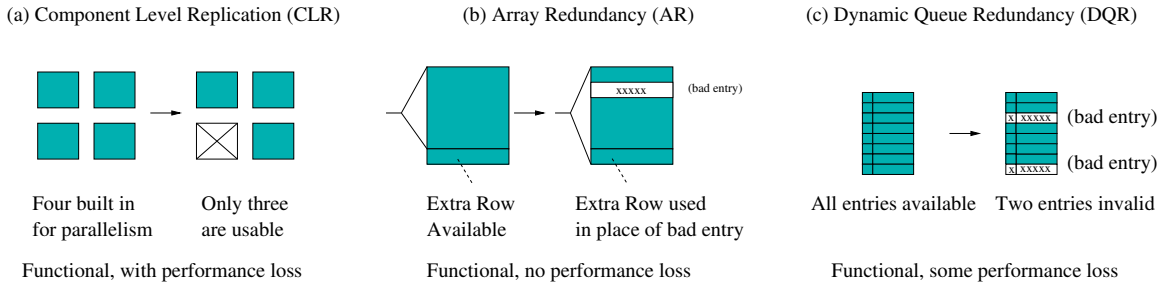

Figure 1. Basic Redundancy Models.

periments show that disabling one or two entries in most of these queues results in at most 1% loss in performance.

Elements of the processor not listed in the table, like random control logic, and logic that is used to implement the redundancy model itself, have no redundancy coverage in our example design. Nevertheless, approximately 85% of the total area of the processor has coverage through redundancy, as compared to 50% with AR alone in the L1 and L2 caches. This configuration and aggregate model is used for the uniprocessor and multiprocessor yield analysis described throughout this paper. Since mainstream processors [2] already employ redundant rows and columns in caches, the baseline yield (Y_{BASE}) corresponds to a processor with AR in the L1 and L2 caches.

4 Methodology

Our methodology for calculating overall chip yield integrates a basic yield model and a microprocessor area model with the redundancy model of the chip components. The yield of the chip computed thus is then linked with its measured end-performance across the range of different configurations to obtain the *performance averaged yield* (Y_{PAV}). The remainder of this section describes each of these models in greater detail.

4.1 Random Defect Limited Yield Model

In this paper, we have adopted the Poisson Yield model for modeling the random yield component [13]. Our methodology can be extended to use other commonly studied and more detailed yield models such as the Negative Binomial model [10]. The Poisson Yield (Y_P) Model models the random defects to be completely independent and is described by:

$$Y_P \propto e^{(-D0 \times A \times KR)} \quad (1)$$

where $D0$ is the defect density measured in defects per cm^2, A represents the area of the component in cm^2, and KR is the kill ratio or the fraction of the total component area that is sensitive to defects. The kill ratio models the interaction between the defect size and the layout feature size, and increases as the ratio of defect size to the feature size increases. The ITRS [17] has set a target of 83% for the random-defect limited yield of microprocessors. We obtain a Y_{BASE} of 85.4% at 250nm using the defect density provided by the ITRS, for a normal defect to feature size ratio, and a chip area of $320mm^2$, validating our input parameters to the Poisson Yield model.

4.2 Chip Area Model

Estimation of individual component yield requires detailed area models of the processing cores and caches. We configured Cacti 3.0 [19], an integrated memory access time, energy, and area model, to derive area estimates of L1 and L2 caches, TLBs, register files, and all on-chip queues. To model the area of functional units we used an empirically derived, technology-independent area model [5]. To esti-

Structure	Percentage of total area
L2 cache	49.0%
L1 D cache	12.7%
L1 I cache	5.5%
Integer functional units	6.3%
Floating point functional units	6.7%
On-chip storage structures (except caches)	11.1%
Misc. components (BIU, PLL,I/O pads etc.)	6.0%
Random control logic	2.7%
Total Area at 250nm	$325mm^2$

Table 2. The Uniprocessor Model

mate the area of miscellaneous blocks such as I/O pads and clock distribution trees, we developed an empirical model based on our analysis of the Alpha 21264 floorplan [9]. We validated our area model against the Alpha 21264 microprocessor floorplan area [9] and calculated the error to be 3.8%. Table 2 shows the area of the processor model described in Section 3, and its distribution among its most significant components.

4.3 Overall Chip Yield Model

Section 3 describes how a single processor component may have more than one form of redundancy. If the multiple redundancies are non-hierarchical in nature, the overall component yield is simply equal to the product of the individual region yields corresponding to the different redundancy models. The individual yields can be composed using a simple product because the Poisson model treats defects as completely independent. On the other hand if the component has redundancies that compose hierarchically, we begin by applying the method at the lowest level at which the redundancies of the regions are non-hierarchical and then reapply the method recursively at each higher level of hierarchy. The chip area model is used to estimate the area of the different component regions. The redundancy model of a region is one of CLR, AR, DQR or *no redundancy*. The Poisson Yield model is used directly to calculate the statistical yield of a region with no redundancy. The method to calculate the yield of a region with one of the three primary redundancy schemes is described below.

Yield with the basic redundancy models: A redundancy model specifies the minimum number of working entries the component must possess to ensure correct overall functionality. The overall component yield is therefore the sum of the probabilities associated with all the configurations in which the component has at least the minimum number of working entries, out of the total number of entries including spares. The Y_R from this calculation is summarized using the well known binomial expansion:

$$Y_R = \sum_{i\,=\,mins}^{(bs+se)} C_{mins}^{(bs+se)} \times \mathrm{Pr(Good)}^{i} \times \mathrm{Pr(Bad)}^{(bs+se-i)} \quad (2)$$

where C_r^n is the *combinations* operator, $mins$ is the subset of entries required for correct functionality, bs represents the base number of entries in the component, and se is the number of spare entries. The probability of a entry being functional or invalid is computed using the Poisson Yield model. For example, caches that have AR are provided with enough redundant rows and columns to greatly improve yield and at the same time show no reduction from peak performance. Hence in this case $mins$ becomes equal to bs, and the value of se is dependent on the cache capacity. With CLR in the clusters, the processor could potentially have a configuration with only one functional cluster, in which case se is equal to zero and $mins$ is equal to one. Hence $Y_{OVERALL}$, our overall yield metric, not only includes the traditionally accounted fully functional chips but also includes chips with degraded components. The minimum subset of entries for each on-chip component determined by the specific redundancy model is given in Table 1.

4.4 Performance Averaged Yield Model

While $Y_{OVERALL}$ treats all the resulting yield configurations equally regardless of their degraded state, Y_{PAV} specifically aims to differentiate between fully functional chips and chips with degraded components. Our design of the Y_{PAV} metric achieves this by using the IPC of the resulting chip configuration as the discriminating measure. Using this formulation captures both the effects of redundancy—improvements in yield and reductions from peak performance. Adding three steps to the algorithm for computing $Y_{OVERALL}$ gives us Y_{PAV}. First, the $MAXIPC$ corresponding to the base configuration (which is the maximal configuration) is calculated. Each degraded configuration is then associated with a *relative IPC*, which is the ratio of its IPC to the $MAXIPC$. Finally the yield of each configuration is scaled by its *relative IPC* and accumulated to give Y_{PAV}. This is described by the equation:

$$Y_{PAV} = \sum_{i\,\in\,all\ configurations} Y_i \times \frac{IPC_i}{MAXIPC} \quad (3)$$

To evaluate the performance of the various degraded configurations we used the sim-alpha simulator [3] which models the Alpha 21264 core in detail. First, we configured sim-alpha to resemble our processor model. We further made modifications that enable us to simulate the different degraded configurations by selectively disabling on-chip components. We chose seven benchmarks (Figure 2) from the SPEC2000 benchmark suite and *sphinx*, a speech recognition benchmark, to provide a wide range of behavior in

	Benchmark category	Benchmark name	FFWD (x100M)	RUN	MaxIPC
INT	Memory intensive	181.mcf	336.3	100M	0.13
		sphinx	60	200M	0.57
	Processor bound	164.gzip	332	100M	1.76
		252.eon	207.3	100M	1.29
FP	Memory intensive	171.swim	1196	100M	1.02
		179.art	66.3	100M	0.26
	Processor bound	183.equake	193.4	100M	1.11
		177.mesa	639.9	100M	1.34

Figure 2. Benchmarks used for performance experiments

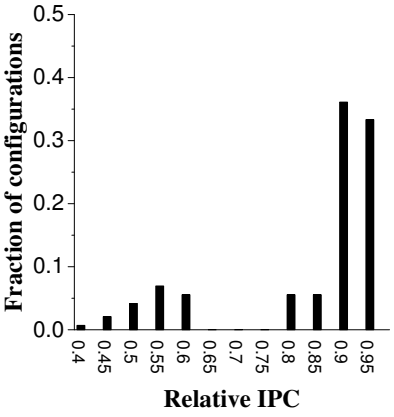

Figure 3. IPC distribution for the different configurations.

IntFus	4	3	2	4	4	4	4	4	2	2	1	1	4	2	1
FpFus	4	4	4	3	2	4	4	4	2	2	1	1	4	2	1
IL1 (KB)	64	64	64	64	64	32	64	64	64	32	64	32	64	32	32
DL1 (KB)	64	64	64	64	64	64	32	64	64	32	64	32	64	32	32
L2 (MB)	1	1	1	1	1	1	1	0.5	1	0.5	1	0.5	0	0	0
Rel IPC	1.0	0.97	0.93	0.98	0.99	0.97	0.97	0.94	0.93	0.85	0.73	0.69	0.65	0.50	0.44

Table 3. Relative IPCs for a few sample degraded configurations.

their usage of the memory system and the execution resources. Figure 2 also shows the number of instructions skipped to reach the start of the execution phase ($FFWD$), the number of instructions simulated (RUN), determined using SimPoint [18], and the maximum IPC for each benchmark at the base configuration.

Two important factors contribute to Y_{PAV} being nearly equal to $Y_{OVERALL}$. To describe the factors we plot the normalized IPC distribution for the range of all the allowed processor configurations (Figure 3), not accounting for their actual yield or likelihood of occurence. First, the graph shows that 80% of the configurations have a *relative IPC* (Rel IPC) greater than 0.8. The remaining configurations having *Rel IPC* around 0.55 correspond to the chips with a fully defective L2 cache. The left section of Table 3 shows harmonic mean *Rel IPCs* of a small subset of chip configurations having *Rel IPC* greater than 0.8. In the right section, the *Rel IPC* drops below 0.8, with the last column corresponding to our most degraded configuration. Second, there is enough redundancy in our processor model that most of the yield is also concentrated in configurations with high *Rel IPC*, and highly degraded configurations such as in the right section of the table never occur and hence provide no contribution to yield. Hence in all of the product terms contributing to Y_{PAV} (Equation 3) with non-zero yield (Y_i) the associated *Rel IPC* is close to one.

5 Results

In this section we present our results for the yield enhancement we observe at future technologies and chip microarchitectures as a function of the defect characteristics and the redundancy model.

5.1 Chip Topologies

Future chip microarchitectures have substantial flexibility in using the larger number of transistors that can fit in a given chip area. In the case of special-purpose processors, where the desired functionality remains fairly constant with time, the required performance can be achieved with no additional features in the processor architecture. As shown in Figure 4a, the area of the uniprocessor in the constant-architecture scheme decreases rapidly with decreasing feature size because the microarchitecture is kept constant. However, with successive microprocessor generations, the dominant trend in general purpose processor design has been to add microarchitectural features that enhance the processor's functionality and consume the extra silicon area. Figure 4b illustrates this constant-area uniprocessor model, where the relative proportions of the core area and the area occupied by caches is kept approximately constant. Technology scaling trends and considerations on multi-thread performance have influenced some emerging architectures to include multiple processors within a single

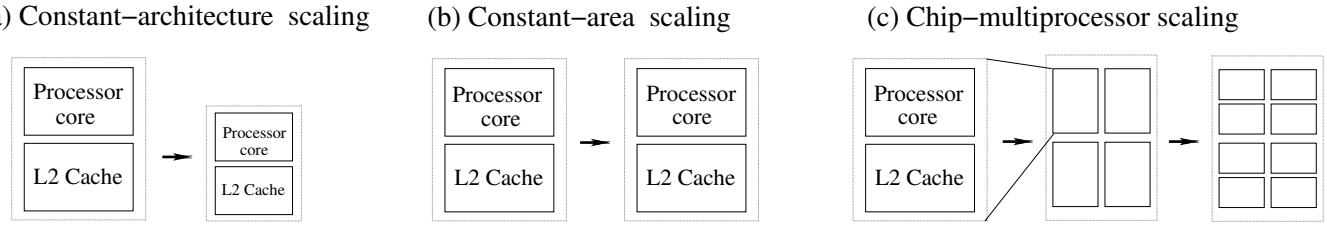

Figure 4. Chip topologies.

chip, which has substantial implications for yield. Figure 4c illustrates the CMP (Chip multiprocessor) model built using the constant-architecture uniprocessor model as the building block.

5.2 Uniprocessor Yield

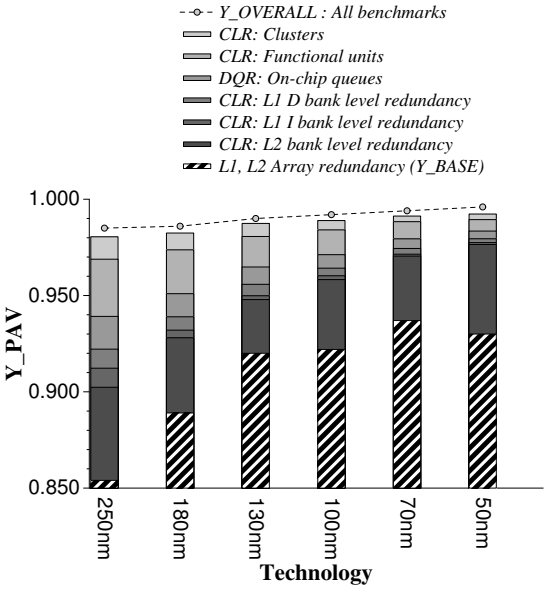

Figure 5. Yield for a constant-architecture uniprocessor model at normal defect size.

Constant-architecture Uniprocessor Yield: Figure 5 shows Y_{PAV} obtained by incrementally adding different flavors of on-chip redundancy to the constant-architecture uniprocessor model. For instance, at 100nm the maximum contribution comes from L2 bank level redundancy, and CLR in the functional units dominates among all the other types of redundancy, which together increase Y_{PAV} to 98.8%. Across technologies, Y_{BASE} increases from 85.4% to a maximum of 93.7% because the gain from the rapidly decreasing chip area outweighs the increased susceptibility to yield loss due to the higher kill ratio. Second, the contribution of L2 bank level redundancy continues to be significant, and all the other types of redundancy give progressively diminishing returns. This is because the L1 and L2 caches occupy almost 70% of the chip area and the absolute area occupied by the remaining components becomes vanishingly small at smaller feature sizes. Finally, Y_{PAV} increases from 98% at 250nm to 99.2% at 50nm, and since most of the configurations lie within 20% of maximum performance (Figure 3) $Y_{OVERALL}$ (indicated by the dotted line) is at most 0.4% above Y_{PAV} across all technologies. The above result is significant because it shows that even though Y_{BASE} improves with technology, Y_{PAV} can be further improved by adding microarchitectural redundancy.

Constant-area Uniprocessor Yield: Unlike the constant-architecture model Y_{BASE} decreases substantially from 85.4% at 250nm to 59.5% at 50nm, as the area of the chip components remain constant across technologies, and the kill ratio increases with decreasing feature size. Hence, exploiting the greater available redundancy at smaller feature sizes, whether in the form of more functional units, cache banks, offers greater improvements to Y_{BASE}, achieving $Y_{OVERALL}$ ranging from 98% at 250nm to 91.3% at 50nm. However, the increasing difficulty in achieving scalable performance by scaling an out of order superscalar uniprocessor, has induced the adoption of CMPs as discussed in the next section.

5.3 Multiprocessor Yield

In this paper, we explore two types of multiprocessor redundancy. In intra-processor redundancy, a chip can have its processors in any of the allowed internally degraded states, but the entire chip is considered bad once the available redundancy is exhausted in even one of its processors. On the other hand a processor in a chip with only inter-processor redundancy becomes useless if any fault resides in it. However, if enough of the remaining processors are functional, the chip can still be operational. In this paper we consider the chip to be functional as long as there is at least one good processor, but in practice our models never produce chip configurations with more than two bad processors per chip. We calculate chip performance as the aggregate performance of all the cores on the chip, since we model the multiple threads to be independent. The algorithm for calculating Y_{PAV} from Section 4.4 can be naturally extended to a multiprocessor by modifying each step to account for

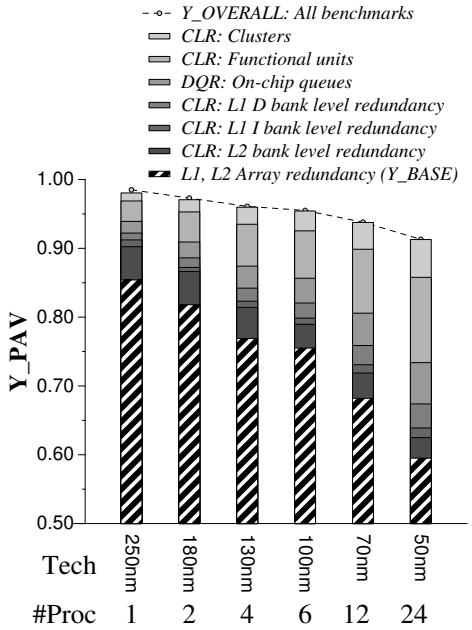

Figure 6. Yield with intra-processor redundancy at normal defect size.

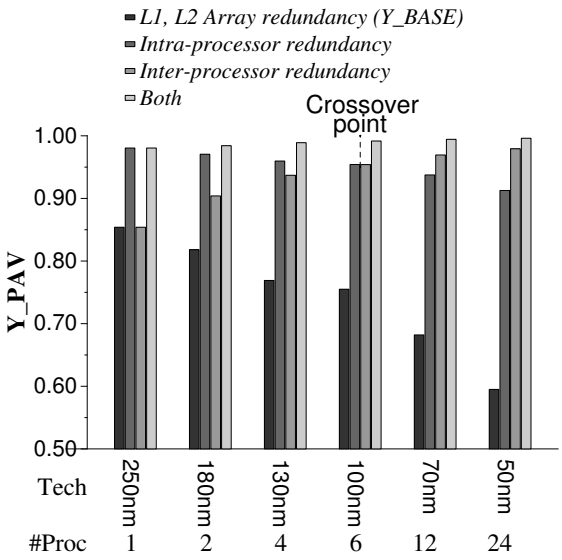

Figure 7. Comparison of Y_{PAV} for different redundancy models.

the IPC of the entire multiprocessor (whether the configuration is fully functional or degraded).

5.3.1 Yield with Intra-processor Redundancy

Figure 6 plots Y_{PAV}, across all technologies, obtained by incrementally adding redundancy to each processor in a multiprocessor chip with intra-processor redundancy. The x-axis shows the feature size and the number of processors per chip at each technology. At any given technology adding redundancy improves Y_{PAV} substantially, CLR in the functional units give maximum yield benefit, and the benefits from L2 bank level redundancy, DQR in the queues, and CLR in the clusters are comparable. For instance, at 70nm adding redundancy dramatically improves Y_{PAV} from 68.2% to 93.7%. There are three interesting features that can be observed across technologies. First, the Y_{BASE} decreases substantially from 85.4% at 250nm to 59.5% at 50nm, because the kill ratio increases considerably at smaller feature sizes. Second, the instances of intra-processor redundancy on the chip increases linearly with the number of processors, and as a result the addition of redundancy leads to greater improvements in yield at smaller feature sizes. For instance, at 180nm Y_{PAV} increases by 4.4% on adding CLR in the functional units, whereas it increases by 12.4% at 50nm. Third, the higher yield benefits, depending on the redundancy model, imply that more chips are degraded at smaller technologies. But as the area occupied by a single processor decreases, its Y_{BASE} increases (see Figure 5), and hence the probability of it being defective decreases. Combined with the increasing number of processors per chip, the fraction of degraded processors per chip decreases with technology. Hence, even though the number of degraded chips increases at smaller technologies, each resulting degraded chip configuration contains a majority of fully functional processor cores and very few degraded processors. As a result, Y_{PAV} continues to be within 0.2% of $Y_{OVERALL}$ at all technologies. Although there are significant benefits from adding redundancy, Y_{PAV} with all the types of redundancy drops from 98% at 250nm to 91.3% at 50nm due to higher kill ratio.

5.3.2 Comparison of Redundancy Models

Figure 7 compares Y_{PAV} obtained using four different redundancy models. With only AR, Y_{PAV} decreases rapidly from 85.4% at 250nm to 59.5% at 50nm. Having intra-processor redundancy alone achieves high Y_{PAV}, which decreases slightly from 98% at 250nm to 91.3% at 50nm. Inter-processor redundancy gives coverage over the entire area of the chip and hence Y_{PAV} increases uniformly from 85.4% at 250nm to 98% at 50nm. The yield benefits offered by intra and inter-processor redundancy crossover at 100nm because of the opposite trends in their Y_{PAV} across technologies. While this analysis assumes a constant defect density across technologies, larger defect densities will shift the crossover point to the right because the fault susceptibility per unit area of silicon increases, and hence fine grained redundancy becomes more appropriate. Also while our CMP design is composed of a number of relatively small Alpha 21264-like cores, future CMP designs may take advantage of much larger uniprocessor cores to achieve technology

scalable high performance over a wide range of applications [16]. Consequently, there will be fewer processors and significantly greater intra-processor redundancy than inter-processor redundancy per chip, which will again shift the crossover point to the right. Since intra and inter-processor redundancy offer different types of coverage, having both intra and inter-processor redundancy provides consistently high Y_{PAV} ranging from 98% at 250nm to 99.6% at 50nm, with a maximum improvement in Y_{PAV} of 3.75% over having only one of the types of redundancy.

6 Conclusions

This paper, examines the redundancy in modern microarchitectures that can be used to enhance their yield, and evaluates the trade-off between performance and yield within the context of microprocessors and chip multiprocessors. We propose a new yield metric called *performance averaged yield* (Y_{PAV}) which accounts for the level of performance degradation on all functioning chips. By exploiting microarchitectural redundancy we demonstrate that Y_{PAV} can be improved to as high as 99.6% at 50nm, with a maximum reduction in performance in any chip of less than 20%, a substantial improvement from a $Y_{OVERALL}$ of 60% achieved when only considering the defect-free parts.

Today's systems that provide fail-in-place capabilities do so at the system level and typically provide hot spares for power supplies, processors chips, memory modules, and disks [2]. We advocate pushing fail-in-place inside the boundaries of a single chip or processor and allowing defective components to continue to operate, perhaps with somewhat degraded performance. Of course fail-in-place also requires techniques for detection and recovery from intermittent and transient failures that occur during a program's execution, and some such mechanisms are summarized in the literature [1, 15].

The regularity and redundancy that we exploit is synergistic with several technology and design trends. Managing increasing design complexity demands modular design techniques that reuse chip components, thus creating redundancy opportunities. Second, the increase in wire delay relative to transistor switching time will likely lead to partitioned architectures composed of replicated hardware modules [16]. Finally, looming limits on energy and heat have led architects to suggest trading power for performance by selectively disabling microarchitecture components [7]. We expect that future systems designers will take advantage of replication and partitioning to meet these joint goals of power, performance, reliability, and ease of design.

References

[1] T. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. *International Symposium on Microarchitecture*, pages 196–207, November 1999.

[2] D. C. Bossen, A. Kitamorn, K. F. Reick, and M. S. Floyd. Fault-tolerant design of the IBM pseries 690 system using Power4 processor technology. *IBM Journal of Research and Development*, 46(1):77, January 2002.

[3] R. Desikan, D. Burger, and S. W. Keckler. Measuring experimental error in microprocessor simulation. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 266–277, July 2001.

[4] J. A. Farrell and T. C. Fischer. Issue logic for a 600MHz Out of Order execution microprocessor. *IEEE Journal of Solid-State Circuits*, 33(5), 1998.

[5] S. Gupta, S. Keckler, and D. Burger. Technology independent area and delay estimations for microprocessor building blocks. Technical Report TR-00-05, Department of Computer Sciences, The University of Texas at Austin, Austin, TX, Feb. 2001.

[6] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski. Logic bist for large industrial designs: Real issues and case studies. In *International Test Conference (ITC)*, pages 358–367, January 1999.

[7] A. Iyer and D. Marculescu. Microarchitectural level power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(3):230–239, June 2002.

[8] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Patnaik, and J. Torellas. FlexRAM: Towards an Advanced Intelligent Memory System. *International Conference on Computer Design*, October 1999.

[9] J. Keller. The 21264: A Superscalar Alpha Processor with Out-of-Order Execution. Microprocessor Forum presentation, October 1996.

[10] I. Koren and Z. Koren. Defect tolerant VLSI circuits: Techniques and yield analysis. In *Proceedings of the IEEE*, volume 86, pages 1817–1836, September 1998.

[11] K. Krewell. Marketing PC Performance. Microprocessor Report, November 2001.

[12] X. Li, A. J. Strojwas, and M. F. Antonelli. Holistic Yield Improvement Methodology. *Semiconductor Fabtech Journal*, 8(7):257–265, July 1998.

[13] W. Maly and J. Deszczka. Yield estimation model for VLSI artwork evaluation. In *Electronic Letters*, volume 19, pages 226–227, March 1983.

[14] J. F. Meyer. On evaluating the performability of degradable computer systems. In *Proceedings of the IEEE*, volume 29, pages 720–731, August 1980.

[15] S. K. Reinhardt and S. Mukherjee. Transient Fault Detection via Simultaneous Multithreading. In *International Symposium on Computer Architecture*, pages 25–36, July 2000.

[16] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. Keckler, and C. Moore. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 422–433, June 2003.

[17] The International Technology Roadmap for Semiconductors. Semiconductor Industry Association, 2001.

[18] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002.

[19] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An integrated cache timing, power and area model. Technical report, Compaq Computer Corporation, August 2001.

[20] C. H. Stapper, A. N. McLaren, and M. Dreckmann. Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product. *IBM Journal of Research and Development*, 24:398–409, May 1980.