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
IBM Technical Contacts : John Keaty, Rob Bell, and Mootaz Elnozahy

Abstract

Continued advancements in fabrication technology and reductions in feature size create 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 microarchitectural redundancy then increases to 99.6%.*

1 Introduction

Since the advent of the microprocessor in 1971 [5], technology features sizes have been reduced from $10\mu m$ to $0.13\mu m$ and transistor counts have soared from 2,300 to over 100 million. While lithography trends suggest chips 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 makes integrated circuits more susceptible to manufacturing defects that tend to reduce yield. The Semiconductor Industry Association [20] has set a target of 75% (75 good chips for every 100 manufactured) for overall microprocessor yield. While manufacturing engineers have made substantial innovations in materials and clean rooms to achieve this target at current technologies, realizing this target in future processes may prove extremely costly.

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. Because the cost of replacing faulty computer system components in a large system can be high, many vendors have developed fail-in-place systems requiring hot spares [23].

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 *correct* 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. We further explore different granularities of degraded components, from processor subcomponents to whole processors, for uniprocessor chips and future architectures based on chip-multiprocessors [25]. To compute the chip yield we use a microprocessor component area model in combination with a basic yield model. Since there can be a performance penalty depending on the degraded configuration, we use a microprocessor simulator to measure the chip end-performance across the range of different configurations. Based on this analysis we extend *performance binning* to include a new method of differentiating chips that come off the fabrication line. Chips of different end-performance can be marketed at different prices, extending the current manufacturers use of speed binning, in which chips that run at different peak clock rates are sold at different prices. 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. We extend this analy-

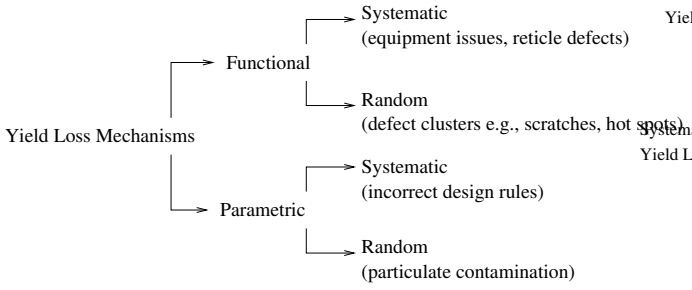


Figure 1. Yield Loss Components.

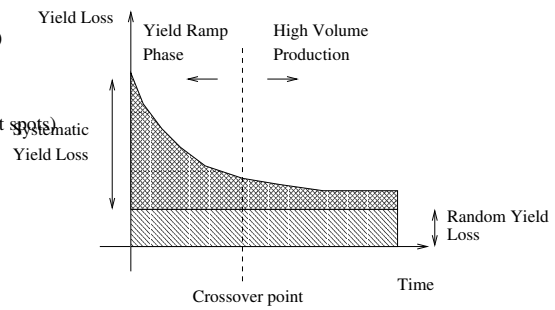


Figure 2. Yield VS time curve of a typical product.

sis to chip-multiprocessor architectures at future technologies, where we show that microarchitectural redundancy provides greater benefits in improving Y_{PAV}. These results show that many random manufacturing defects can be tolerated by exploiting microarchitectural redundancy. They also suggest that augmenting fail-in-place systems with chip-level diagnostics and reconfiguration could provide more graceful degradation when components become faulty. The remainder of this paper is organized as follows. Section 2 provides background on the trends in system design and manufacturing, such as yield, speed binning, and built-in self-test (BIST) that motivate our work. 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 major technology and design trends.

2 Background

Understanding yield loss is a critical activity in semiconductor device manufacturing. The overall yield is influenced by many factors, including the maturity of the fabrication process, the ability of a particular design to tolerate defects, and the ability to identify usable parts from unusable ones.

2.1 Sources of Yield Loss

Figure 1 provides a simple breakdown of some of the common yield loss categories. Functional yield loss occurs when a device fails to meet the intended functionality from a logical point of view. Parametric yield loss occurs when otherwise functional devices fail to fall within the allowed range of acceptable electrical characteristics. Both types of yield loss can be caused by either *systematic* defects or *random* defects. Systematic defects result from problems in the manufacturing process such as contamination of materials or imprecise calibration of the equipment. Random defects, on the other hand, are the result of inevitable particle impurities in the air and are much more difficult to overcome. The effects of these defects over time are illustrated in Figure 2 which shows an initial phase of technology deployment dominated by systematic failures, with an eventual crossover to a more mature phase dominated by the random defects.

2.2 Design for Yield

The ground rules define the basic allowable structures in a particular technology. There are several ways in which they can be modified to incorporate additional *design for yield* guidelines that minimize the effects of common yield detractors. For example, at the circuit level, guidelines that discourage the use of tall stacked transistor structures and noise sensitive structures can help reduce parametric yield losses. In layout, filling otherwise unused tracks with metal to establish a regular pattern can enable better equipment calibration during manufacturing. For high volume manufacturing, feedback from yield analysis and iteration of the design to address these detractors is an important aspect of yield management. A good overview of the different defect tolerance techniques used in VLSI circuits is provided in [12].

In designs with a high degree of regularity, such as DRAMs and SRAMs, it is common to make use of explicit redundancy to help improve yield. By including a small number of redundant rows and columns in the structure of the RAM, along with steering logic in the decoders, the yield losses that would normally result from a small number of random defects in the main array can be minimized. We

observe that many structures in modern chip design, beyond just DRAM and SRAM, contain some degree of regularity. In this paper we propose to extend the *design for yield* techniques in the microarchitectural level by specifically identifying the redundancy in different on-chip components and the mechanisms by which it can be exploited.

2.3 Product Binning

Some degree of process variation across wafers and wafer lots is a natural and accepted aspect of semiconductor fabrication. As a result of these process variations, some chips can operate at faster clock frequencies than others. Today, it has become common for manufacturers to separate these parts into *speed bins* and to offer them for sale at different prices. This practice enables a broader spectrum of potential applications for the product, and effectively increases the productive yield as well.

Recently, and at various times in the past, some manufacturers 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 [13]. Although the current practice is motivated, at least in part, by marketing tactics, it opens up the option for improving yield based on partially good chips that offer a range of performance. 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. The resulting product would offer somewhat less performance, but when binned appropriately, offers overall enhanced product yield.

2.4 Enhanced BIST

As the number of transistors on a chip increases, so does the complexity and volume of the test patterns required to identify and diagnose faults. Driven by the enormous potential expense of suitable testers and the associated test time, many chips today are augmented with built-in self test (BIST) functions to partially relieve the testing burden. Although BIST does not eliminate the need for traditional test patterns, it is used extensively for testing on-chip RAM structures, for stressing the design during burn-in, and for speed binning.

The yield improvement schemes proposed in this paper rely on the ability to specifically identify where faults have been detected, and whenever possible, to circumvent the problem by disabling or reconfiguring the faulty resource. As a result, 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. The challenges here will be to define the appropriate granularity for the BIST domains, and to develop automatic pattern generators for isolating faults in structures that contain more logic circuitry than basic RAMs.

By exploiting the natural structural regularity and replication apparent in many chips today, along with enhanced BIST controllers, we expect to reduce both the systematic and random yield losses. In doing so, the technique offers the possibility of identifying more usable chips during the critical technology learning phase, and increasing the ultimate yield during the full production phase. In this paper, we focus primarily on the yield loss due to random defects, but recognize that many of the techniques discussed can also help improve yield losses due to some types of systematic defects.

3 On-Chip Redundancy Model

In this paper, we identify three primary types of redundancy as a basis for our redundancy model (Figure 3). Component level redundancy (CLR) involves multiple exact copies of some component in the microarchitecture. With Array Redundancy (AR), array structures and the associated decoders are expanded to include redundant rows and columns of cells. Dynamic Queue Redundancy (DQR) recognizes that the use of dynamically allocated queues involves a protocol that must be prepared for the possibility that a particular queue is currently full, and new entries must be held off until a spot opens up. Each redundancy model is explained in detail below.

3.1 Basic Redundancy Types

In CLR, the component is typically replicated to provide additional performance through parallelism, but only a subset is actually required for correct functionality. In essence, each component that has CLR has a *resource* line associated with it. The component's BIST module sets the *resource* line to be permanently BUSY in the event that the component is disabled due to internal faults. The parent control logic is then augmented to use only those components whose resource lines are FREE. The intra and inter-cluster issue logic and other components that use CLR can use this mechanism to avoid defective units during execution. The Alpha 21264 [11, 7], for example, has two integer clusters each with two integer arithmetic units. The CLR concept can be applied within a cluster since it can operate correctly with only one functional unit. The clusters are not exact copies of each other in this particular design, and the instruction dispatch logic is statically biased to distribute instructions between them. But it is possible to formulate a similar design where they are symmetric, and then the CLR model could be applied here as well. In the future, as designers continue to struggle with managing the complexity

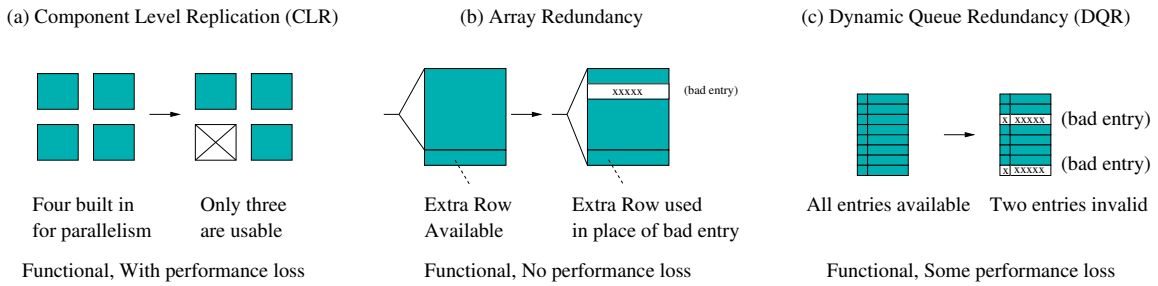


Figure 3. Basic Redundancy Models.

of large designs, we expect this sort of *design for reuse* to become more common.

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. This technique is already commonly used in many types of RAM chips as well as in the embedded RAM structures found in more general purpose chips such as microprocessors. From a yield perspective, the attractive thing about AR is that a relatively small investment in area can offer excellent defect tolerance for the entire structure. In many cases, this can drive the yield loss due to these structures to very low levels with no loss in performance.

DQR is the third type of redundancy we explore. Basically, 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, and the existing protocols can be leveraged to stall the machine when the *available* queue entries are full. Downstream logic simply needs to understand that queue entries marked invalid should never be processed. In highly pipelined designs, as well as designs that support dynamic reordering of operations, these types of queues are common. For example, in the Alpha 21264, the reorder buffers, the issue window, the register remappers, the load buffers, and the store buffers are all implemented as queues. There is a minimum size to these queues to support basic functionality, and the larger sizes are intentionally used to gain performance. Nevertheless, our experiments show that disabling one or two entries in most of these queues results in at most 1% loss in performance. Alternatively, a design might include spare queue entries in these structures to allow some defect tolerance without losing any performance, similar to AR.

In some cases, it is possible for a single structure to offer more than one type of redundancy. For example, a set associative cache might be structured to allow both CLR (disabling one set) and AR (redundant row steering in one of the other sets). Similarly, in the future, it is likely that many chips will contain multiple processors. In this case, we can imagine a set of *intra-processor* redundancies as well as *inter-processor* redundancy at the next level of hierarchy. As designs strive for more outstanding operations in flight, it is likely that potential for redundancy will increase over time.

3.2 Proposed Microarchitecture

As a basis for analysing the effects of the different redundancy types, we have defined a processor model that is similar to the Alpha 21264 [11, 7]. Both the integer and floating point clusters are symmetric and each have 2 functional units within them. The processor model also has an on-chip L2 cache of 1MB. Table 1 shows more detail on the specific parameters of our processor model, along with the redundancy model that we have adopted for each of the on-chip components. The *base capacities* characterize the processor configuration at which it achieves maximum performance. The *spare entries* provided in the components are used only in the face of defects and do not contribute to additional performance. The execution clusters and the internal ALUs are covered with the CLR model. The hierarchial nature of the redundancy for the clusters and ALUs provides coverage over the control logic of these components. The L1 caches and L2 cache are covered by both the Array Redundancy model and, since they are set associative, the CLR model. Consistent with the common industry practice [10], the redundant rows and columns are about 2.5% of the base cache capacity. The bank level redundancy for the caches provides coverage over the peripheral logic of these structures also. We also allow the configuration with no on-chip L2 cache, *ie.*, with both L2 cache banks defective. The TLBs are covered by the AR model. All of the instruction window queues, register files, map tables, reorder buffers, and storage queues are associated with the DQR model, implemented by providing spare entries and thereby exhibit no loss in performance. Elements of the processor not listed in the table, including random control logic, have no redundancy coverage in our example design. Nevertheless, the parts of the design that have coverage through redundancy

Processor Redundancy Configuration			
Resource	Base capacity / Spare entries	Redundancy Model	Minimum operational size
Integer Instruction Window	20 / 1	DQR	20
Floating point Instruction Window	20 / 1	DQR	20
Integer Register File	80 / 2	DQR	80
Floating point Register File	72 / 2	DQR	72
Integer Map Table	32 / 1	DQR	32
FP Map Table	32 / 1	DQR	32
Int Alus per cluster	2 / 0	CLR	1
Int Mult per cluster	2 / 0	CLR	1
FP Alus per cluster	2 / 0	CLR	1
FP Mult per cluster	2 / 0	CLR	1
Integer clusters	2 / 0	CLR	1
Floating point clusters	2 / 0	CLR	1
Reorder Buffer	80 / 2	DQR	80
Load queue	32 / 1	DQR	32
Store queue	32 / 1	DQR	32
ITLB (Fully associative)	128 / 2	AR	128
DTLB (Fully associative)	128 / 2	AR	128
L1 I cache (2-way associative)	64KB / 1.5KB	AR, CLR	32KB
L1 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

constitute approximately 85% of the total area of the processor. This configuration and aggregate model is used for the 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. To model the effects for larger chips in future technologies, this same basic configuration is used with a variety of chip mulitprocessor (CMP) topologies.

4 Methodology

Our methodology for calculating chip yield integrates a basic yield model and a microprocessor area model with the redundancy model of the chip components. The *basic yield* model is a simple probability distribution that calculates the random yield of a given area of silicon. The area of the chip components themselves are estimated using the *microprocessor area* model. The *yield with redundancy* model breaks a component into its redundant and non-redundant pieces before computing the component yield. 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 metric of performance in this paper is throughput (measured in instructions per clock or IPC), and is computed using a microprocessor simulator. The remainder of this section describes each of these models in greater detail.

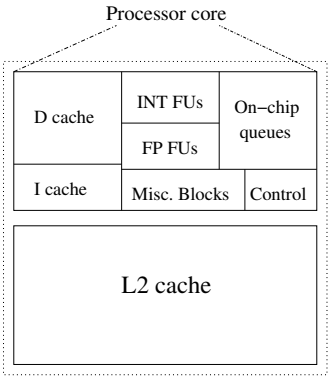
4.1 Random Defect Limited Yield Model

Yield loss is a function of the size, material, location and the process step in which a defect is introduced. In this paper we have adopted the Poisson Yield model [14] for modeling the random yield component, because it allows us to simplify the mathematical treatment and focus more on the interaction between the redundancy models and the resulting yield trends. Our methodology can be extended to use other commonly studied yield models such as the Negative Binomial model.

The Poisson Yield (Y_P) Model [14] assumes 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 which is the fraction of the total component area that is sensitive to defects. The kill ratio is a function of the defect type, defect size and the feature size, and increases as the ratio of defect size to the feature size increases. For instance, a larger defect can more easily lead to a bridging fault between two metal tracks. The Poisson Yield model equation exhibits an exponential dependence of die yield on component area, defect density, and the kill ratio. Therefore, manufacturers can improve yield either by reducing the chip area or by reducing the defect density; yield tends to get worse with smaller feature sizes because



Structure	Percentage of total area
L2 cache	49.05%
L1 D cache	12.68%
L1 I cache	5.45%
Integer functional units	6.28%
Floating point functional units	6.72%
On-chip storage structures (Register files, Tlbs, rename table, Issue queues, Reorder buffer, Load/Store queue)	11.06%
Miscellaneous components (BIU, PLL, I/O pads, Wiring channels)	6.01%
Random control logic	2.77%

Figure 4. The Uniprocessor Model

the kill ratio increases. The ITRS [20] 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 $320mm^2$. This result validates 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. To model the area of the on-chip L1 and L2 caches we used Cacti 3.0 [22] which accounts for the cache capacity, subbank organization, line size, associativity, number of ports and the technology generation. Cacti also provides the cache aspect ratio and area efficiency[1] which can be used to determine the fraction of area occupied by the peripheral logic and the data. Since the design style of most other on-chip storage structures like the register files, instruction window are similar to caches, Cacti can be appropriately configured to derive a reasonable estimate of the area, aspect ratio and efficiency of these structures also. To model the area of functional units we used an empirically derived, technology-independent area model [8]. We used a method of simple manual layouts to estimate the area of random control logic components including the select, rename and instruction wakeup logic, based on the logic level block diagrams that are available in the literature [16]. To estimate 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 [7, 11].

We validated our area model against the Alpha 21264 microprocessor whose detailed floorplan statistics are available in [7, 11]. Alpha 21264 area from the floorplan is $314mm^2$ and the estimated area using the area model is $302mm^2$, which is an error of 3.8%. Figure 4 illustrates the uniprocessor model described in Section 3, with the L2 cache coupled to the processor core. The total area of the chip was calculated using the area model to be $325mm^2$ at 250nm. Figure 4 also shows the distribution of the chip area among its most significant components.

[1]The area efficiency of a memory structure can be defined as the ratio of the area of the memory cells to the total area of the cache.

4.3 Yield with Redundancy Model

To compute the yield of a component requires breaking it into its redundant and non-redundant pieces. For example, Array Redundancy in memory structures provides coverage over defects that occur in the area occupied by the data cells, but a defect in the decode logic would still be fatal. The ability to distinguish between the regions included in the redundancy model and those that are still vulnerable to defects is fundamental to calculating the component yield.

Yield with partial redundancy: Since defects in the Poisson Yield model are considered to be completely independent, the yield of a component that has partial redundancy can be described by the equation:

$$Y = Y_{PNR} \times Y_R \quad (2)$$

where Y_{PNR} represents the yield for the area of the component that has no redundancy, and Y_R is the yield of the region that is covered by the redundancy model. The Poisson Yield model alone can be directly used to compute the yield of the area without redundancy. For storage components that use either the AR or the DQR models we use the efficiency of the SRAM array to evaluate the fraction of area devoted to the peripheral logic and the data. On the other hand, on-chip structures with CLR have their entire area covered by the redundancy model. The yield of the area

with redundancy is also computed using the Poisson Yield model but is a function of the redundancy model employed.

Yield with the basic redundancy models: The three redundancy models we described in Section 3 offer coverage over the building blocks of on-chip components. A component building block is either an entry in a storage structure or queue, or a unit such as a cluster or a functional unit, which we will generally refer to here as a *primary redundant circuit* (PRC). The total capacity of a component is then the sum of its base number of $PRCs$ and the number of spare $PRCs$.

A redundancy model specifies the minimum number of working $PRCs$ the component must possess to ensure correct overall functionality. The component yield is then simply the probabilty that it has at least its minimum subset of $PRCs$ working out of the total number of $PRCs$ including the spares. A particular configuration of the component is specified by its number of working $PRCs$. A specified configuration can be achieved in multiple ways depending on exactly which of its $PRCs$ are working, and is calculated using the *combinations* (C_r^n) operator. Both the number of working $PRCs$ and the number of ways in which the configuration can be met decide the probability of a component being in that configuration. The overall component yield is therefore the sum of the probabilties of all the configurations in which the component has at least the minimum number of $PRCs$ working. The Y_R from this calculation is summarized using the well known binomial expansion:

$$Y_R = \sum_{i=mins}^{bs+se} C_{mins}^{(bs+se)} \times \Pr\, PRC\ W^i \times \Pr\, PRC\ D^{bs+se-i} \tag{3}$$

where PRC W is a working PRC, PRC D is a defective PRC, $mins$ is the subset of $PRCs$ required for correct functionality, bs represents the base number of $PRCs$ in the component, and se is the number of spare $PRCs$. The probability of a PRC 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. Components with DQR also have very similar specifications. On the other hand, choosing to have CLR in the clusters for example, introduces the possibility of having configurations with only one cluster functional. Hence in this case se is equal to zero and $mins$ is strictly lesser than bs, with its exact value determined by the specific component. The minimum subset of $PRCs$ for each on-chip component determined by the specific redundancy model is given in Table 1.

4.4 Performance Averaged Yield Model

To have a fair comparison between the processors that are fully functional and those with IPC degradation we must account for the performance while calculating yield. The processor model described in Section 3.2 defined the peak performance of the processor as that achieved when all of its components contain their base capacity. The overall yield described above includes processors that exhibit a range of different end-performance results, which we will refer to as $Y_{OVERALL}$. We have formulated the Y_{PAV} metric so that it captures both the effects of redundancy—improvements in yield and reductions from peak performance. Y_{PAV} is calculated by associating with the yield of each possible processor configuration (Y_i), a score equal to its performance relative to the maximum, and is given by:

$$Y_{PAV} = \sum_{i\ =\ all\ configurations} Y_i \times \frac{IPC_i}{MAXIPC}$$

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 then modified the simulator to support symmetric clusters. We further made modifications that enable us to simulate the different degraded configurations by selectively disabling on-chip components. Figure 5 shows the benchmarks we used in our experiments. We chose seven benchmarks from the SPEC2000 benchmark suite and *sphinx* a speech recognition benchmark to provide a wide range of behavior. The applications *mesa, equake, eon,* and *gzip* show relatively high IPC and are more sensitive to the available execution resources. The applications *sphinx, mcf, swim* and *art* are memory intensive in nature and show greater sensitivity to cache capacities. For each benchmark we simulated the sequence of instructions which capture the representative phase of the program, determined by using SimPoint [21]. Figure 5 also shows the number of instructions skipped to reach the start of the execution phase ($FFWD$), the number of instructions simulated (RUN), and the maximum IPC for each benchmark at the base configuration.

Figure 6 plots the normalized IPC distribution for the range of allowed configurations. The graph shows that most of the configurations fall into the region that is within 20% of MaxIPC. The set of bars around 55% MaxIPC correspond to the configurations with no on-chip L2 cache. To give an idea of the sensitivity of IPC to the capacities of different components, Table 2 shows the normalized harmonic mean IPC values for a subset of all the chip configurations allowed by our methodology. The top portion of the

	Benchmark category	Benchmark name	FFWD (x100M)	RUN	MaxIPC
INT	Memory Intensive	181.mcf	336.3	100M	0.1330
		sphinx	60	200M	0.5707
	Processor bound	164.gzip	332	100M	1.7588
		252.eon	207.3	100M	1.2909
FP	Memory Intensive	171.swim	1196	100M	1.0245
		179.art	66.3	100M	0.2616
	Processor bound	183.equake	193.4	100M	1.1109
		177.mesa	639.9	100M	1.3398

Figure 5. Benchmarks used for performance experiments

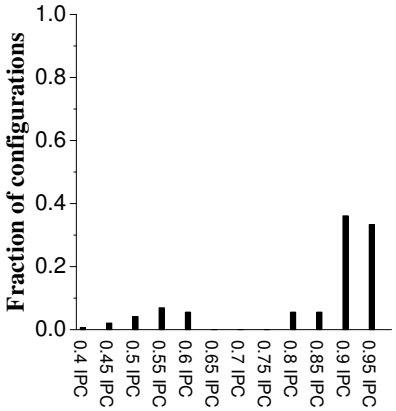


Figure 6. IPC distribution for the different configurations.

table contains the configurations for which the maximum reduction in IPC is less than 20% across all benchmark categories. In the bottom portion there is at least one category for which the relative IPC drops below 80%. When we combine this IPC binning with the yield model, our results indicate that the number of chips with less than 80% MaxIPC rapidly approaches zero.

5 Results

This section presents our results for the yield and defect density projections at future technologies and chip microarchitectures. We begin by describing the three different chip architectures that are investigated in this paper. We proceeed to examine the yield trends for a uniprocessor with and without redundancy. We then analyse the benefits of microarchitectural redundancy for chip-multiprocessor architectures, and finally end with a comparison of the yield improvements attained using different redundancy models.

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, to achieve their target performance. If the desired functionality remains fairly constant with time, as in the case of special purpose processors, the required performance can be achieved with no addtional features in the processor architecture. The constant-architecture scheme is shown in Figure 7a, where the capacities of the microarchitectural structures are kept constant across all technologies. Because the capacities are kept constant, the area of the uniprocessor decreases rapidly with decreasing feature size. However, trends in general purpose processor design have demonstrated that die sizes do not shrink with successive microprocessor generations. Instead microarchitectural features have been added that enhance the processor's functionality and consume the silicon area. Figure 7b illustrates the constant-area uniprocessor model, where the chip area is kept constant across all technologies and hence the sizes of the microarchitectural strucutures are allowed to increase. The relative proportions of the core area and the area occupied by caches are kept approximately constant.

Technology scaling trends and considerations on multithread performance have influenced some emerging architectures to include multiple processors within a single chip, which has substantial implications for yield. Figure 7c illustrates the CMP model built using the constant-architecture uniprocessor model as the building block. The number of processors that can be accomodated per chip increases from 1 at 250nm to 24 at 50nm, given that the fraction of chip area consumed by L2 caches is kept approximately at 49% across all technologies.

5.2 Uniprocessor Yield

Technology advancements in the future are expected to involve introduction of new process steps and materials, thus potentially increasing the yield sensitivity to particles and the defect density. Figure 8 plots Y_{BASE} of the uniprocessor chip assuming that the defect densities are kept constant at their value at 250nm, with substantial investments in process control mechanisms. The Y_{BASE} for a uniprocessor with constant area decreases from 85% at 250nm to 60% at 50nm. As the area of the uniprocessor increases the area of the components without any redundancy also increases. This factor is compounded by the growing kill ratio at smaller technologies to result in considerable yield

Configurations					Benchmark Category				
IntFus	FpFus	IL1 (KB)	DL1 (KB)	L2 (MB)	Integer Memory Intensive	Integer Memory Non-Intensive	FP Memory Intensive	FP Memory Non-Intensive	All Benchmarks
4	4	64	64	1	1.0	1.0	1.0	1.0	1.0
3	4	64	64	1	0.998	0.97	0.997	0.94	0.975
2	4	64	64	1	0.986	0.874	0.999	0.884	0.932
4	3	64	64	1	0.999	0.987	0.997	0.95	0.983
4	2	64	64	1	0.999	0.999	0.997	0.95	0.986
4	4	32	64	1	0.999	0.904	0.997	0.999	0.973
4	4	64	32	1	0.995	0.956	0.995	0.937	0.97
4	4	64	64	0.5	0.94	0.999	0.872	0.949	0.938
2	2	64	64	1	0.986	0.868	0.999	0.882	0.93
2	2	32	32	0.5	0.928	0.80	0.877	0.80	0.848
1	1	64	64	1	0.935	0.56	0.961	0.625	0.728
1	1	32	32	0.5	0.885	0.54	0.848	0.609	0.689
4	4	64	64	0	0.492	0.618	0.76	0.8737	0.654
2	2	32	32	0	0.474	0.369	0.749	0.555	0.503
1	1	32	32	0	0.432	0.319	0.702	0.450	0.439

Table 2. Relative IPCs for the different classes of benchmarks

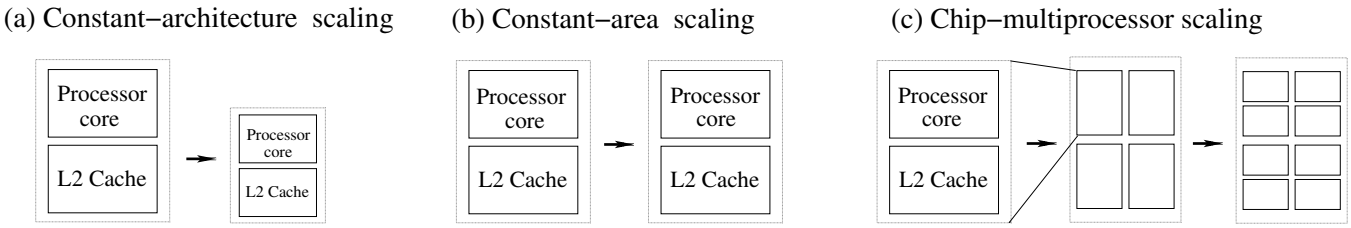


Figure 7. Chip topologies.

loss. On the other hand, Y_{BASE} for a uniprocessor with constant architecure increases from 85% at 250nm to 93% at 50nm. This increase in yield is because the gain from the rapidly decreasing chip area outweighs the increased susceptibility to yield loss due to the higher kill ratio. For both uniprocessor models, Y_{BASE} decreases with larger defect size because of the higher kill ratio.

Figure 9 shows how defect densities must scale with technology to achieve the target 83% yield. While aggressive reductions in defect densities are required in the constant-area uniprocessor model, larger defect densities can be tolerated at future technologies in the constant-architecture model. The reasons for this trend are logically the same as for Figure 8. Therefore, in the absence of microarchitectural redundancy, manufacturers must either invest in more aggressive mechanisms to decrease the defect sensitivity in the designs or accept lower final yields at future technologies.

Figure 10 shows $Y_{OVERALL}$ obtained by incrementally adding different flavors of on-chip redundancy to the constant-architecture uniprocessor model. At 100nm for example, the maximum contribution comes from L2 bank level redundancy, which increases $Y_{OVERALL}$ to 96% from a Y_{BASE} of 92.2%. CLR in the functional units dominates among all the other types of redundancy, which together further increase $Y_{OVERALL}$ to 99.2%. Since bins with finite number of chips have an observed performance loss of at most 20% (Section 4.4), regardless of technology, Y_{PAV} (indicated by the dotted line) is 98.8%, which is only 0.4% less than $Y_{OVERALL}$. Across technologies, Y_{BASE} increases from 85.4% to a maximum of 93.7%, 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 actual area occupied by the remaining components becomes vanishingly small at smaller feature sizes. As a result, $Y_{OVERALL}$ with all the types of redundancy increases from 98% at 250nm to 99.6% at 50nm, and Y_{PAV} is at most 0.4% below $Y_{OVERALL}$ 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

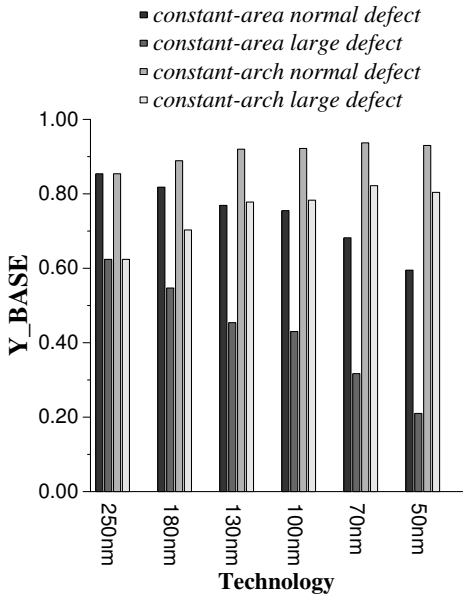


Figure 8. The effect of technology and processor model on yield.

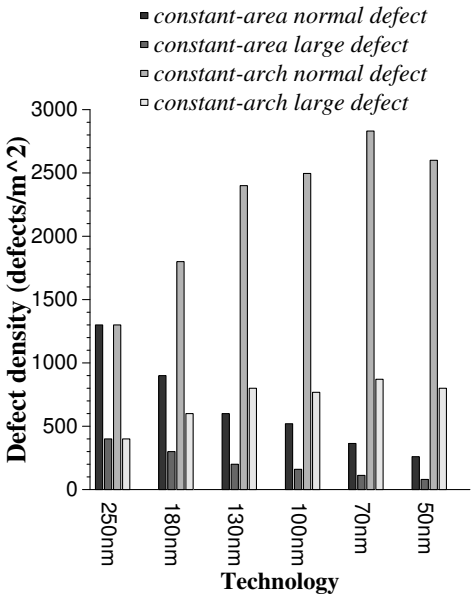


Figure 9. Target defect densities required to achieve 83% random-defect limited yield.

adding microarchitectural redundancy.

Unlike the constant-architecture model where the area of some components become insignificant at smaller feature sizes, the area of all the components remain constant with technology in the constant-area model. As a result, adding redundancy to a component will provide sustained benefits at all technologies. Hence adding microarchitectural redundancy will offer even greater improvements to Y_{PAV} in the constant-area model.

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 in a chip with inter-processor redundancy, any defect within a processor will render the whole processor bad, all the remaining processors have maximum performance and the chip can continue to function as long as there is at least one operational processor.

Regardless of the redundancy model employed, multiprocessor performance is determined by two parameters—the total number of cores (N_c), and the performance of each core (IPC_i). The performance of an individual core (IPC_i) is dependent on the application characteristics, the communication overhead among threads, and the level of degradation. Our base assumption in this study is that threads are all independent of one another. Therefore the performance of a workload can be defined as the aggregate performance of all the cores on the chip.

Extending the equation for Y_{PAV} from Section 4.4 to a multiprocessor with intra-processor redundancy:

$$Y_{PAV} = \sum_{j\ =\ all\ configurations} Y_j \times \frac{\sum_{i=1}^{N_c} IPC_{ij}}{(N_c \times MAXIPC_{core})} \tag{4}$$

where, $MAXIPC_{core}$ is the peak IPC of a fully functional processor. Since the atomicity of failure in the inter-processor redundancy model is a whole processor, a core has at most two states corresponding to $MAXIPC_{core}$ or zero IPC. The expression for Y_{PAV} of a multiprocessor with inter-processor redundancy is then given by:

$$Y_{PAV} = \sum_{j\ =\ all\ configurations} Y_j \times \frac{N_{wj}}{N_c} \tag{5}$$

where, N_w is the number of fully functional processors on the chip.

5.3.1 Yield with Intra-processor Redundancy

Figure 11 plots $Y_{OVERALL}$, across all technologies, obtained by incrementally adding redundancy to processor components in a multiprocessor with intra-processor redundancy. The xaxis shows the feature size and the number of processors per chip at each technology. At a given

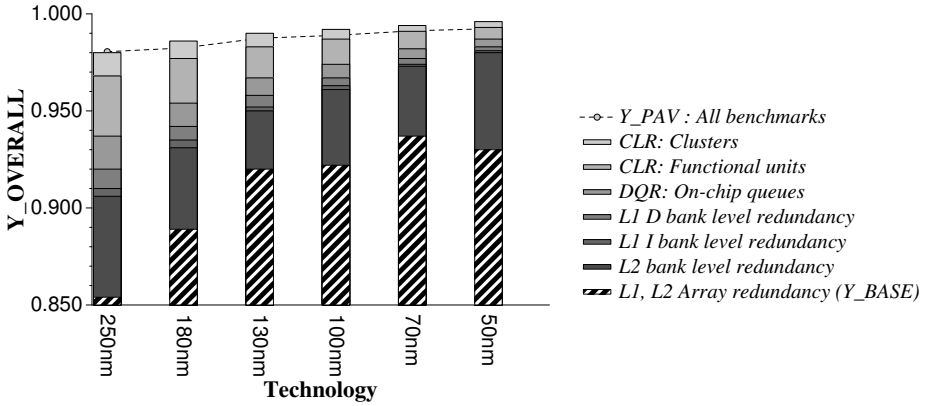


Figure 10. Yield for a constant-architecture uniprocessor model at normal defect size.

technology e.g., 70nm, adding redundancy dramatically improves $Y_{OVERALL}$ from 68.2% to 93.8%. Since functional units exhibit the maximum amount of CLR they provide the greatest improvement in yield. The benefits from L2 bank level redundancy, DQR in the queues, and CLR in the clusters are comparable across all technologies. Our experiments indicate that with all the types of redundancy, the fraction of degraded processors per chip is small. Further, the observed performance loss at the different processor configurations is also small (Table 2). As a result, Y_{PAV} (indicated by the dotted line) is within 0.1% of $Y_{OVERALL}$.

The total amount of intra-processor redundancy on the chip increases linearly with the number of processors. As a result the addition of CLR and DQR lead to greater improvements in yield at smaller feature sizes. For instance, at 180nm $Y_{OVERALL}$ with DQR in the on-chip queues is 91.1% which is improved to 95.5% by adding CLR in the functional units. At 50nm, $Y_{OVERALL}$ with DQR in the on-chip queues is 73.4% which increases to 85.8% with CLR in the functional units. The greater difference between the two $Y_{OVERALL}$ values implies that there are more degraded chips at smaller technologies. But as the area occupied by a single processor decreases, its Y_{BASE} increases (see Figure 10), 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. 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, $Y_{OVERALL}$ with all the types of redundancy drops from 98% at 250nm to 91.3% at 50nm due to higher kill ratio.

5.3.2 Yield with Inter-processor Redundancy

Figure 12 plots Y_{PAV} for a multiprocessor with inter-processor redundancy. Inter-processor redundancy gives coverage over the entire area of the chip and hence dramatically improves $Y_{OVERALL}$, approaching 100% at technologies beyond 180nm. The yield at 250nm alone is low because only one processor resides on the chip, which amounts to having no redundancy at all. Also recall that the fraction of degraded processors per chip decreases with technology. As a result, Y_{PAV} increases uniformly from 85% at 250nm to 98% at 50nm.

5.3.3 Combining Intra and Inter-processor Redundancy

Since intra and inter-processor redundancy offer different types of coverage, combining the two redundancy models provides even greater benefits in yield. Figure 13 compares Y_{PAV} obtained using four different redundancy models. 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.

A comparison of the improvements offered by intra and inter-processor redundancy models shows that, down to 100nm Y_{PAV} obtained using intra-processor redundancy is higher than from inter-processor redundancy after which we see greater benefits from the inter-processor redundancy model. The crossover point is dependent on the defect parameters and the processor granularity at each technology. While the analysis has assumed a constant defect density across technologies, larger defect densities will shift the crossover point to the right because the fault susceptibility of a given area of silicon increases, and hence fine grained redundancy becomes more appropriate. Also a more coarse-grained CMP built out of much larger uniprocessor cores will have fewer number of processors per chip. Consequently, there will be substantially more intra-processor than inter-processor redundancy, which will again shift the crossover point to the right.

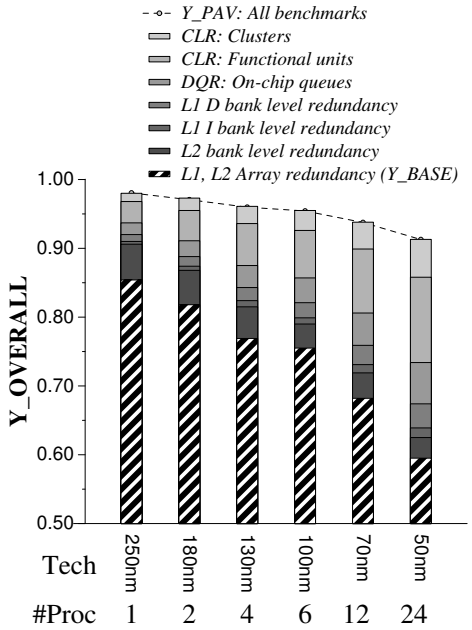


Figure 11. Yield with intra-processor redundancy at normal defect size.

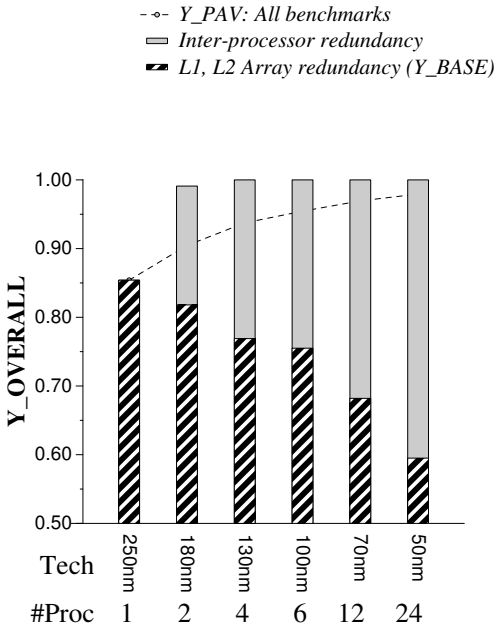


Figure 12. Yield with inter-processor redundancy at normal defect size.

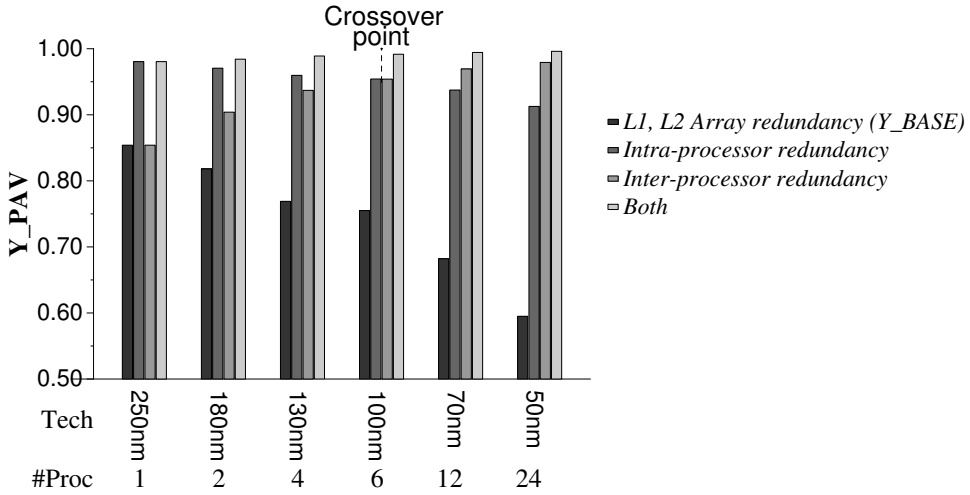


Figure 13. Comparison of Y_{PAV} for different redundancy models.

6 Conclusions

In this paper, we examine the features of modern microprocessors that can be used to enhance their reliability by exploiting regularity and redundancy within the chip microarchitecture. We also evaluate the trade-off between performance and yield through the use of redundant components within the context of a microprocessor and a chip multiprocessor. We focus on relatively coarse grain components within the microarchitecture such as execution units and cache banks, and show that mechanisms already exist within the processor control logic that easily disable the defective components from being used during program execution. By exploiting such intra-processor redundancy, we demonstrate that the total fraction of functional chips at 50nm can be increased from 60% to 91.3%, with a maximum reduction in performance in any chip of less than 20%. Inter-processor redundancy, in which some processing cores in a chip multiprocessor are allowed to be defective, has a similar effect. Based on these observations, we propose a new yield metric called *performance averaged yield* (Y_{PAV}) which accounts for the level of perfor-

mance degradation on all functioning chips. Exploiting microarchitectural redundancy can improve Y_{PAV} to as high as 99.6% at 50nm, a substantial improvement from 60% achieved when only considering the defect-free parts.

These same techniques of intra and inter-processor redundancy can be applied to fail-in-place systems. 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 [23]. 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. Defects can be detected through periodic in-system testing such as built-in self test (BIST), and defective components can be disabled dynamically, with the overall system experiencing graceful degradation. Of course fail-in-place also requires mechanisms for recovery from failures that occur during a program's execution. Hardware and software techniques that offer this capability are summarized in the literature [1, 6, 17, 18, 4].

The regularity and redundancy that we exploit in this paper is also synergistic with several technology and design trends. First, design complexity has grown dramatically as more transistors have become available to integrated circuit designers. Managing this complexity has become a tremendous challenge for both design and verification, and will demand modular design techniques that reuse hardware components within a chip, 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, whether they be processors, ALUs, or a combination of both [24, 15]. Finally, looming limits on energy and heat have led architects to suggest trading power for performance by modulating the clock frequency in different regions of a chip [19], dynamically reducing memory structure sizes, or selectively disabling microarchitecture components [9]. 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] E. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, September 2002.

[5] F. Faggin, M. E. Hoff, S. Mazor, and M. Shima. The history of the 4004. *IEEE Micro*, 6(16):10–20, December 1996.

[6] J. Gaisler. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture. In *International Conference on Dependable Systems and Networks*, pages 409–415, june 2001.

[7] B. A. Gieseke, R. L. Allmon, D. W. Bailey, J. Bradley, S. M. Briton, J. D. Clouser, H. R. Fair, J. A. Farrell, M. K. Gowan, C. L. Houghton, J. B. Keller, T. H. Lee, D. L. Leibholz, S. C. Lowell, M. D. Matson, R. J. Matthew, V. Peng, M. D. Quinn, D. A. Priore, M. J. Smith, and K. E. Wilcox. A 600MHz superscalar RISC microprocessor with out-of-order execution. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 176–178, February 1997.

[8] 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.

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

[10] 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.

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

[12] 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.

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

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

[15] R. Nagarajan, K. Sankaralingam, D. Burger, and S. W. Keckler. A design space evaluation of grid processor architectures. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pages 40–51, December 2001.

[16] S. Palacharla. *Complexity-Effective Superscalar Processors*. PhD thesis, Department of Computer Sciences, University Of Wisconsin Madison, 1998.

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

[18] E. Rotenberg. AR/SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors. In *International Symposium on Fault Tolerant Computing*, pages 84–91, 1998.

[19] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *8th International Symposium on High-Performance Computer Architecture*, pages 29–40, February 2002.

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

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

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

[23] L. Spainhower and T. A. Gregg. IBM s/390 parallel enterprise server g5 fault tolerance: A historical perspective. *IBM Journal of Research and Development*, 43(5/6):863–873, September/November 1999.

[24] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, W. L. Jae-Wook Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, March 2002.

[25] J. M. Tendler, J. S. Dodson, J. J. S. Fields, H. Le, and B. Sinharoy. Power4 system microarchitecture. *IBM Journal of Research and Development*, 26(1):5–26, January 2001.