On the opportunity to improve system yield with multi-core architectures

Yury Markovsky and John Wawrzynek
UC Berkeley
{yurym,johnw} @ cs.berkeley.edu

Keywords: yield prediction, redundancy, fault tolerance, multi-core

Abstract

With increasing process variability and defects, VLSI architectures are moving toward multiple on-chip cores that offer redundancy, independence and isolation—an opportunity for a dramatic improvement in chip manufacturing yield. This work develops yield and cost models for "sparing," a strategy of architecting redundant on-chip cores, and later at deployment or run time, using only a subset of the resources that meets functional and performance specification. We show that core "sparing" reduces the total chip cost from the traditional $O(A^4)$ down to $O(A^{\frac{1}{2}})$, where A is the area of the chip, without performance degradation associated with the alternatives. This strategy is more cost-efficient than fine-grain redundancy and fault-tolerance, and it can be deployed easily in the existing OS infrastructure.

1 Introduction

Chip manufacturing yield is among the key factors affecting chip cost. Correcting poor yield requires intervention at multiple levels: devices, circuits, and architecture, but each layer has a limited impact on the final product yield. For example, semiconductor process improvements are constrained by the discrete and stochastic particle nature of wafer processing. Designers use larger than necessary transistor sizes and leave extra space between lithographic features to combat process variations and defects. However, this "over-engineering" dramatically diminishes the performance improvement expected from Moore's Law scaling in logic density, power consumption and clock rates. Designers also use circuit techniques, such as differential logic for better noise immunity, or introduce redundancy in circuits (*e.g.* DMR), to reduce the impact of defects and variations. These circuit techniques and device over-engineering create "fine-grain" redundancy. An alternative to "fortifying" circuits could be an architecture with "coarse-grain," core-level redundancy — a multi-core architecture.

Consider a multi-core architecture with the cores isolated from one another via distinct clock, voltage domains, and a latency insensitive, asynchronous inter-core communication fabric. With this architecture, each core runs at its own rate and contains faults and performance variations within. This architecture results in an inherently redundant fault and performance variation isolation model that can dramatically improve manufacturing yield. At post-manufacturing time, or even during chip power up or dynamically at run time, the chip can test and disable faulty and under-performing cores, and select a *subset* of cores to run an application. This is not a novel idea: fine-grain "sparing" is used in DRAM [17], and currently coarse-grain "sparing" is employed in Niagara [12], Cell [9] and other processors. [8] models the architecture yield with block redundancy for wafer scale integration (similar to "Module redundancy" in Section 2).

As process technology is moving toward smaller dimensions, the impact of manufacturing defects and variations increases sharply. The question of how to best utilize chip resources and introduce redundancy to improve chip yield is still open. We develop a model for a multi-core die yield and cost with core "sparing," and examine several questions:

- What are the bounds of core "sparing"?
- How does core redundancy compare to module redundancy and device over-engineering?
- With the trends in semiconductor technology, what is the *optimal* core area that minimizes the number of faulty on-chip cores and maximizes the yield?
- What is the relationship between area overhead of multi-core architectures and the yield?

The paper starts with Section 2, which creates a taxonomy of on-chip redundancy schemes and discusses their requirements, limitations, and the previous work. Section 3 reviews a traditional die yield model. Section 4 develops a new die cost model with core "sparing". To compare the impact of core sparing with the impact of

other redundancy schemes, Sections 5 and 6 develop models for device over-engineering and module redundancy. Section 7 and 8 discuss the results and conclude.

2 Resource Redundancy

As semiconductor device size shrinks, system yield and reliability become increasingly difficult to manage due to a combination of physical defects and parametric variations. Although process and manufacturing engineers are constantly finding new ways to optimize yield through improved accuracy and precision of semiconductor processing steps, the quantities of substances used approach ones to tens of atoms, and thus make it impossible to control reliably. When dealing with a stochastic processes on a very small sample size, redundancy must be used effectively to manage performance and yield. We examine two critical questions: (1) where to use redundancy and (2) how to manage it? The following classifies different redundant resource schemes and analyzes them based on cost, implementation, and the impact on performance variations and yield. Table 2 summarizes these schemes with the compact simple models developed in this work.

Device Over-engineering. To improve yield and resilience to process variation, engineers employ device and interconnect widths and inter-device spacings larger than required by design rules. Although this technique has a significant permanent impact on area and performance, it reduces variance on such parameters as transistor threshold voltage and line edge roughness [3]. Increased feature sizes reduce the critical area affected by defects, and thus can significantly improve the yield (assuming equivalent defect size distribution and die size).

This technique must be used sparingly due to its high cost. Depending on details, it has from linear to quadratic impact in die area, and up to a linear increase in delay and power consumption with respect to the "oversize" factor. Due to its static nature, decisions about chip area allocation must be made at design time and for the worst case, rendering device over-engineering less effective in future semiconductor generations when tiny physical dimensions preclude our ability to carefully control the manufacturing process.

Fine grain circuit redundancy With greater uncertainty in the manufacturing process, circuit techniques such as error-correcting codes (ECC) or double module redundancy (DMR) can detect and correct both manufacturing and transient faults. With redundant circuits, the designer can contain the problem to avoid its propagation through the system. With appropriate coding the area overhead of these techniques can be smaller than that of device over-engineering. Still, they incur a permanent cost on circuit performance, even they do not lengthen the critical path. By adding redundancy, one increases the number of "near critical" paths in the circuit, which due to process variations result in a greater chance of a longest outlier path that determines the circuit performance. However, by overclocking a circuit and relying on error correction to compensate for variations and defects, the system can sometimes optimize overall power/performance and yield [2]. As with device over-engineering, decisions about where to add redundancy and which codes to use must be made early at design time and for the worst case.

Module redundancy There are two principle differences between fine grain circuit redundancy and module redundancy. (1) On larger modules, one typically encounters resource replication (*e.g.* DMR) rather than code-based error correction because it is a more natural and general solution. Coding works well on small arithmetic and communication circuits, but it is non-trivial for complex modules. (2) The larger module size allows for more flexible run-time resource management controlled in hardware or even in software.

For example, consider the following fault management scenarios on a super-scalar processor with several redundant ALUs. Scenario 1: every instruction runs through three distinct ALUs and the majority result is used [18]. Scenario 2: only some instructions (perhaps in the "reliable" section of a program) run in this TMR mode. Scenario 3: temporal redundancy (running instructions several times through the same ALU) used on some or all instructions [1]. Scenario 4: normal mode, all instructions run once through an ALU to maximize instruction throughput and processor performance. Hardware or software can select a scenario at run time and utilize the flexibility to use the redundant modules as a fault-resilience mechanism and as a raw computational resource.

Device over-engineering, fine grain circuit and module redundancy share two common problems: (1) interdependence and (2) spatial correlation. First, since the redundant devices, circuits and modules form a part of a larger design, their faults and performance variations can propagate through the entire system. Second, since redundant components (*e.g.* ALUs) are typically located in the same part of the die, they are identically influenced by spatially correlated performance variations and defect clustering. For example, *all* redundant components might be underperforming according to specification, rendering the entire die nonfunctional. These problems combine to reduce die performance and yield. An ideal solution would maximally isolate the redundant components from each other and distribute them over the area of the chip. Although this does necessarily imply a multi-core architecture, multi-core is the most natural manageable solution.

Core-level redundancy Let us focus on processor cores, although different cores would not qualitatively affect our discussion. With Globally Asynchronous Lo-

cally Synchronous (GALS) implementation and an appropriate communication fabric (*e.g* a network-on-chip), we achieve both functional and performance isolation between the cores. With redundant cores, the system can be fault-tolerant. Two aforementioned problems are solved: (1) no interdependence, the faults and variations only affect a core but not its peers; (2) cores are distributed all over the die. They can operate independently at different performance points dictated by intra-die variations, and be less affected by defect clustering.

After manufacturing the chip is tested, and faulty cores can be disabled with fuses. Well designed inter-core communication protocol ensures that disabled cores simply appear unavailable and are not used by their neighbors. Alternatively, managing such a multi-core environment can be accomplished with an operating system. A task scheduling algorithm (*e.g.* task stealing) can automatically adopt to a system with cores with heterogeneous performance, since each core processes tasks at its own rate. The defective cores do not run at all, and since they do not steal tasks, they appear busy or unavailable. The primary impediment to such a flexible multi-core system is the communication fabric overhead—currently being addressed by many researchers [10, 13]. Early results indicate that an on-chip network can reduce power consumption up to 40% with core voltage and clock scaling at the small performance penalty of 10% on a 6x6 array of processors [21, 22]. More work is still required to obtain a definitive answer, since the reported results are not general, but tied to design choices and assumptions.

3 Traditional Chip Cost Model

This section is a short review of cost modeling for traditional single core chips. We use the yield model based on negative binomial defect distribution [14, 11], where the die yield is the probability that n defects hit the critical area A given an average defect distribution D_0:

$$p(n,A,D_0) = \frac{\Gamma(\alpha+n)}{n!\Gamma(\alpha)} \frac{(AD_0/\alpha)^n}{(1+AD_0/\alpha)^{n+\alpha}} \quad (1)$$

We are interested in *defect-free* area, $n = 0$:

$$p(0,A,D_0) = (1+AD_0/\alpha)^{-\alpha} \quad (2)$$

Our die cost model ignores test and packaging costs because they are not relevant to the discussion:

$$C_{die} = \frac{C_{wafer}}{Dies/Wafer \times Y_{die}} \quad (3)$$

Param	Description
C_{die}	die cost
C_{wafer}	processing cost for a wafer (1250)
Y_{die}	die yield
Y_0	gross wafer yield (1)
Y_{core}	core yield (a function of core area)
A_{die}	die area
D_{wafer}	wafer diameter (300mm)
D_0	avg defect density (0.0002/mm^2)
α	defect clustering factor (3)
A_o	*per core* infrastructure overhead ($1mm^2$)
A_f	total fault-free, functional die area
N_f	total number of fault-free cores

Table 1: Parameters used in the models. (Default values) are those used in the graphs.

The wafer cost is fixed and outside of our control, but the die area determines both $Dies/Wafer$ and die yield Y_{die}:

$$Dies/Wafer = \frac{\pi \times D_{wafer}}{\sqrt{2A_{die}}} \quad (4)$$

$$Y_{die} = Y_0 p(0, A_{die}, D_0) \quad (5)$$

where we assume that the critical area $A_C = A_{die}$, which gives a pessimistic yield prediction, but does not affect the trends discussed in the work. Table 1 summarizes the parameters in these and other models developed in this work. By combining A_{die} and C_{die}, we obtain a formulation of die cost:

$$C_{die} = \frac{C_{wafer}\sqrt{2A_{die}}\left(1+\frac{D_0 \times A_{die}}{\alpha}\right)^{\alpha}}{\pi \times D_{wafer} \times Y_0} \quad (6)$$

$$C_{die} \sim O\left(A_{die}^{\alpha+\frac{1}{2}}\right) \quad (7)$$

The cost grows as a high-degree polynomial function of die area. With our default case $\alpha = 3$, $C_{die} = f(A_{die}^{3.5})$.

4 "Sparing" defective cores in multi-core chip

Architectures with redundant cores can disable faulty cores and operate a subset of a die. The area overhead of such an architecture stems from independent core-level power delivery networks, and inter-core communication infrastructure that includes synchronization, arbitration for shared resources (*e.g.* buses, links), *etc*. Let us designate A_o to be *per core* overhead in this architecture. If a designer requires functional, fault-free die area A_f, how should the chip be partitioned into cores to minimize total

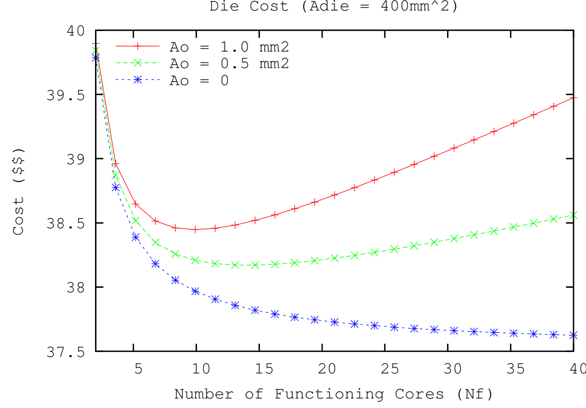


Figure 1: Cost *vs* the core count for different overheads. Notice a different optimal N_f for each overhead.

die cost? Let N_f be the number of functional equal size cores, and each core incurs a fixed area overhead A_o:

$$A_{core} = \frac{A_f}{N_f} + A_o \quad (8)$$

$$A_{die} = A_{core} \times \frac{N_f}{Y_{core}} = A_{core} \times \frac{N_f}{p(0, A_{core}, D_0)} \quad (9)$$

The total die area A_{die}, which determines chip cost, comprises $N = \frac{N_f}{Y_{core}}$ cores: N_f functional and $(N - N_f)$ defective ones. When defective cores are disabled, the chip is left with N_f functional cores with the aggregate area A_f. Substitute Equation 5 for core yield:

$$A_{die} = N_f \left(\frac{A_f}{N_f} + A_o\right) \left(1 + \frac{D_0}{\alpha} \left(\frac{A_f}{N_f} + A_o\right)\right)^{\alpha}$$

Each manufactured N-core die contains the desired N_f fault-free cores, and therefore $Y_{die} = 1$. The trade off is that core sparing scheme produces larger but perfect yielding die, but fewer of them per wafer. Let us designate Y_0 as gross yield to capture external factors such as wafer and facility yield. The total die cost is constant for a given total die area:

$$C_{die} = \frac{C_{wafer}}{Dies/Wafer \times Y_0} = \frac{C_{wafer} \times \sqrt{2A_{die}}}{\pi \times D_{wafer} \times Y_0} \quad (10)$$

$$= \frac{C_{wafer} \sqrt{2N_f \left(\frac{A_f}{N_f} + A_o\right) \left(1 + \frac{D_0}{\alpha} \left(\frac{A_f}{N_f} + A_o\right)\right)^{\alpha}}}{\pi \times D_{wafer} \times Y_0}$$

Figure 1 illustrates the equation above. With a few cores (large cores), the die cost is high due to low core yield. As the number of cores grows, the overhead of separate clock, voltage domains and interconnection fabric of many small cores begins to dominate the area and the cost. For each overhead A_o, there is a unique number of functional cores N_f that minimizes the total die area and the total die cost. To find optimal N_f, solve $\frac{\partial A_{die}}{\partial N_f} = 0$ to obtain (details are omitted for brevity):

$$N_{f,opt} = A_f \times \frac{D_0 \left(\frac{\alpha - 1}{\alpha}\right) + \sqrt{D_0^2 \left(\frac{\alpha+1}{\alpha}\right)^2 + \frac{4D_0}{A_o}}}{2\left(1 + \frac{D_0 \times A_o}{\alpha}\right)}$$

The optimal core size is:

$$A_{opt} = \frac{A_f}{N_{f,opt}} = \frac{2\left(1 + \frac{D_0 \times A_o}{\alpha}\right)}{D_0 \left(\frac{\alpha - 1}{\alpha}\right) + \sqrt{D_0^2 \left(\frac{\alpha+1}{\alpha}\right)^2 + \frac{4D_0}{A_o}}}$$

The optimal core size that minimizes total die area (both functional and faulty cores) depends only on defect density D_0, semiconductor process α and overhead A_o. Substitute $N_{f,opt} = \frac{A_f}{A_{opt}}$ into Equation 10 to obtain:

$$C_{die} = \frac{C_{wafer}}{\pi D_{wafer} Y_0} \times \sqrt{\frac{2A_f}{A_{opt}} (A_{opt} + A_o) \left(1 + \frac{D_0}{\alpha} (A_{opt} + A_o)\right)^{\alpha}} \quad (11)$$

$$C_{die} \sim O\left(\sqrt{A_f}\right) \quad (12)$$

Multi-core architectures that enable core sparing asymptotically reduce chip cost and provide flexibility for yield management at manufacturing, deployment and run time. Figure 2 illustrates the relationship between die cost C_{die} and functional area A_f for traditional monolithic unicore architecture and for a multi-core architecture with sparing. Optimal core size (A_{opt}) was used for the multi-core case. One can observe the clear difference in asymptotic behavior. Figure 3 illustrates the overhead of multi-core architecture with sparing, the ratio of functional area to the total die area ($\frac{A_f}{A_{die}}$) that includes both the spared defective cores and $A_o \times N$ of infrastructure overhead. Additionally, the figure shows that the optimal core size A_{opt} increases as expected with the infrastructure overhead A_o.

5 Yield and Over-engineering

Core "sparing" results in an asymptotic cost improvement, but how does it compare with redundancy from over-engineered semiconductor devices? An accurate general yield model that relates the transistor and interconnect sizes and the chip yield is difficult to create without circuit structure and layout details. The model below captures the trends instead. The yield depends on the defect-sensitive "critical" area A_C [4]:

$$A_C = A_{die} \int_0^{\infty} K(x) S(x) dx$$

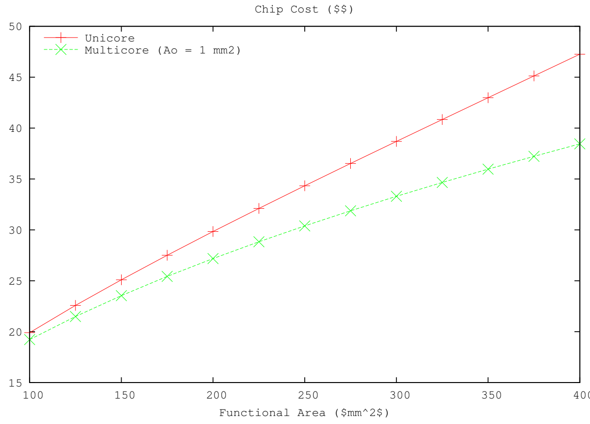


Figure 2: Die cost for the unicore *vs* multi-core die

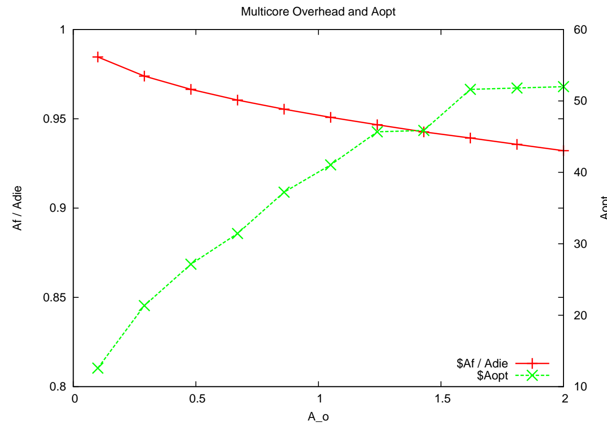


Figure 3: The ratio of functional A_f to the total die area A_{die} is mostly determined by multi-core overhead A_o.

where x is defect diameter, $K(x)$ is the fault probability kernel, and $S(x)$ is defect size distribution. Assume that defects have a circular shape. $K(x)$ and $S(x)$ depend on the process feature size as illustrated in Figure 4. The actual values of these parameters are not critical to our discussion, but the overall monotonicity is. Defect size distribution $S(x)$ can be defined as [6]:

$$S(x) = \begin{cases} 0, & \text{if } x < x_0 \\ \frac{k}{x^3}, & \text{otherwise} \end{cases}$$

where x_0 is the minimum defect size, *i.e.* defects smaller than x_0 have no impact on yield. Fault probability kernel usually has the following definition:

$$K(x) = \begin{cases} 0, & \text{if } x < \lambda_0 \\ f(x), & \text{if } \lambda_0 \leq x < w_{max} \\ 1, & \text{otherwise} \end{cases} \quad (13)$$

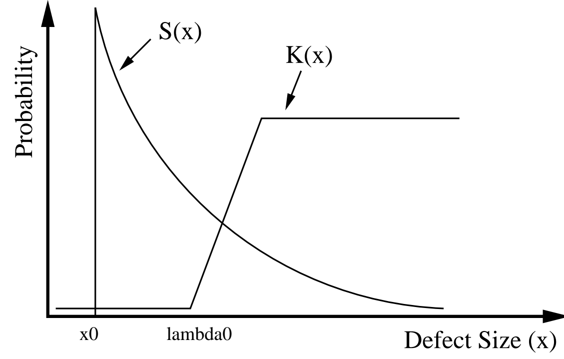


Figure 4: Illustration of $S(x)$ and $K(X)$ functions.

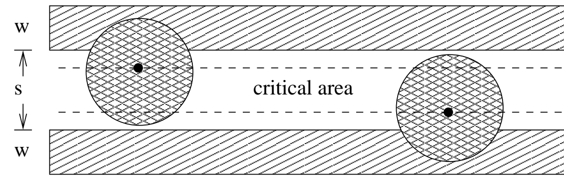


Figure 5: The critical area for a "short" fault located between the dashed lines.

where $0 \leq f(x) \leq 1$ is a monotonically increasing function that represents the probability that a defect of size x creates a fault. A defect larger than the maximum size w_{max} will result in a fault independent of its location on a die. Figure 5 illustrates a "short" defect in the interconnect structure, where all interconnect segments are parallel to one another. For this "short" fault $f(x) = x - s$ and $w_{max} = w + s$.

Consider die area A_{die} with feature size of 2λ. A crude estimate of the total device count (TDC) on the die is:

$$TDC = \frac{A_{die}}{s_0 \lambda^2}$$

where s_0 is the actual device size in terms of λ^2 squares (*e.g.* a minimum size transistor could be 6–8 λ^2). When a designer increases device feature sizes by a factor of F to improve resilience to faults and variations, the total device count drops quadratically for the same die area while expected operating frequency drops linearly [15]:

$$TDC = \frac{A_{die}}{s_0 (F\lambda)^2} \text{ and } Freq = \frac{f_0}{F}$$

How does F affect the yield? As F increases, defect size distribution $S(x)$ does not change for a given process. However, the fault probability $K(x)$ decreases because λ_0 is replaced with $F\lambda_0$ in Equation 13. Since $f(x)$ is monotonically increasing, we can approximate $K_F(x)$ with unit

step function to get the worst case fault probability:

$$K_F(x) = u(x - F\lambda_0) = \begin{cases} 0, & \text{if } x < F\lambda_0 \\ 1, & \text{otherwise} \end{cases} \quad (14)$$

The critical area A_C affected by defects:

$$\begin{aligned} A_C &= A_{die}\int_0^\infty K_F(x)S(x)dx & (15)\\ &= A_{die}\int_0^\infty u(x-F\lambda_0)S(x)dx & (16)\\ &= A_{die}\int_{F\lambda_0}^\infty \frac{k}{x^3}dx & (17)\\ &= A_{die}\frac{k}{2F^2\lambda_0^2} & (18)\end{aligned}$$

$S(x)$ is a valid probability distribution, which requires $\int_{-\infty}^{\infty} S(x)dx = 1$ and thus $k = 2x_0^2$. Assume, for example, that minimum defect size $x_0 = \frac{1}{2}\lambda_0$, to obtain:

$$A_C = A_{die}\frac{2\frac{1}{4}\lambda_0^2}{2F^2\lambda_0^2} = A_{die}\frac{1}{4F^2} \quad (19)$$

which suffices to illustrate our point that the critical area decreases as device features grow by $\times F$.

The reduced critical area, however, comes at the expense of a lower device count and lower clock frequency. To make a fair metric, instead of focusing on the die cost, we maximize the product of the total device count (TDC) and clock frequency. In effect, this maximizes the number of functional devices in a fixed die area and their speed – the total on-chip functionality (features). Let N be the total number of on-chip cores, both faulty and functional. Setting $s_0 = 1$ and $f_0 = 1$ for simplicity (they do not affect the qualitative result), we obtain:

$$\begin{aligned} A_{core} &= \frac{A_{die}}{N} - A_o & (20)\\ Y_{core} &= Y_0 p\left(0, \frac{A_{core}}{4F^2}, D_0\right) & (21)\\ TDC &= N \times Y_{core} \times \frac{A_{core}}{4F^2} & (22)\\ Freq &= \frac{1}{F} & (23)\\ \text{maximize} \quad & TDC \times Freq & (24)\end{aligned}$$

Figure 6 illustrates $TDC \times Freq$ for $A_{die} = 400mm^2$, and shows that to maximize the number of fault-free transistors running at the highest possible clock frequency, $F = 1$ and $N = 11$ for our parameters: $D_0 = 0.0002/mm^2$, $\alpha = 3$ and multi-core overhead $A_o = 1mm^2$. Although, there is no closed form solution for F to demonstrate that $F = 1.0$ (no device oversizing) always maximizes device count and frequency product, it is the case on a range of parameters analyzed. Although over-engineering devices

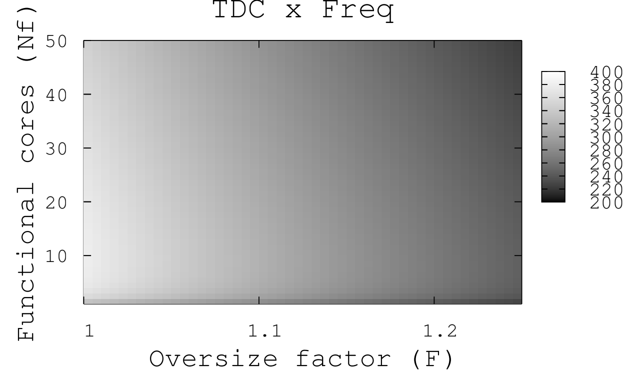


Figure 6: Device oversizing *vs* functional cores on a $A_{die} = 400mm^2$. $N = 11$, $F = 1$ maximize $TDC \times Freq$.

improves the yield for a fixed die area, the reduction in the device count and the frequency undermines the gains. "Sparing" cores is more effective to maximize chip functionality.

The story is not complete, however, since we have not considered the positive impact of device over-engineering on reducing stochastic process variations. For example, the variance on such parameters as transistor threshold voltage, decreases as feature sizes grow ($\sigma_{V_t} \sim \frac{1}{F}$). We are developing yield models based on device feature sizes and parametric variability. Together variation and defect based models would paint a complete picture and a prescriptive approach for simultaneous optimization.

6 Circuit and Module Redundancy

Although redundant circuits and modules do not offer the same resource management flexibility to the OS as the redundant cores, there are situations where redundant circuits are most effective (*e.g.* DRAM columns).

The yield model for cores with redundant resources depends on two parameters: $r \in [0,1]$ – the fraction of the area fortified by redundancy, $R \in [1,\infty)$ – area replication factor that describes the overhead. For example, if 20% of the area is consumed by ALUs, implemented with DMR, then $r = 0.2$ and $R \approx 2$ ignoring comparator overhead.

If we need functional area A, then the total area including the redundant resources is:

$$A_{total} = A(1-r) + ArR = A(1-r+rR) \quad (25)$$

Assuming that the redundant module structure tolerates up to N faults:

$$Y = p(0, A(1-r), D_0) \times \Sigma_{n=0}^{N} p(n, ArR, D_0) \quad (26)$$

The yield for the area A is the product of probabilities that A_{total} is fault-free and the sub-section ArR with "redundant" modules contains zero or one fault. This naturally

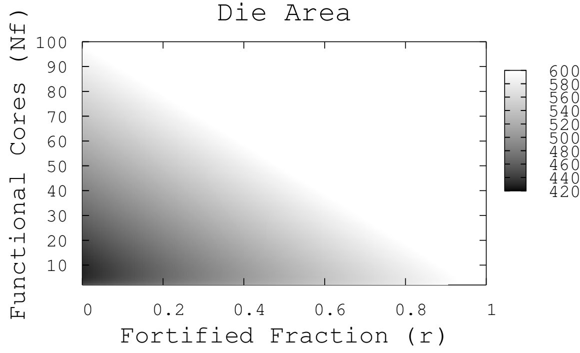
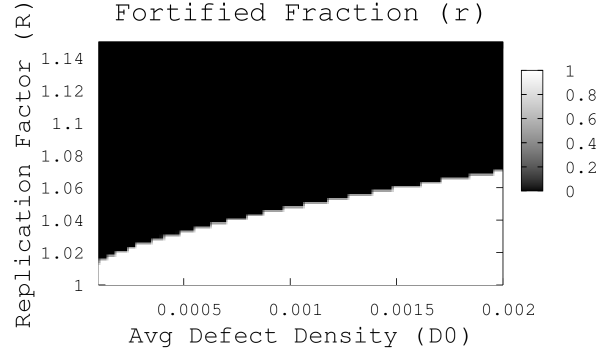
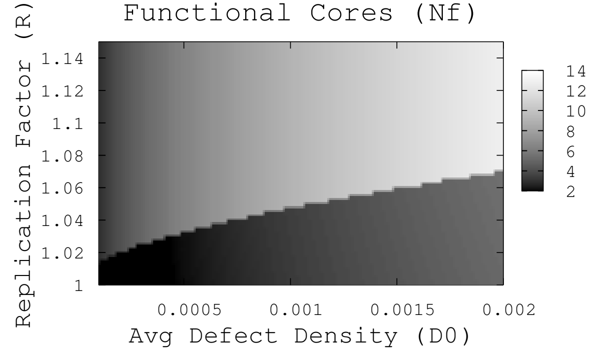


Figure 7: The total die area *vs* the fraction r of the core area that contains redundant modules, and the number of active cores N_f. ($A_f = 400, A_o = 1, R = 1.5$)



(a) Optimal r *vs* replication factor R and average defect density D_0.



(b) Optimal N_f *vs* replication factor R and average defect density D_0.

Figure 8: Optimal (r, N_f) parameters to minimize the total die area A_{die} for functional area $A_f = 400$.

extends to an arbitrary number of tolerable faults. Combining with the multi-core yield model, we obtain the total die area model. If the design requires total functional area A_f that is partitioned into N_f cores, then

$$\begin{aligned} A_{core} &= \frac{A_f}{N_f}(1 - r + rR) + A_o \\ Y_{core} &= p(0, A_{core}(1-r), D_0) \times \Sigma_{n=0}^{N} p(n, A_{core} rR, D_0) \\ A_{die} &= \frac{N_f}{Y_{core}} \times A_{core} \end{aligned}$$

Figure 7 illustrates the relationship between the total die area and the choices for r and N_f. The total die area includes the fault-free cores and the spared faulty cores. Each functional core provides A_f/N_f of fault-free area, but occupies larger area A_{core} that includes redundant modules and multi-core overhead (A_o).

Figure 8 shows that the dominant factors that determine the optimal (r, N_f) are the replication factor R and defect density D_0. There are two regions shown in the graphs: (1) low replication factor R suggests that modular redundancy should be used on the entire core ($r = 1$). (2) As replication overhead R increases, the overhead of modular redundancy becomes too expensive and core sparing is strongly favored.

7 Discussion

Table 2 summarizes the effect of different redundancy levels on yield, area and clock frequency. As Sections 4 and 6 demonstrate the core and module redundancy can minimizes total die area and the die cost. The primary advantage of this coarser grain redundancy over the fine grain device over-engineering is its minimal impact on clock frequency.

Figure 9 compares the die cost for different redundancy levels. Each curve represents the die cost under optimal parameters: core size A_{opt}, fortified fraction r, and over-engineering factor F. The x-axis shows functional area. Core sparing consistently results in the minimum die cost compared to other schemes. Based on our models, the best use of die area is a set of independent cores that can be spared. Other forms of redundancy do not provide sufficient yield boost to compensate for their area overhead.

Section 4 demonstrates that core "sparing" results in $C_{die} \sim O(\sqrt{A})$ if the optimal core area is used. The optimal core area depends only on the process parameters: defect density (D_0), process complexity α, performance variations (σ), and the area overhead (A_o). A_o is the result of independent clock and voltage domains, and the asynchronous communication fabric. Without this overhead, simply selecting the smallest possible core size would minimize the die cost, since the probability of a core being defective or "slow" would be minimal.

Even ignoring the overhead, one cannot select an arbitrarily small core, since we require a core with a minimum functionality and compute power. Modern CAD tools and compilers cannot partition or parallelize applications into arbitrary sized communicating components. However, tools cannot be blamed entirely, since many classes

	Yield	*Area*	Frequency
Device Overengineering ($\times F$)	$p\left(0, \frac{A}{F^2}, D_0\right)$	$TDC \sim F^{-2}$	$Freq \sim F^{-1}$
Circuit and module (r, R)	$p\left(0, A\left(1-r+rR\right), D_0\right) + p\left(1, ArR, D_0\right)$	$A(1-r+rR)$	small, but with variations $\sim \left(R \times \frac{\sigma}{\mu}\right)^{-1}$
Core redundancy	$p\left(0, \frac{A}{N_a} + Ao, D_0\right)$	$\frac{N_a}{Y_{core}} \times A_{core}$	negligible impact, or improvement w/ independent CLKs

Table 2: Comparison of different redundancy levels

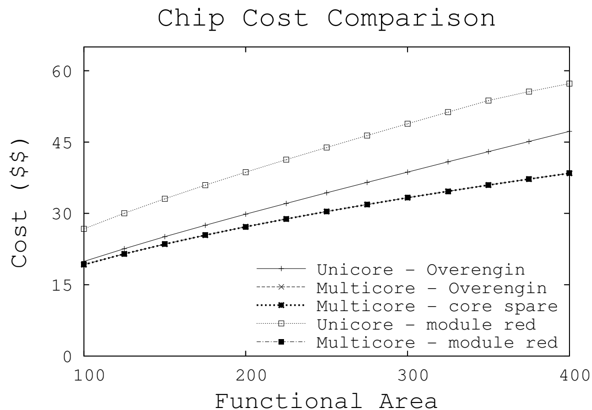


Figure 9: Die cost comparison with different redundancy schemes: x-axis shows *functional, fault-free* area. Input parameters: $A_o = 1mm^2/core$, $\alpha = 3$, $R = 2$.

of applications do not partition arbitrarily, but naturally divide into coarse-grain tasks or very fine-grain logical functions, but nothing in between. This limitation places application specific restrictions on useful processor core sizes, which in turns affects achievable die yield.

What is the typical processor core area today? Existing commercial and academic multi-core architectures show that a super-scalar processor core ranges from 27–51$G\lambda^2$, while a simple, in-order core requires 0.5–11$G\lambda^2$ of silicon area [7, 19, 20, 5, 16, 12]. The rest of the die typically contains caches, which are protected by ECC. Architects striving to optimize yield have a range of options for core area, *i.e.* they can select a core to meet their application requirements and also minimize the impact of defects and performance variations.

8 Future Directions and Conclusion

Redundancy in some form is and will increasingly be required to combat manufacturing defects and performance variations. The models presented help to answer several critical questions. Where and how to use redundancy in circuit and architecture design? Should we strive toward fewer, but reliable cores with fine grain redundancy, or make greater number of smaller but possibly faulty cores? Our simple model of device over-engineering and module redundancy suggest the latter: many unreliable, high-performance cores. However, to properly answer this question, we will investigate the relationship between delay, area, power, error correction enabled by each type of redundancy.

We are working to extend the defect-based yield models with intra-chip performance variations to allow joint architecture parameters optimization. This requires model parameters and results to be validated and calibrated with real world data.

Multi-core architectures with independent clock and supply voltage domains offer an opportunity for core sparing to optimize manufacturing chip yield. Sparing dramatically improves chip yield and reduces chip cost from the traditional $O(A^4)$ to $O(\sqrt{A})$, when optimal size cores are used.

9 Acknowledgments

The authors thank Nicholas Weaver for motivating ideas in this work, and acknowledge the support of the GSRC Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. Additional support also came from the Berkeley Wireless Research Center (BWRC).

References

[1] Todd Austin, David Blaauw, Trevor Mudge, and Krisztin Flautner. Making typical silicon matter with RAZOR. In *IEEE Computer*, volume 37(3), March 2004.

[2] D. Bertozzi, L. Benini, and G. de Micheli. Low power error resilient encoding for on-chip data buses. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, page 102,

Washington, DC, USA, 2002. IEEE Computer Society.

[3] Keith A. Bowman and James D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. In *IEEE JSSC*, Feb 2002.

[4] Nicola Campregher, Peter Y. K. Cheung, George A. Constantinides, and Milan Vasilko. Analysis of yield loss due to random photolithographic defects in the interconnect structure of fpgas. In *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pages 138–148, New York, NY, USA, 2005. ACM Press.

[5] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar. An integrated quad-core opteron processor. In *ISSCC*, 2007.

[6] Albert V. Ferris-Prabhu. *Introduction to Semiconductor Device Yield Modeling*. Artech House Publishers, August 1992.

[7] Joshua Friedrich, Bradley McCredie, Norman James, Bill Huott, Brian Curran, Eric Fluhr, Gaurav Mittal, Eddie Chan, Yuen Chan, Donald Plass, San Chu, Hung Le, Leo Clark, John Ripley, Scott Taylor, Jack Dilullo, and Mary Lanzerotti. Design of the power6 microprocessor. In *ISSCC*, 2007.

[8] J. C. Harden and N. R. Strader II. Architectural yield optimization for wsi. *IEEE Trans. Comput.*, 37(1):88–110, 1988.

[9] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the cell multiprocessor. *IBM J. Res. Dev.*, 49(4/5):589–604, 2005.

[10] Manho Kim, Daewook Kim, and G. E. Sobelman. Mpeg-4 performance analysis for a cdma network-on-chip. In *International Conference on Communications, Circuits and Systems*, May 2005.

[11] Predictions Software Ltd. Yield modeling, 2005.

[12] Umesh Gajanan Nawathe, Mahmudul Hassan, Lynn Warriner, King Yen, Bharat Upputuri, David Greenhill, Ashok Kumar, and Heechoul Park. An 8-core 64-thread 64b power-efficient sparc soc. In *ISSCC*, 2007.

[13] Umit Y. Ogras, Radu Marculescu, Puru Choudhary, and Diana Marculescu. Voltage-frequency island partitioning for gals-based networks-on-chip. In *Proc. IEEE/ACM Design Automation Conf., San Diego*, June 2007.

[14] David A. Patterson and John L. Hennessy. *Computer architecture: a quantitative approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[15] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall.

[16] Nabeel Sakran, Marcelo Yuffe, Moty Mehalel, Jack Doweck, Ernest Knoll, and Avi Kovacs. The implementation of the 65nm dual-core 64b merom processor. In *ISSCC*, 2007.

[17] C. H. Stapper. Improved yield models for fault-tolerant memory chips. *IEEE Trans. Comput.*, 42(7):872–881, 1993.

[18] C. E. Stroud. Yield modeling for majority voting based defect-tolerant vlsi circuits. In *IEEE Southeastcon*, pages 229–236, 1999.

[19] Sriram Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Priya Iyer, Arvind Singh, Tiju Jacob, Shailendra Jain, and Sriram Venkataraman. An 80-tile 1.28tflops network-on-chip in 65nm cmos. In *ISSCC*, 2007.

[20] Yutaka Yoshida, Tatsuya Kamei, Kiyoshi Hayase, Shinichi Shibahara, Osamu Nishii, Toshihiro Hattori, Atsushi Hasegawa, Masashi Takada, Naohiko Irie, Kunio Uchiyama, Toshihiko Odaka, Kiwamu Takada, Keiji Kimura, and Hironori Kasahara. A 4320mips four-processor core smp/amp with individually managed clock frequency for low power consumption. In *ISSCC*, 2007.

[21] Zhiyi Yu and Bevan Baas. Performance and power analysis of globally asynchronous locally synchronous multi-processor systems. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 378–384, March 2006.

[22] Zhiyi Yu, Michael Meeuwsen, Ryan Apperson, Omar Sattari, Michael Lai, Jeremy Webb, Eric Work, Tinoosh Mohsenin, Mandeep Singh, and Bevan Baas. Performance and power analysis of globally asynchronous locally synchronous multi-processor systems. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 378–384, March 2006.