Multi-Granularity Redundancy in Multi-Core SIMT

Daniel A. Epstein, Kevin Skadron
Dept. of Computer Science
University of Virginia
Charlottesville, VA 22904 USA
dae5y@virginia.edu,
skadron@cs.virginia.edu

Brett H. Meyer
Dept. of Electrical and Computer Engineering
McGill University
Montréal, QC H3A 0E9 Canada
brett.meyer@mcgill.ca

ABSTRACT

Die yields are decreasing. The structure of multi-core SIMT architectures makes them suitable for applying redundancy at multiple levels of granularity, *e.g.*, at the core- and lane-level. We observe that applications prefer SIMT architectures with different kinds of sparing; redundancy can reduce cost by over 20% in some applications.

Keywords

System-level design, yield, redundancy

1. INTRODUCTION

As manufacturing processes scale to small feature sizes, devices are more likely to experience transient upsets, early performance degradation, and even failure in the field [1]. Furthermore, manufacturing devices that operate according to their specification in the first place is also increasingly challenging: yield losses are mounting, due to systematic and random defects [2], as well as parametric failure resulting from process variation [3]. Computers must now be over-provisioned at the microarchitectural-level (*e.g.*, using redundant execution units) or system-level (*e.g.*, using redundant cores) to ensure that systems with some defects can satisfy system specifications and can, therefore, be sold.

We hypothesize that given different performance and cost targets, different redundancy strategies emerge as most cost-effective. Yield can be improved in a number of ways [4]:

- by increasing design margins (over-engineering),
- by implementing redundant circuitry (functional unit or microarchitectural redundancy), and
- by integrating spare cores (and other components).

It is clear from industrial adoption that disabling defective cores improves yield by salvaging otherwise defect dice; the recovered dice occupy a different (lower cost) market segment, but the fact that they can be sold at all reduces the manufacturing cost of the defect-free dice. However, a question remains: under what circumstances does such coarse-grained redundancy *optimize* yield?

In this paper, we investigate the application of redundancy at multiple granularities to Single-Instruction, Multiple-Data (SIMD) architectures. Multi-core SIMD are especially well-suited to incorporating multiple granularities of redundancy, as there are multiple levels of design abstraction at which components are replicated. In our study, we focus on Single-Instruction Multiple-Thread (SIMT) architectures, where this replication occurs within (a) the system, which consists of multiple cores; and, (b) the cores, each of which consist of a front-end (consisting of the instruction cache, and fetch and decode stages) and multiple processing elements (hereafter referred to as *lanes*, consisting of the register file, data cache, and execution and write-back stages).

Embedded SIMT architectures present a unique opportunity for yield improvement. As different applications exhibit different types and amounts of parallelism, different designs are expected to strike the best performance-cost trade-offs. As the relative mix of components (cores vs. lanes) changes, the most effective strategy for improving yield is also expected to change (redundant cores vs. redundant lanes).

In this paper, we investigate the opportunity to reduce manufacturing costs in the presence of random defects by employing multi-granularity redundancy in SIMT architecture. We observe that when considering the space of performance-cost trade-offs, some design points call for coarse-grained sparing, others fine-grained sparing, and others still no redundancy at all. This is significant: tools are therefore needed to assist designers with determining when and what sort of redundancy to apply to optimize such systems.

2. RELATED WORK

Prior work has proposed the application of redundancy in order to improve manufacturing yield. Redundancy has long been employed to protect memories [2]. More recently, reconfiguraable circuits have been proposed to reduce the cost of redundancy in logic [5]. StageNet further reduces this by allowing cores with defective pipeline stages to make use of functional stages in neighboring cores [6].

Core level redundancy has already been adopted by industry and applied to general-purpose multiprocessors [7, 8]. Recent research has automated the process of allocating redundant cores, and memories in network-on-chip-based embedded systems for cost-effectively improving the yield of application-specific systems [9]. Other work has investigated the relative benefits of circuit- and core-level redundancy in general-purpose processors and multi-processors [10, 11, 4]. In general, the authors observe that as core counts increase, the relative benefits of core sparing increases, too.

Our work differs from the above in that we are specifically concerned with improving the yield of multi-core SIMT. The design of multi-core SIMT is unique from the standpoint that, since these systems simultaneously take advantage of thread-level parallelism (using multiple cores) and data-level parallelism (using multiple lanes per core), different applications call for different mixes of cores and lanes per core. As a result, different applications also call for different mixes of redundancy; this is the focus of our investigation.

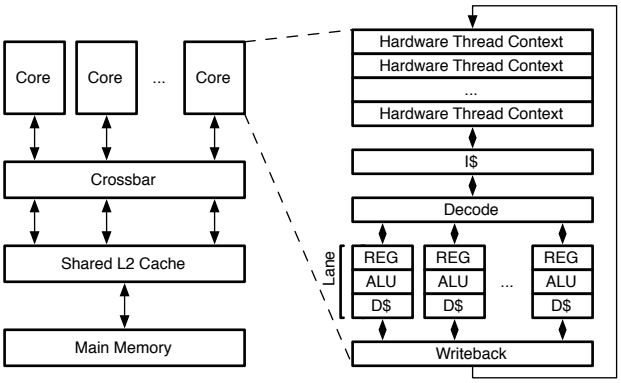


Figure 1: The hierarchical design of multi-core SIMT architectures makes them particularly well-suited to yield enhancement via coarse-grained redundancy.

3. OPTIMIZING SIMT YIELD WITH MULTI-GRANULARITY REDUNDANCY

An example multi-core SIMT system is illustrated in Figure 1. At the top level, a SIMT architecture is composed of cores, interconnect, and the shared memory hierarchy. Each core contains an instruction cache (I$), and front-end (FE) fetch and decode logic. This FE controls the execution of several processing elements (PEs) or *lanes*. n-wide SIMT denotes a SIMT core with n lanes. Each lane is composed of a register file, execution logic and a data cache (D$). In order to hide memory access latency, each lane supports fine-grained hardware multithreading in which multiple threads have their registers co-resident on a PE.

Multi-core SIMT architectures take advantage of both thread-level parallelism (TLP) and data-level parallelism (DLP): the lanes of individual cores take advantage of DLP, for instance, by executing independent loop iterations; each core can execute different sets of iterations of a loop, different threads in an application, or different applications.

SIMT performance is heavily application dependent: different applications express different parallelism and memory bandwidth requirements; different applications, therefore, will often be best accelerated by different architectures.

This variability in the design space for multi-core SIMT systems has an important consequence for yield enhancement: systems optimized for different applications may call for different redundancy. Fortunately, both the programming model and physical design of SIMT architectures make implementing redundancy at each of these granularities both relatively easy and particularly beneficial, as compared with equivalent strategies in general purpose designs.

3.1 Core Sparing

Like homogeneous multi-core systems, multi-core SIMT yield can be increased by integrating spare cores. When one core fails, it is easily replaced; the performance effect of the relative locations in the interconnect of the spare and its replacement can be minimized by virtualizing the on-chip communication network [12]. In this paper, we consider cold spares: spare cores are only used when a core is defective. Yield loss occurs when a system doesn't have enough functioning cores to satisfy the specification. Hot spares and performance binning are the subject of future work.

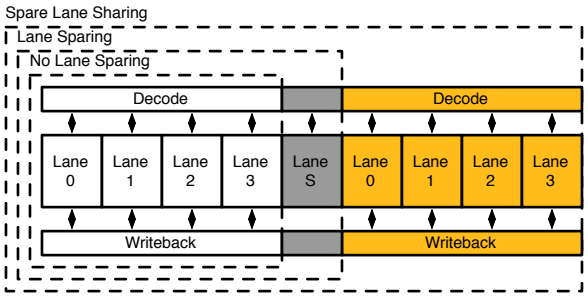


Figure 2: The physical design of SIMD architecture allows two cores to potentially share a single redundant lane.

3.2 Lane Sparing

One factor that distinguishes multi-core SIMT architectures from general purpose multi-core architectures is an additional level of hierarchy in the organization. Each SIMT core is essentially a multi-processor in its own right, but with redundant FEs removed to improve power and area efficiency for applications that exhibit data-level parallelism and hence can leverage lockstep execution among the PEs in a core. SIMT core yield can be increased be integrating spare lanes, as illustrated in Figure 2. Without spare lanes, the core is composed of the white components. When a core integrates a spare lane, the gray portions of decode and writeback and Lane S are used to replace the faulty lane. As with spare cores, the performance effect of the location of the spare relative the failed lane can be minimized with virtualization. Core failure occurs when a core doesn't have enough functioning lanes to satisfy the specification. Spare lanes provide superior defect tolerance to spare functional units in general purpose cores for two reasons. First, the lanes are larger as a fraction of the core. As a result, they are both more susceptible to defect, but yield also benefits more from covering them with redundancy. Second, the lanes are homogeneous. As a result, a single redundant lane covers all other lanes in the core. Functional unit duplication, on the other hand, covers a smaller fraction of core area and is also less versatile.

3.3 Spare Lane Sharing

The homogeneity of lanes, and the significant role they play in SIMT architecture, presents an additional, unique opportunity for low-cost yield enhancement: sharing a spare lane between two cores, illustrated in Figure 2. When the two cores in the figure share a spare lane, if either the white core or the orange core have a spare lane, Lane S and the corresponding decode and write-back logic can be employed; this only prevents core failure when there isn't a defective lane in both cores. The performance cost of cores sharing access to a redundant resource is high in general purpose processors [6]; the hierarchical design of SIMT cores, however, means that the spare lane can be included in either core with only marginally greater performance loss than if the spare were dedicated to a single core. Quantifying this potential performance loss is the subject of future work.

4. EXPERIMENTAL SETUP

In order to determine what form of redundancy is most applicable given a particular (a) configuration and (b) application, we performed an extensive performance and cost

Table 1: SIMT Configuration Fixed Parameters

Component	Configuration
Core	1 GHz, 0.9 Vdd, in-order, Alpha ISA
Instruction cache	16 KB, 4-way set associative, 32 B block size, 1 cycle hit latency, 4 MSHRs, LRU, write-back
Data cache	Variable size, 8-way set associative, 32 B block size, MESI directory-based coherence, 3 cycle hit latency, LRU, write-back, 32 MSHRs, up to 8 requests
L2 cache	4096 KB, 16-way set associative, 128 B block size, 30 cycle hit latency, LRU, write-back, 64 MSHRs, up to 8 requests each
Crossbar	300 MHz, 57 GB/s
Memory	300 cycle access latency

evaluation of a wide variety of multi-core SIMT systems on a set of diverse benchmarks.

4.1 SIMD Configurations and Benchmarks

We used MV5 [13] to simulate the performance of SIMD configurations. For each configuration, we fixed a number of parameters; these are summarized in Table 1. The configurations we considered varied the number of cores, lanes per core (SIMT width), hardware threads per core (SIMT depth), and L1 data cache size. We considered systems with N cores, $N \in \{1, 2, 4, 6, \ldots, 20\}$. We considered systems with W lanes and D threads, $W, D \in \{1, 2, 4, 8, \ldots, 64\}$ with the constraint that the number of threads per lane never exceeds the number of threads per core. We also considered systems with L1 D$ sizes of S KB, $S \in \{16, 32, 64\}$.

We selected benchmarks from several suites: Minebench [14], SPLASH-2 [15], and Rodinia [16]. The set we selected, Fast Fourier Transform, Filter Edge Detection, Thermal Simulation, LU Decomposition, Merge Sort, Shortest Path, KMeans Clustering, and SVM Learning, have been previously used in the literature to evaluate multi-core SIMD performance [17]. We also analyzed the average performance across all of the benchmarks as an overall estimate.

4.2 Redundancy and Cost Evaluation

We evaluated the area and yield of each configuration, considering the baseline (no redundancy) and a number of variants with redundancy. We applied core-level redundancy, lane-level redundancy, shared redundant lanes, and every combination therein. Redundant lane sharing is applied by dividing the system into pairs of cores and allocating a single spare lane for each pair.

To estimate area, we measure the realistic sizes for the different units of a core based on a publicly available die photo of the AMD Opteron processor in 130 nm technology, scaled to 65 nm assuming a 0.7 scaling factor per generation. We assume each SIMT lane implements a 32-bit data path.

We calculate yield using the Poisson Model [18],

$$Y_{Component} = e^{-D_0 \cdot A \cdot KR}, \quad (1)$$

where D_0 is the defect density, A is the component area, and KR is the *kill ratio*. The kill ratio represents the component's sensitivity to defects: a kill ratio of 85% implies that 85% of defects cause component failure. The kill ratio and defect density were obtained from the ITRS 2009 report on electrical defects [18]. We considered the yield of caches, SIMD lanes, and glue logic, whose yields were multiplied together to determine the yield of a core. The yield of the system is the product of the yield of its cores.

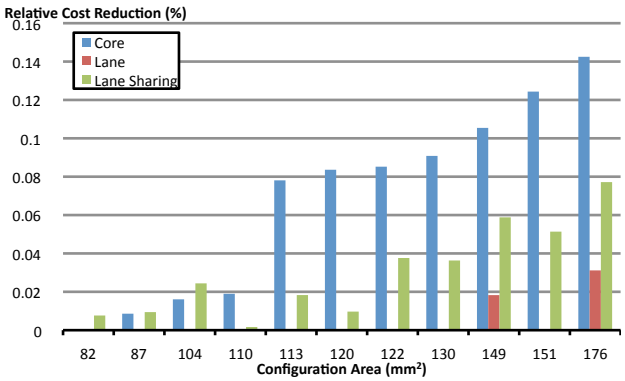


Figure 3: Relative cost reduction when designing for the average case (across all benchmarks).

When redundant cores and SIMT lanes are introduced, the yield follows a binomial distribution

$$Y_{Redundant} = \sum_{i=b}^{b+s} C_i^{(b+s)} \cdot Y_{Component}^{i} \cdot Y_{Component}^{(b+s-i)}, \quad (2)$$

where b is the base number of components and s is the number of spares. This model assumes that the yield of each component being replicated is independent and identical.

Once area and yield have been estimated, the cost of a single die can be estimated [19] as

$$Cost_{die} = \frac{Cost_{wafer}}{Dice/Wafer \cdot Y_{die}}, \quad (3)$$

where we estimate the dice per wafer as

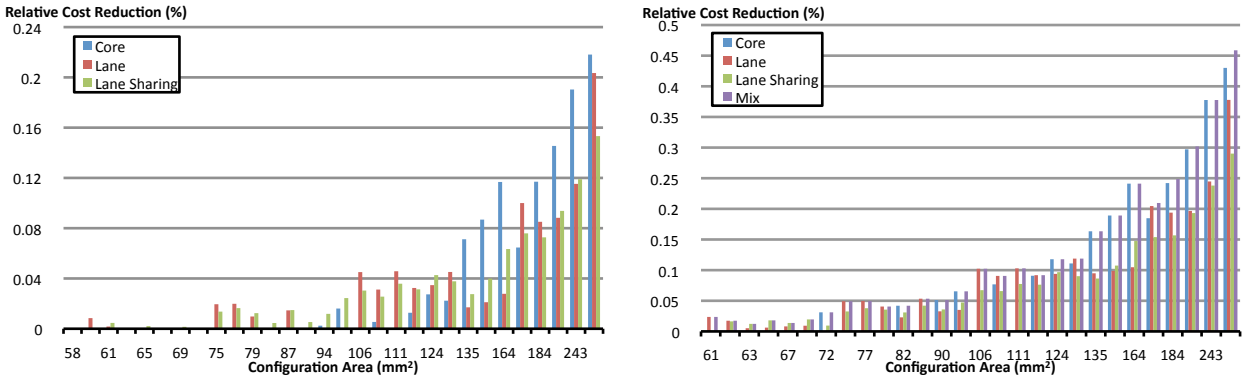
$$Dice/Wafer = \frac{\pi \cdot (Diam_{wafer}/2)^2}{Area_{die}} - \frac{\pi \cdot Diam_{wafer}}{\sqrt{2 \cdot Area_{die}}}. \quad (4)$$

We assume a wafer with a diameter of 300 mm, and that $Cost_{wafer}$ is \$3,000. We assume that the wafer yield is 1, ignoring the effect of entire wafers being discarded if more defects occur than a given threshold.

5. RESULTS

To fully characterize the design space, we performed extensive evaluation, considering an average of 850 configurations for each benchmark. To identify configurations of note, we then selected those on the cost-performance Pareto-optimal front for each benchmark, and in the average case. In general, the points on the Pareto-optimal curves varied in terms of cache size, but all had few SIMT lanes compared to points not on the curve. As performance improved, the number of cores, and thus the cost, increased, eventually reaching a point where incremental improvements require significant increases in cost.

On average across eight benchmarks (Figure 3), redundancy does not improve yield inexpensively enough to reduce the die cost (Eq. (3)) of performance-cost Pareto-optimal configurations until processors are larger than 82 mm^2. Between 82 mm^2 and 104 mm^2, lane sharing reduces cost by 1-2%. As area exceeds 104 mm^2, this improvement increases to over 8% under a core sparing regime; improvement peaks at 14% for a 176 mm^2 design.



(a) Filter

(b) Filter under double defect density

Figure 4: An example of application-specific relative cost reduction.

The cost improvement each benchmark experiences due to redundancy is not identical, however. KMeans clutering begins to show improvements of 1% or more to cost around processor areas of 69 mm^2. For ShortestPath, which utilizes Dynamic Programming to search a Chess game tree, using core redundancy is always advantageous to other types, rising gradually from 2% to 14% as processor area increases. Filter (Figure 4), which performs edge detection by convolution on an input image, redundancy begins to be viable around processor areas of $75mm^2$ and lane *sparing* results in the greatest cost reduction.

In order to observe the benefit of multi-granularity redundancy under adverse manufacturing conditions, we repeated our experiments for a scenario under which the ITRS targeted defect densities are doubled. Figure 4b plots the die cost reduction for designs on the Pareto-optimal front in this case. When the defect density is doubled, redundancy more consistently reduces cost for designs of less than 75 mm^2. The importance of redundancy also grows much faster, with over 16% cost reduction for processors larger than 135 mm^2. It is also noteworthy that in this case, some configurations (178, 189, and 311 mm^2) benefit the most from a mix of redundancy techniques, as indicated by the mix series.

6. CONCLUSION

We investigated multi-granularity redundancy in SIMT architectures and showed that, in some cases, shared spare lanes are preferable to spare cores. When different applications require different configurations for optimal performance-cost trade-offs, different yield improvement strategies emerge; this diversity only grows under increasing defect densities. For example, cost savings using core sparing starting at 2% occur for the ShortestPath benchmark at areas of at least 79 mm^2, while the Filter benchmark sees cost reductions using lane sparing at areas larger than 75 mm^2. This has significant implications for designers: the design space for SIMT architectures is large and complex, and as yield and reliability concerns grow, designers will need tools to search for the best application-specific designs. The development of such tools is the subject of ongoing research.

7. REFERENCES

[1] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE MICRO*, vol. 25, no. 6, 2005.

[2] I. Koren and Z. Koren, "Defect tolerance in VLSI circuits: Techniques and yield analysis," *Proceedings of the IEEE*, vol. 86, no. 9, 1998.

[3] E. Humenay *et al.*, "Impact of process variations on multicore performance symmetry," in *DATE'07*, Feb. 2007.

[4] Y. Markovsky and J. Wawrzynek, "On the opportunity to improve system yield with multi-core architectures," in *DFM&Y'07*, October 2007.

[5] V. V. Kumar and J. Lach, "Heterogeneous redundancy for fault and defect tolerance with complexity independent area overhead," in *DFT'03*, November 2003.

[6] S. Gupta *et al.*, "StageNet: Reconfigurable fabric for constructing dependable CMPs," *IEEE Trans. Comput.*, vol. 60, Jan. 2011.

[7] S. Rusu *et al.*, "A 45nm 8-core enterprise Xeon® processor," in *ISSCC'09*, February 2009.

[8] J. A. Kahle *et al*, "Introduction to the Cell multiprocessor," *IBM Journal of Research and Development*, vol. 49, 2005.

[9] B. H. Meyer *et al.*, "Slack allocation for yield improvement in NoC-based MPSoCs," in *ISQED'10*, March 2010.

[10] S. Premkishore *et al.*, "Exploiting microarchitectural redundancy for defect tolerance," in *ICCD'03*, Oct. 2003.

[11] E. Schuchman and T. Vijaykumar, "Rescue: A microarchitecture for testability and defect tolerance," in *ISCA'05*, June 2005.

[12] L. Zhang *et al.*, "Defect tolerance in homogeneous manycore processors using core-level redundancy with unified topology," in *DATE'08*, March 2008.

[13] J. Meng and K. Skadron, "Avoiding cache thrashing due to private data placement in last-level cache for manycore scaling," in *ICCD'07*, 2007.

[14] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary, "Minebench: A benchmark suite for data mining workloads," in *IISWC'06*, 2006.

[15] S. C. Woo *et al.*, "The SPLASH-2 programs: Characterization and methodological considerations," in *ISCA'95*, 1995.

[16] S. Che *et al.*, "A performance study of general purpose applications on graphics processors using CUDA," *JPDC*, 2008.

[17] J. Meng, D. Tarjan, and K. Skadron, "Dynamic warp subdivision for integrated branch and memory divergence tolerance," in *ISCA'10*, June 2010.

[18] SIA, *International Technology Roadmap for Semiconductors*, 2009. http://public.itrs.net.

[19] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Francisco, CA: Morgan Kaufman, 4th ed., 2007.