

Reliability Tradeoffs in Design of Cache Memories

Hossein Asadi, Vilas Sridharan, Mehdi B. Tahoori, David Kaeli
Northeastern University, Dept. of ECE, Boston MA 02115
Email: {gasadi, vilas, mtahoori, kaeli}@ece.neu.edu

Abstract

Cache memory is a fundamental component of all modern microprocessors. Caches provide for efficient read/write access to memory, and their reliability is essential to assure dependable computing. Among the primary threats to cache reliability are soft (transient) errors. When incident in a cache, these errors can corrupt data values or result in invalid state, and can easily propagate throughout the system to cause data integrity issues.

In most modern processors, a significant portion of the chip area is dedicated to the L2 cache. In this paper, we examine the vulnerability (on a per bit basis) of different components of on-chip L2 caches, including data and tag bits. Vulnerability per bit is important when considering the reliability of a design; the higher the per-bit vulnerability of a structure, the greater its impact on overall reliability. We show that the vulnerability per bit of the tag array and data array differ significantly, and thus should motivate our implementation decisions when considering design tradeoffs for reliability.

1 Introduction

Cache memory is a fundamental component used to enhance the performance of modern-day microprocessors. Data caches, typically implemented in static random access memory (SRAM) technology, store frequently accessed data close to the processor pipeline in order to reduce the latencies associated with accessing off-chip data memory. Errors occurring in the level 1 and 2 caches can propagate to main memory and can easily lead to data integrity issues [17].

Soft errors, also called Single Event Upsets (SEUs), are the main source of vulnerability in a memory system [21]. Most soft errors are caused either by neutrons

from cosmic rays or by alpha particles from packaging material. Currently, memory cells are the most vulnerable components to soft errors [9, 19]. Estimated soft error rates measured on typical designs (such as microprocessors, network processors, and network storage controllers) show that sequential elements and unprotected SRAMs contribute to 49% and 40% of the overall soft error rate, respectively [19].

Most previous studies have targeted L1 caches [2, 29] and processor pipelines [20, 26, 28]. The reliability of these system elements is crucial for both system dependability and performance. In addition, many modern processors implement protection, including error correction codes (ECC) on L2 data cache arrays [11, 14, 15]. However, only a few processors provide protection bits for L2 tag arrays [1, 24], since the extra delay imposed by an ECC computation on tag bits increases cache hit times and miss times. As on-chip L2 caches increase in size, the size of their tag arrays increases proportionally. This growth will increase the likelihood of errors in the tag array, and will increase the vulnerability of L2 tag addresses to soft errors.

When using a write-back policy for L2 caches, errors on tag addresses can have a more serious impact on data integrity than errors in the data arrays. This is because data array errors can be recovered by employing time redundancy techniques [18]. But in the case of an error in a tag address, the most recent data written to the block cannot be recovered because the original block address is no longer available.

One microarchitectural design choice that can reduce the vulnerability of level 1 data (DL1) caches is to use a write-thru policy instead of a write-back policy. Experiments on SPEC2000 benchmarks show that for a 64KB 4-way set associative DL1 cache, vulnerability to soft errors is reduced by more than 85% by using a write-thru cache [2]. In a write-back cache with allocate-on-write-miss, a single word written to a cache line causes the data

Processor	DL1	IL1	L2
Pentium IV [11]	8K/16K	8K/16K	256K/512K/1M
POWER4 [14]	64K/128K	64K	1.41M
POWER5 [15]	96K*	96K*	1.875M
AMD Athlon64 [1]	64K	64K	1M
Itanium2 [24]	16K	16K	256K
IBM 390 [25]	256K	256K	4M

Table 1. Survey of some cache organizations in recent microprocessors (* = per core).

and tag/status bits in the entire line to become vulnerable until the line is written back to memory. In a write-thru cache, this line is not vulnerable since the data is immediately written to the memory with a very short vulnerable time during the residency period in the store buffer. Unfortunately, write-thru is not a practical choice (for bandwidth reasons) for L2 caches, and as such, almost all current L2 caches employ a write-back policy [1, 11, 14, 15].

In this paper, we explore the vulnerability *per bit* of different components of on-chip caches, as this will allow us to make more accurate assessments of vulnerability. In our analysis we include data, tags, and status bits. We show that the vulnerability per bit of these components differs significantly, and thus should motivate a number of design tradeoffs when considering reliability. In particular, this parameter can be used to guide our selection of protection techniques, such as *hardening* [6, 30] or spatial redundancy techniques (e.g., ECC or parity).

In this paper, we utilize the reliability-performance framework proposed in [2]. Experiments running SPECint2000 and SPECfp2000 show that the vulnerability per bit of L2 tag arrays is significantly higher, and thus more critical than the vulnerability per bit of the DL1 data array.

The rest of this paper is organized as follows. Section 2 describes error rate computation and provides a brief introduction to our reliability computation model. Section 3 presents our experimental setup. Section 4 presents our reliability profiling and simulation results. Section 5 discusses possible hardening techniques that would be appropriate for cache tags, and Section 6 concludes the paper.

2 Background

The *Soft Error Rate* (SER) for a device is defined as the error rate due to SEUs. A system’s error rate bud-

get for repairable components is commonly expressed in terms of the Mean Time Between Failures (MTBF) and the Mean-Time-To-Repair (MTTR), whereas for non-repairable components, Mean-Time-To-Failure (MTTF) is normally used. Failures-in-Time (FIT) is another commonly used error rate metric. FIT error rates are inversely proportional to MTBFs if the reliability function obeys the exponential failure law [12]. One FIT is equal to one failure in a billion hours. Current predictions show that typical FIT rates for latches and SRAM cells (measured at sea level) vary between 0.001-0.01 FIT/bit [10, 22]. The overall FIT rate of a chip is calculated by adding the effective FIT rates of all of the individual components. The FIT rate of each component is the product of its raw FIT rate and its associated Vulnerability Factor. The Vulnerability Factor (VF) is defined as the fraction of faults that become errors [20]. Therefore, the FIT rate of the entire system can be computed as follows:

$$FIT_{Chip} = \sum_i raw\ FIT_{Element(i)} \times VF_{Element(i)} \quad (1)$$

The reliability of a chip during the period [0, t] is defined as the probability that the chip operates correctly throughout this period [12]. The reliability of a chip at time t can be computed as follows:

$$Reliability_{Chip}(t) = e^{-FIT_{Chip} \times t} = e^{-\frac{t}{MTTF_{Chip}}} \quad (2)$$

In this paper, we use the terms Critical Words (CWs) and Critical Time (CT), which we first defined in [2]. We define Critical Words (CW) as those words in the cache that are either eventually consumed by the CPU or committed to memory by a write. In other words, if the CPU reads a word from the cache or if a dirty word of the cache is written to the memory, it is classified as a CW. The *Critical Time* (CT) associated with a CW is defined as the time period in which the context of that CW is important. CT is the elapsed time between the time when the word is brought into the cache and the time when it is used by the CPU; or the elapsed period time from the clock cycle in which a word is changed by the CPU to the clock cycle in which the same data is written back to the memory. If an error in a CW is encountered during its critical time, this can result in an erroneous value being propagated. All other words that are not CWs are called *Non-critical Words* (NWs). Any failure experienced by a NWs should not affect the correct operation of the system.

Suppose that the words $W1$ and $W2$ are fetched into the cache in cycle 10. $W1$ is read by the CPU in cycle 40 and $W2$ is changed by the CPU in cycle 30. In cycle 50, $W1$ is replaced by another word and $W2$ is written back to the memory. The CT of $W1$ is calculated as $40 - 10 = 30$ and the CT of $W2$ is calculated as $50 - 30 = 20$. Note that the replacement time in the read operation (cycle 50) and the entrance time in the write operation (cycle 10) do not affect the critical time. This is shown in Figure 1.

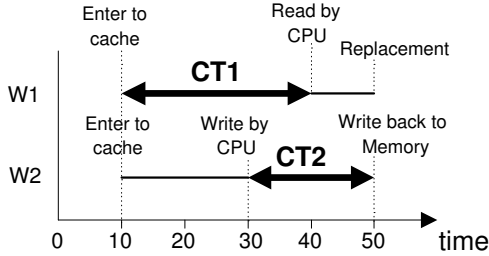


Figure 1. Critical words and critical time definitions.

2.1 Errors in data, tag, and status bits.

To investigate the impact of errors in address tags and status bits, we extend the classification provided in [2] and [17] and study how these errors individually affect tag reliability. There is a tag address associated with every cache line. The width of the tag is a function of the size of the address space, the number of cache lines, and the associativity of the cache. Bit changes in the tag array may cause the following types of errors:

Pseudo-miss: The tag address of a line does not match the requested address tag, but the line should have matched.

Pseudo-hit: The tag address of a line matches the requested address tag, though no match should have been found on this entry.

Replacement error: The tag address of a line is changed after the line has been modified but before it is written to lower memory.

Multi-hit: The tag address of a line is changed to match another tag address in the same cache set.

We model status bits as follows. Status bits typically consist of a valid bit, and in a write-back cache, a dirty bit. Dirty bits are used only in data caches. For an error that occurs in a dirty bit, when the error flips a 0 to a 1,

the flip does not affect data integrity. But when a soft error causes a change from 1 to 0 in a dirty bit, if this line is replaced before it is re-written, the new value of the line is lost.

For an error occurring in a valid bit, if the bit changes from 1 to 0, the impact to data integrity depends on the current state of the line. If the line was clean, there may be a slight impact on performance (i.e., an extra cache miss), but there will be no data corruption. If the line was dirty, however, the most recent data written to the line will be lost.

In our vulnerability computation algorithm for status bits, we have computed the vulnerability exactly for the cases described above. One other situation that occurs for valid bits is when an error changes the valid bit from 0 to 1. This will change the status of an invalid line to a valid line. This would corrupt data only if the line is requested by the CPU before it is replaced by another clean line. We have chosen not to model this effect since our past experiments have shown that it constitutes less than 0.2% maximum to the overall vulnerability.

Results that are never re-referenced (results of dynamically dead instructions) contribute to the vulnerability only if they are written to the next level of memory. Results which are referenced by dynamically-dead instructions will contribute to the vulnerability. Since our methodology is focused on computing the reliability of the cache and not the entire system, we say a word is vulnerable if it propagates from the cache to another part of the system (either to the processor or to the memory). In addition, the vulnerability of silent stores [18] is computed properly: assuming no intervening reads, the first store will not contribute to the cache vulnerability.

3 Experimental Setup

For these experiments, we used the *sim-outorder* processor model provided in the *SimpleScalar 3.0* framework [5]. To evaluate vulnerability, we have extended the SimpleScalar model to integrate our reliability estimation methodology. Our evaluation uses a subset of the SPECint2000 and SPECfp2000 benchmark suite [27] compiled for the ALpha ISA [16]. We utilize the SimPoint early simulation points in all of our results [23].

The default system parameters (cache size, associativity, etc.) are provided in Table 2, and were chosen to be representative of modern state-of-the-art processors. As can be seen in Table 1, the typical size of first-level data and instruction caches (DL1 and IL1) and the typical size

Configuration Parameter	Value
Processor	
Functional Units	4 integer ALUs, 1 integer multiplier/divider 4 FP ALUs, 1 FP multiplier/divider
LSQ Size / RUU Size	8 Instructions / 16 Instructions
Fetch / Decode / Issue / Commit Width	4 / 4 / 4 / 4 instructions/cycle
Fetch Queue Size	4 instructions
Cycle Time	1 ns
Cache and Memory Hierarchy	
L1 Instruction Cache (IL1)	64KB, 1-way, 64 byte lines 1 cycle latency, 18Kbit tag array
L1 Data Cache (DL1)	64KB, 4-way, 64 byte lines Write-thru, no allocate on write miss 1 cycle latency, 20Kbit tag array
L2	1MB unified, 8-way 128 byte lines, 6 cycle latency, Writeback
Memory	100 cycle latency
Branch Logic	
Predictor	Combined, bimodal 2KB table two-level 1KB table, 8 bit history
BTB	512 entry, 4-way
Mis-prediction Penalty	3 cycles

Table 2. Default configuration parameters used in our simulations.

of second-level caches (L2) in current high-performance processors are on the order of 64KB and 1MB, respectively.

4 Reliability Profiling Results

Figure 2 shows the L2 cache vulnerability profile for twenty SPEC2000 benchmark programs. From this data, we can determine the likelihood of having no failures (*reliability*) based on the target workload, duration, and raw environmental FIT rate. Our work shows that the data vulnerability constitutes more than 95% of the total L2 vulnerability. Accordingly, most current microprocessors use some form of spatial redundancy (e.g., ECC) on the L2 data bits [1, 11, 14, 15, 24], reducing the effective vulnerability of these bits to zero.

However, as Figure 3 shows, the L2 tag vulnerability is not insignificant. On average, it is greater than the vulnerability for the L1 data cache when running the SPEC2000 benchmarks. This is despite the fact that the L2 tag array has significantly fewer bits than the L1 data cache. Figure 4 shows the L2 cache tag vulnerability per bit for the same twenty SPEC2000 programs compared to the vulnerability per bit of the DL1 cache. Note that the per bit vulnerability of the L2 tags is about the same as for the

L2 data, but is almost 6 times greater than for the vulnerability obtained for the L1 data cache. On average, each bit protected in the L2 tags will reduce vulnerability more than a bit protected in the DL1.

This result leads us to the following conclusion. When looking at hardening techniques for memory elements, the vulnerability of a structure only tells part of the story. One should also look at the vulnerability per bit of the structure. If the per-bit vulnerability of a structure is high, more expensive error protection such as ECC or hardening may be required. If the per-bit vulnerability is low, less expensive methods such as parity can be used.

Notice that the per-bit vulnerability can also be viewed as follows: given a bit flip, what is the probability that a strike occurs during a *critical time*. Then the probability that a bit flip will occur during a critical time is higher on average for the L2 tags (0.35) than for the L1 data cache (0.06).

The higher vulnerability per bit of the L2 tag array can mostly be explained by the write-back L2 cache. To demonstrate this, we ran the *gcc* benchmark and profiled the distribution of critical time for both the DL1 data and L2 tag arrays. We chose *Gcc* because it had similar total vulnerability for L2 tag and L1 data, but significantly higher per-bit L2 tag vulnerability. The main

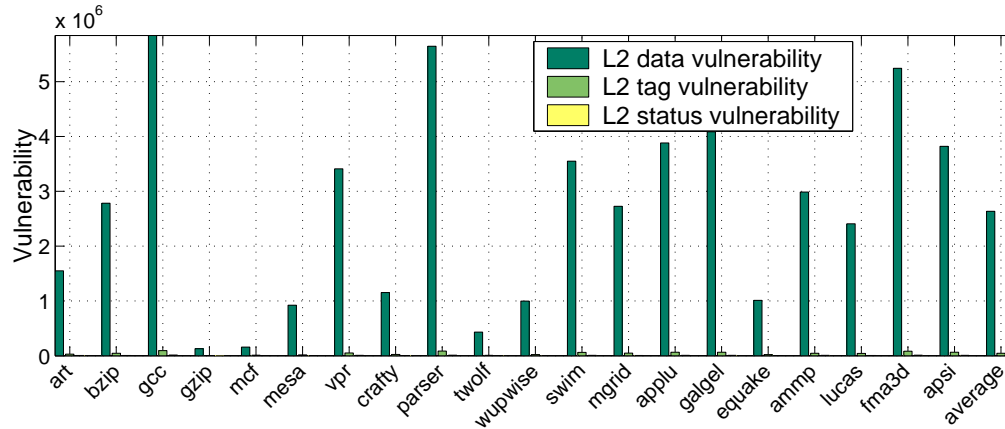


Figure 2. L2 Cache Vulnerability Profile (Total Vulnerability).

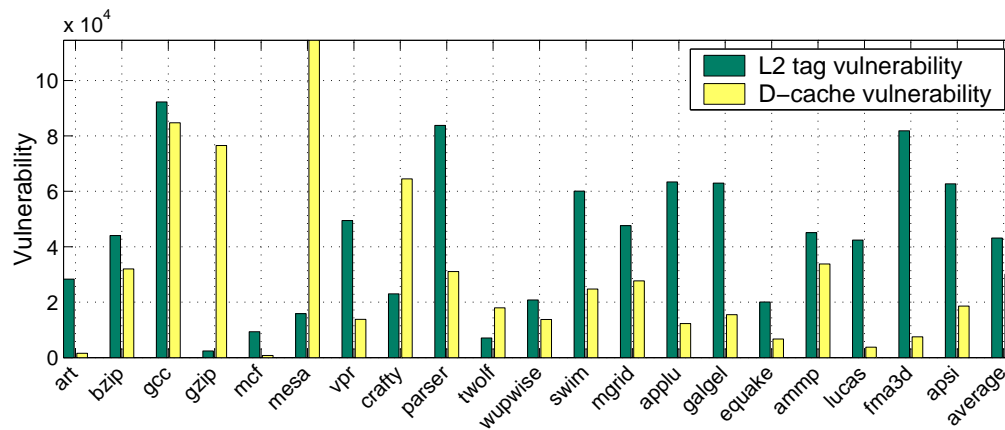


Figure 3. Comparison of L2 Tag Total Vulnerability with DL1 Total Vulnerability.

feature of note was that a large proportion (almost half) of L2 blocks had a critical time of several million cycles, while virtually no DL1 blocks had such a large critical time. The blocks with high critical times are dirty blocks in the L2 cache, which are critical until replacement in a write-back cache. This skews the distribution of critical times in the L2, causing the mean critical time of an L2 tag to be several times higher than for an L1 data block. This type of criticality in the L2 is known as *replacement error* [2].

5 Discussion

Design decisions that address vulnerability and reliability inevitably involve a number of design tradeoffs. Two of the most common tradeoffs are area and performance; power is also sometimes a constraint. When de-

signing reliability features into hardware, designers must consider how best to decrease vulnerability without impacting performance, power, and area.

A common question that arises is whether to use ECC, parity or hardening techniques to protect memory cells. In order to answer this question, we should first understand a little more about the origin of soft errors and the various hardening techniques that can be applied.

5.1 Origins of Soft Errors

Alpha particles arising from chip packaging material and neutrons from cosmic rays are the primary sources of soft errors. Alpha particles have charge, and thus, interact with matter via coulomb interaction. Neutrons have no charge, and their interaction with silicon is quite different. However, neutron strikes tend to generate more

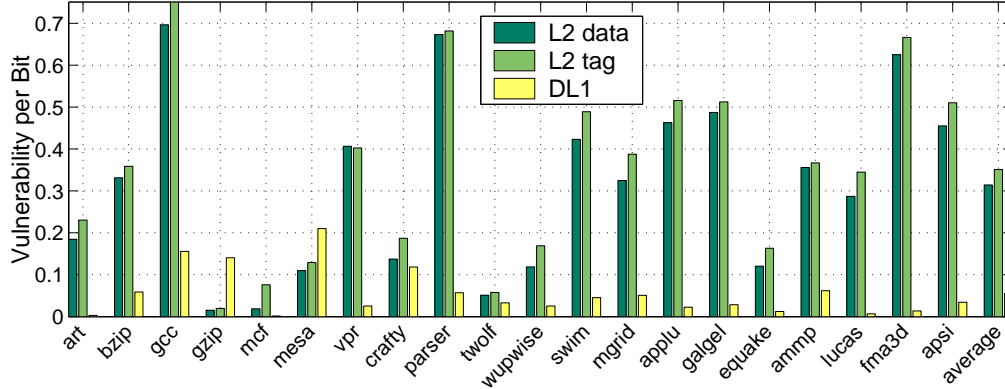


Figure 4. Comparison of L2 Tag Per-bit Vulnerability with DL1 Per-bit Vulnerability.

charge than alpha particle strikes. Typical strikes inject tens of femto-coulombs within a short period of up to tens of pico-seconds.

When a particle strikes a sensitive region of an SRAM cell, the charge that accumulates could exceed the minimum charge that is needed to flip the value stored in the cell, resulting in a soft error. The smallest charge that results in a soft error is called the critical charge (Q_{crit}) of the SRAM cell [8]. Particles that generate more charge than Q_{crit} will cause a soft error.

5.2 Hardening techniques

The objective of conventional hardening techniques is to increase the capacitance of critical nodes, and hence, to increase the reliability of the cell [30]. There are several ways to achieve this goal that vary in their effectiveness and their costs.

A basic hardening method is to increase transistor sizes, thereby increasing capacitance. A larger transistor can dissipate (sink) the injected charge more quickly, so that the transient does not achieve sufficient magnitude and duration to propagate to gates in the fanout. Using this technique, the added capacitance can have an overall impact of 6% to 8% increase on memory access time and 13% to 15% increase on block area [7]; the exact impact depends on the specific memory architecture, technology process, and the desired level of protection [31].

Another hardening technique is to isolate critical nodes in a circuit from the radiation-induced charge by using deep N-Well technology. Experimental results show that this technique reduces vulnerability to alpha particles by approximately 1.5x to 2.5x with a minimal area and performance impact [7].

A third technique to reduce soft-error susceptibility of

SRAM cells is to tweak the threshold voltage V_t of the device [6]. Unfortunately, this technique only works on asymmetric SRAM cells [3], and has no impact on the Q_{crit} for standard SRAM cells. Table 3 provides a summary of selected hardening techniques in terms of area impact and improved reliability.

5.3 ECC, Parity, or Hardening?

Both L2 tags and L1 data arrays can be on the critical path for data accesses, and as such, any reliability solution needs to have a very minimal performance impact. As such, ECC may be inappropriate for either array due to the relatively high performance cost for the ECC computation. While ECC and parity detect any single errors in SRAMs cells, many hardening techniques that have been proposed [6, 13, 30] try to dissipate or suppress the injected charge as quickly as possible, thus preventing the error from occurring at all. While these hardening techniques may impose a smaller performance penalty than ECC or parity, they may not eliminate all single errors.

However, the key factor about these hardening techniques is that they eliminate injected charges of up to particular energy (e.g., $Q = 0.2pC$). As particles of lower energy are far more plentiful than particles of higher energy, hardening an SRAM cell against particles of, for example, $0.2pC$ energy, will protect it against 80% to 90% of all single errors [4]. (This percentage depends on technology process, circuit characteristics and environmental factors such as particle flux.) Of the remaining particles that cause a single error, only 6% of these will cause a program error if they hit in the L1 data cache (since the vulnerability per bit of the DL1 is approximately 0.06). Similarly, 35% of these particles will cause a bit flip in the L2 tags (vulnerability per bit = 0.35); in order to keep

Technique	Area Impact	Vulnerability Reduction
ECC [7]	4-50%	>10x
Parity [7]	1-15%	>10x
Increase C [31]	13-15%	>20x
Deep N-Well [7]	Small	1.5-2.5x

Table 3. Survey of some cache hardening techniques.

the same error rate, the L2 tags should be hardened to protect against higher-energy particles than the L1 data cache.

6 Conclusion

This paper has built upon previous work in soft error modeling and proposed examining the per-bit vulnerability of the L1 data and L2 cache tags. The per-bit vulnerability can be used as a guide as to the level of protection needed for the specific bit array in question. We modeled this and simulated this using the SPEC2000 benchmark suite. Our data shows that the L2 cache tags have a high per-bit vulnerability compared to the L1 data cache, and as such, may warrant a higher level of protection from soft errors.

In our future work we plan on incorporating the ability to model the full impact of different hardening and spatial redundancy mechanisms. The goal will be to find the best mix of solutions for the class of workloads targeting a system.

References

- [1] AMD Athlon(TM) 64 Processor, <http://www.amd.com>.
- [2] H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli, "Balancing Performance and Reliability in the Memory Hierarchy," Proc. of the IEEE Intl. Symp. on Performance Analysis of Systems and Software (ISPASS05), pp. 269-279, March 2005.
- [3] N. Azizi, et al, "Low-Leakage Asymmetric-Cell SRAM", IEEE International Symposium on Low Power Electronics and Design, 2002.
- [4] S. J. Scitutto, "A System for Air Shower Simulations," Technical Report, Department of Fisica, National Univ of La Plata, Oct. 2001.
- [5] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," University of Wisconsin-Madison, Computer Science Dept., Technical Report No. 1342, June 1997.
- [6] V. Degalahal, R. Ramanarayanan, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, "The Effect of Threshold Voltages on the Soft Error Rate Memory and Logic Circuits," Proc. of 5th International Symposium on Quality Electronic Design, pp. 503-508, 2004.
- [7] N. Derhacopian, V.A. Vardanian, Y. Zorian, "Embedded Memory Reliability: the SER Challenge," IEEE Int. Workshop on Memory Technology, Design and Testing, pp. 104-110, 2004.
- [8] L. B. Freeman, "Critical Charge Calculations for a Bipolar SRAM Array," IBM Journal of Research and Development, Vol. 40, no. 1, pp 119-129, 1996.
- [9] J. Gaisler, "Evaluation of a 32-bit Microprocessor with Built-in Concurrent Error-Detection," Proc. of 27th Intl. Symp. on Fault-Tolerant Computing(FTCS-27), pp. 42-46, June 1997.
- [10] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walstra, and C. Dai, "Impact of CMOS Scaling and SOI on soft error rates of logic processes," Symposium on VLSI Technology, Digest of Technical Papers, PP. 73-74, June 2001.
- [11] Intel Pentium IV Processor, <http://www.intel.com>.
- [12] B. W. Johnson, "Design & analysis of fault tolerant digital systems," A&W Longman Publishing, ISBN:0-201-07570-9, Boston, MA, 1988.
- [13] J. Kumar and M.B. Tahoori, "A Low Power Soft Error Suppression Technique for Dynamic Logic", proc. of the IEEE Symposium on Defect and Fault Tolerant (DFT), 2005
- [14] S. Behling, R. Bell, P. Farrell, H. Holthoff, F.O. Connell, and W. Weir, "The POWER4 Processor Introduction and Tuning Guide," IBM redbooks, www.redbooks.ibm.com/pubs/pdfs/redbooks/sg247041.pdf, Nov. 2001.
- [15] R. Kalla, S. Balaram, J.M Tendler, "IBM Power5 Chip: a Dual-Core Multithreaded Processor," IEEE Micro, pp. 40-47, Vol. 24 , Issue 2, Mar-Apr 2004.
- [16] R. Kessler, "The Alpha 21264 Microprocessor," IEEE Micro, 19(2):24-36, March 1999
- [17] S. Kim and A. K. Somani, "Area Efficient Architectures for Information Integrity in Cache Memories," Proc. of the Intl. Symp. on Computer Architecture (ISCA'99), pp. 246-255, Atlanta, Georgia, May 1999.

- [18] K. M. Lepak and M. H. Lipasti, "Silent Stores for Free," Proc. of the Intl. Symp. on Microarchitecture (MICRO-33), pp. 22-31, Dec. 2000.
- [19] S. Mitra, N. Seifert, M. Zhang, Q. Shi and K. Kim, "Robust System Design with Built-In Soft-Error Resilience", IEEE Computer, vol. 38, pp. 43-52, Feb. 2005.
- [20] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," Proc. of the Intl. Symp. on Micro-architecture (MICRO-36), pp. 29-40, 2003.
- [21] H. T. Nguyen and Y. Yagil, "A Systematic Approach to SER Estimation and Solutions," Proceedings of the Intl. Reliability Physical Symp., pp. 60-70, Dallas, Texas, 2003.
- [22] E. Normand, "Single Event Upset at Ground Level," IEEE Trans. on Nuclear Science, Vol. 43, No. 6, pp. 2742-2750, Dec. 1996.
- [23] E. Perelman, G. Hamerly, and B. Calder "Picking Statistically Valid and Early Simulation Points," Proc. of the Intl. Conference on Parallel Architectures and Compilation Techniques, September 2003.
- [24] S. Rusu, H. Muljono, and B. Cherkauer, "Itanium 2 processor 6M: higher frequency and larger L3 cache," IEEE Micro, pp. 10-18, Vol. 24, Issue 2, Mar-Apr 2004.
- [25] T.J. Slegel, E. Pfeffer, and J.A. MaGee, "The IBM eServer z990 microprocessor," IBM Journal of Research and Development, Vol. 48, No. 3/4, pp. 295-310, April 2004.
- [26] J.C. Smolens, B.T. Gold, J. Kim, B. Falsafi, J.C. Hoe and A.G. Nowatryk, "Fingerprinting: Bounding Soft-error-detection Latency and Bandwidth," IEEE Micro, Vol. 24, No. 6, pp. 22-29, Nov-Dec 2004.
- [27] SPEC CPU2000 Benchmarks, <http://www.specbench.org/osg/cpu2000>.
- [28] C. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, "Techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor" Proc. of the Intl. Symp. on Computer Architecture (ISCA'04), pp. 264-275, June 2004.
- [29] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Siavassubramaniam, "ICR: In-Cache Replication for Enhancing Data Cache Reliability," Proc. of the Intl. Conference on Dependable Systems and Networks (DSN), pp. 291-300, June 2003.
- [30] Q. Zhou and K. Mohanram, "Transistor Sizing for Radiation Hardening," Proc. International Reliability Physics Symposium (IRPS), pp. 310-315, 2004.
- [31] Y. Zorian, V. A. Vardanian, K. Aleksanyan, K. Amirkhanyan, "Impact of Soft Error Challenge on SoC Design," Proc. of the On-Line Testing Symposium (IOLTS), pp. 63-68, July 2005.