Explaining Cache SER Anomaly Using DUE AVF Measurement

Arijit Biswas, Charles Recchia, Shubhendu S. Mukherjee, Vinod Ambrose, Leo Chan, Aamer Jaleel, Athanasios E. Papathanasiou, Mike Plaster, and Norbert Seifert

Intel Corporation

Abstract

We have discovered that processors can experience a super-linear increase in detected unrecoverable errors (DUE) when the write-back L2 cache is doubled in size. This paper explains how an increase in the cache tag's Architectural Vulnerability Factor or AVF caused such a super-linear increase in the DUE rate. AVF expresses the fraction of faults that become user-visible errors. Our hypothesis is that this increase in AVF is caused by a super-linear increase in "dirty" data residence times in the L2 cache.

Using proton beam irradiation, we measured the DUE rates from the write-back cache tags and analyzed the data to show that our hypothesis holds. We utilized a combination of simulation and measurements to help develop and prove this hypothesis. Our investigation reveals two methods by which dirty line residency causes super-linear increases in the L2 cache tag's AVF. One is a reduction in the miss rates as we move to the larger cache part, resulting in fewer evictions of data required for architecturally correct execution. The second is the occurrence of strided cache access patterns, which cause a significant increase in the "dirty" residency times of cache lines without increasing the cache miss rate.

1. Introduction

Soft errors continue to pose a challenge for microprocessor designers. These errors arise from bit flips caused by alpha particles from packaging material or atmospheric neutrons [19]. To meet the soft error rate (SER) requirement of a target market segment, designers add an appropriate amount of error protection to a processor. To gauge how these protection schemes can reduce a processor's SER, practitioners have evolved a set of modeling and measurement methodologies. A soft error can be modeled either as a silent data corruption (SDC) or a detected unrecoverable error (DUE) event [12]. An SDC event, as the name suggests, causes data corruption, whereas a DUE event usually leads to a system halt, but does not cause any data corruption. The SDC error rate of a processor, given in *Equation 1* below, is the product of two independent components—

$$\text{SDC SER} = \sum_{\text{all unprotected circuits}} (\text{SDC AVF} \times \text{Intrinsic Error Rate of the Circuit})$$

Equation 1. SDC Equation

$$\text{DUE SER} = \sum_{\text{all protected, uncorrectable circuits}} (\text{DUE AVF} \times \text{Intrinsic Error Rate of the Circuit})$$

Equation 2. DUE Equation

SDC AVF and Intrinsic Error Rate of the Circuit—summed over all circuits that have no appropriate protection[1], where AVF or Architectural Vulnerability Factor expresses the fraction of faults that results in user-visible errors. Similarly, the DUE rate of a processor, given in *Equation 2* [14], is the product of the DUE AVF and Intrinsic Error Rate of the Circuit summed over all circuits on the chip that are protected with an error detection scheme, such as parity. .

Unlike measuring performance or power, however, measuring a processor's SDC or DUE rate is significantly more challenging. Modern CPUs are known to exhibit Mean Time to Failures (MTTFs) of the order of hundreds of years [4]. In this study we use accelerated proton beam testing to accumulate a sufficient number of errors in a reasonable amount of testing time. Even with highly accelerated beam fluxes, numerous repetitions are necessary in product level radiation testing since DUE events by definition stop the normal operation of the system and force a reboot into a clean initial state [2]. A processor's DUE is much easier to measure in this way since the processor typically will have a machine check logged when a DUE event happens. But so far, SDC rates have proven very hard to measure correctly in a complex computing system.

[1] Appropriate protection refers to error detection. No error detection capabilities will result in SDC. In some cases, error detection may be limited in which case SDC may still result in certain situations (triple bit error on a cache line protected by double-bit error detection).

Measuring the AVF of a processor or processor structures has proven even more challenging than measuring a processor's SDC & DUE rates. This is because processors today have no support to inject random faults easily into arbitrary processor structures and observe their effect, which is needed to figure out what fraction of faults will become visible to a user, and hence, result in an error. Li, et al. [9] proposed hardware changes to compute AVFs, but to the best of our knowledge, such mechanisms have not been implemented yet. Measuring the AVF via neutron or alpha particle exposure has also proven difficult because the intrinsic SER of circuits vary widely across processor structures and devices. Sanda, et al. [18] attempted to compute the overall SDC AVF of the Power6™ processor using a combination of simulation and neutron beam experiments. We discuss these and other previous works in more detail in section 7.

This paper examines how to compute the relative DUE AVF of two multi-megabyte level-2 (L2) processor caches using a proton beam to explain an SER anomaly. We discovered that processors can experience a super-linear increase in DUE and correctable ECC (error correcting codes) rate when the size of the L2 cache is doubled. The DUE is triggered by the tag parity. ECC corrections are triggered by the data ECC. Conventional wisdom would suggest that when the cache size is increased by 2x, we should only see a 2x increase in the SER since the per-bit AVF is expected to remain constant. Instead, an error rate of 4x or more was observed in our experiments.

During our investigation, we uncovered several facts:

(1) The super-linear increase in DUE was not observed for all workloads when the cache size was doubled.

(2) Workloads and sites can vary widely across instances in which this can occur

(3) This phenomenon is independent of a particular processor type or pipeline

(4) This was not due to a design or manufacturing difference.

It was critical for us to explain this anomalous behavior, even though the absolute error rate in the larger cache was very small. There are three reasons why we need an explanation. First, the super-linear increase in DUE rate can be a cause of serious concern. Second, the increase in ECC-corrected errors in the data cache is also a cause of concern because many use notification of ECC corrections as an indicator of early failures caused by flaky hardware. In future, even if the parity on the tags is converted to ECC, we will still have the issue of increased ECC errors, so we would certainly need to explain why that happened. Third, we need to ascertain what design changes may be needed to fix this problem.

L2 DUE AVF =
L2 Tag DUE / (L2 Tag FIT/bit x Number of L2 Tag bits) =
Processor DUE / (L2 Tag FIT/bit x Number of L2 Tag bits)

Equation 3. Beam experiment DUE AVF equation for L2 Cache

2x L2 DUE AVF / 1x L2 DUE AVF =
(2x L2 Processor DUE / 1x L2 Processor DUE) / 2

Equation 4. DUE AVF Ratio equation for 2x:1x L2 cache sizes

Interestingly, we had observed super-linear increases in the SER from doubling the size of small write-back caches in our simulation models. This occurred because, while the average AVF across workloads showed little sensitivity to size changes, individual application AVFs could vary widely. The residency time of data that needs to be correct—also known as data that is ACE or required for Architecturally Correction Execution—went up significantly, thereby increasing the AVF. This happened due to an increase in ACE residency times of "dirty", or modified, cache lines caused by fewer early write-backs for the larger cache.

Showing these effects for large multi-megabyte caches was nevertheless a difficult proposition because our industrial-strength detailed performance simulators could not simulate a workload long enough to see the impact of this AVF increase. Cai et al. [4] did observe the same effect by performing cycle-by-cycle simulation of large multi-megabyte caches with an embedded ARM processor. But Cai et al. did not explore in detail why the super-linear effect was observed. In any case, even if we did simulate this effect, without actual measurement we would not be able to conclude with high confidence that this was the root cause of the DUE increase.

Fortunately, because of the nature of our specific problem, we were able to use a proton beam combined with simulation and measurements to establish the root cause of this super-linear SER increase. It turned out that almost all the DUE for the processors in question arose from the parity-protected tags of the write-back L2 cache. This cache signaled a DUE event on any parity error on a "dirty" cache line, thus allowing us to focus primarily on the "dirty" cache line hypothesis by which the ACE residency increases. Consequently, the processor DUE computed through a beam experiment would, in effect, be the DUE of the L2 cache itself. Thus, we have *Equation 3* where FIT stands for Failures in Time (1 FIT = 1 failure in 1 billion hours). Conse-

quently, the DUE AVF of 2x L2 / DUE AVF of 1x L2 is given by *Equation 4*, where 2x L2 refers to the L2 cache with twice the size of the 1x L2 cache. Dividing by 2 accounts for the increased number of tag parity bits in the processor with the 2x larger L2 cache size.

Our results with the benchmarks *libquantum*, *art*, and *swim*—all from the SPEC suite—confirmed our hypothesis. First, using a standalone cache simulator based on PIN [10], we determined that *libquantum* and *swim* showed no change in miss rate, but *art's* miss rate decreased by 6x when the cache size increased from 1 MB to 2 MB. True to our hypothesis, we did not see any change in DUE rates for *libquantum* between a processor with a 1 MB L2 cache and one with a 2 MB L2 cache. But, the DUE rate went up by 4.25x for *art* under the beam between the 1 MB L2 cache processor and the 2 MB one, establishing that the AVF indeed went up by 2.125x[2]. *Swim*, however, posed an anomaly to our initial conjecture. Although *swim's* cache miss rate did not change, the DUE rate increased by 3.87x. We determined that this is still due to the increase in residency of "dirty" data in the processor L2 cache. But, this increase happens in *swim* due to the strided L2 cache access pattern causing dirty data to reside significantly longer in a 2x larger L2 cache.

Our results have two critical implications. First, we need to be careful about how we ascertain the SDC and DUE AVFs of a processor. When we increase the size of a structure—either due to a design change or because we are designing a new processor—we cannot blindly assume that the AVF per bit remains constant. Second, we have to worry about how to flush the L2 cache and other structures, so that we can reduce the "dirty" data residency time, thereby reducing the SDC or DUE rates, as the case may be.

The remainder of this paper is organized as follows. Section 2 describes the steps, tools and methodologies we used during our investigation into the cause of the super-linear cache SER increase and details both our simulation and system measurement models and methodologies. Section 3 explains how we used our models to develop a working hypothesis to explain how increases in AVF can occur when the cache size increases. Section 4 describes the method by which we chose our benchmarks for system measurement in order to clearly prove or disprove our hypothesis. Section 5 details the actual system measurement experiments and their results. Section 0 provides a detailed analysis of the measurement results and provides insight into how these results relate to our initial hypothesis. Section 7 provides an overview of related work in this area while section 8 provides a discussion on what can be done to reduce or eliminate super-linearly increased SER due to increased cache sizes. Finally, section 9 provides our final thoughts and conclusions.

[2] Note that the AVF goes up by 2.125 = 4.25 / 2. The division by 2 is to account for the twice the number of bits in our 2x L2.

Table 1. Experimental System Configuration

CPU	Processor 1x	Processor 2x
Number of CPUs	2	
Frequency	2.8 GHz	3.0 GHz
Trace Cache	12 Kuops	
L1 Data Cache	16 KB	
L2 Cache	1 MB	2 MB
Mfg. Process	Same	
System Memory	2 GB DDR2	
Memory Speed	200 MHz	
Chipset	Intel E7520 Rev. C4	
Southbridge	Intel 82801 EB (ICH5)	
Hard Drive	76.3 GB	

2. Steps, Methodology, & Tools

This section describes the steps, methodology, and tools with which we developed our hypothesis to explain the cache SER anomaly. The steps we took are as follows.

- We investigated and eliminated any possible source of super-linear SER increase other than a significantly increased AVF.
- We used our detailed performance simulator to demonstrate that significant AVF increases were possible when a structure's size is doubled.
- We selected two specific processors with the same core but with different cache configurations—one with 1 MB L2 and other with 2 MB L2—to be our target processors for proton beam testing. We refer to these processors as Processor 1x and Processor 2x to signify the differences in their L2 cache sizes.
- We used our high-level non-timing cache simulator to identify benchmarks that exhibited the proper cache behavior profiles for our target multi-megabyte cache sizes. Specifically, these benchmarks were *art* and *swim* from the SPEC 2000 suite and *libquantum* from the SPEC 2006 suite.
- We took our target processors and placed them under the proton beam while running our target

benchmarks. The target systems were configured as shown in *Table 1* and were running the 32-bit version of Microsoft Windows Server 2003 operating system.

- Finally, we analyzed the results and corroborated our simulation expectations with performance counter data obtained using the Vtune™ performance counter monitoring software.

The rest of this section describes the AVF simulator (Section 2.1), non-timing cache simulator (Section 2.2), and our proton beam experimental set up (Section 2.3).

2.1 Modeling AVF

To prove our hypothesis that doubling the size of a structure can cause a super-linear increase in AVF, we used a detailed performance simulator that models a Core Duo™-like processor in the Asim performance modeling infrastructure [6] and is augmented with the AVF instrumentation. The AVF instrumentation is based on two concepts. The first was the concept of ACE (required for architecturally correct execution) and un-ACE (not required for ACE) [13]. AVF for a bit is then defined as the ratio of the total time a bit is in ACE state and the total simulation time. The second concept was the use of Hamming-distance-one analysis, which allows us to compute the AVF of address-based structures [3]. Using this AVF-instrumented performance model, we simulated over 700 benchmark traces. The traces were generated as representative regions of the given benchmarks based on PinPoints [16].

2.2 Modeling Cache Misses

To prove our hypothesis that residence time in the cache can increase dramatically, thereby causing a significant increase in AVF, we needed a cache simulator that could give us the miss rate of selected benchmarks for large multi-megabyte caches. For this analysis, we used CMP$im [8]. While the detailed performance model referred to in Section 2.1 was much faster than a low-level register transfer level model, it was neither fast enough to run entire benchmarks to completion nor capable of effectively modeling cache AVFs for caches greater than 256 KB.

CMP$im contained no timing information or micro-architectural details and executed only static instruction traces allowing for significantly faster simulations. This allowed us to simulate entire benchmarks with multi-megabyte caches. While we could not compute the cache AVF from such a model, it did give us the cache miss profiles (e.g., miss rate for different benchmarks for different cache sizes) from which we could reason about our hypothesis and develop our experiment.

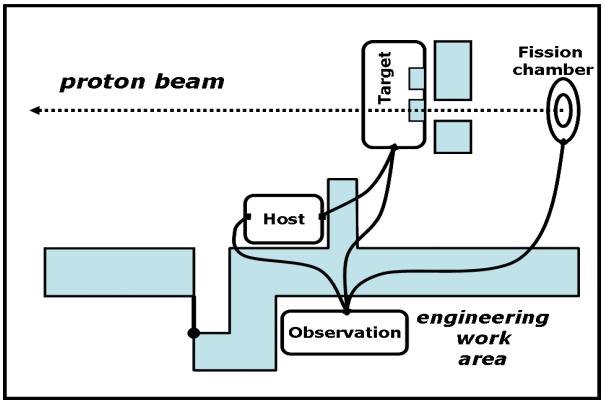

Figure 1. Configuration for proton beam experiment

2.3 Measuring SER with protons

Once we had developed our hypothesis and selected the specific benchmarks based on their cache miss profiles, we had to irradiate the processors running the specific benchmarks under the proton beam. We used industry standard practices in measuring the SER of a chip arising from atmospheric neutrons using an accelerated proton beam, as described in the JEDEC standard [1], Hiemstra and Baril [7], and Sanda, et al. [18]. The experimental process consists of the following steps:

- Booting the system under test
- Continuously looping the benchmark of choice
- Opening the shutter for the particle beam
- Focusing the beam on the chip of interest (*Figure 1*)
- Counting the particle fluence that impinges on the chip until system fails
- Capturing logs of the failure behavior

The beam diameter is selected to be large enough to irradiate the entire silicon chip—in our case the microprocessor under test—but not any other silicon component in the system. Also, we needed to only compute the DUE rate arising from the processor. Hence, we only counted those failures that resulted in a processor machine check architecture log signaling a DUE error. Any run that did not give rise to a processor machine check log was discarded.

We chose the Francis H Burr Proton Therapy Center proton beam in Boston, Massachusetts due to both beam availability and achievable flux considerations. The proton beam available at this center is a mono-energetic 148 MeV proton beam, which is neither a neutron beam nor does it mimic the energy spectrum of atmospheric

neutrons like the beam at LANSCE in Los Alamos. The LANSCE beam is typically used to compute the SER of a chip (after scaling down the flux to reflect the neutron flux rate at a specific altitude). Nevertheless, the use of this mono-energetic beam was appropriate for our experiments. The high-energy proton beam appropriately mimics the behavior of neutrons at this higher energy [19]. The additional Coulombic charge interaction induced by protons is negligible compared to the nuclear forces generated by the high-energy protons [14].

Further, we are only interested in the relative DUE AVFs of two silicon chips in the same process technology. The FIT/bit of the L2 cache tags in both chips resulting from the nuclear interaction of the mono-energetic beam will be the same. Given that AVF itself is independent of the FIT/bit, we can still compute the relative DUE AVF as described in *Equation 4*.

To compute each processor's DUE Mean Time to Failure (MTTF) rate under the mono-energetic proton beam, we must conduct an appropriate number of experiments to ensure statistical significance. For each experimental run we exposed the same chip to the beam until we observe a user-visible error with a corresponding machine check log from the processor signifying a DUE error. This gave us a Time to Failure (TTF) for each experiment. We excluded TTFs that resulted in no machine check logs as these were incidences of SDC that resulted in system crashes and we were only interested in DUE for this study. The run-to-run variation in the measured TTF's for the same chip was large, as expected, given the exponential distribution due to the random nature of the process. We gathered 10 TTF numbers for each chip, which enabled us to detect a DUE MTTF ratio of 4 with 90% confidence, and with greater confidence on MTTF ratios that measured greater than 4. We will also show in Section 5 that 10 runs are sufficient for our results to converge.

3. Developing the Hypothesis

The first step in identifying the cause of the super-linear SER increase was to eliminate possible causes so that what remains is the most likely cause of this behavior. The principle we followed is known as Strong Inference as described by Platt in his landmark paper [17]. Recall from *Equation 2* that DUE rate of a circuit = DUE AVF of the circuit x intrinsic error rate of the circuit. Thus, either the intrinsic error rate had increased without our knowledge or the AVF had increased for some reason.

3.1 Identifying the Cause

We ruled out any external or platform-level issues, such as faulty power supplies or load lines. The remaining internal possibilities can be divided into four categories: electrical, design, manufacturing and AVF. By systematically analyzing the processor chip, we ruled out electrical problems, such as internal power delivery issues, increased RC delay on the cache bit-lines and word-lines due to the larger cache size, etc. We also determined that there were no design-related issues, such as a different or defective SRAM cache cell. We also eliminated any manufacturing issues, such as defective mask design or different process parameters, to be the cause of this problem. This left us with the only likely root cause—that of increased AVF in the larger cache. Proving that AVF was the most likely cause of this problem was no simple feat and is the topic of this paper.

3.2 Identifying the Mechanisms

We had some indication from our AVF-instrumented performance model that significant increases in AVF could arise for the same workload when we increased the size of a structure. For example, when changing the size of a write-back cache from 32K to 64K, some benchmark simulations showed as much as a 100x increase in the AVF of the cache data and tags**.** In yet another example, we observed an AVF increase of about 13x for the data TLB (translation lookaside buffer) for some benchmarks when we increased its size from 64 entries to 128 entries.

The underlying cause of this AVF increase is an increase in ACE data residency time. This ACE residency time in the bigger structure (e.g., cache, TLB, etc) can increase significantly if the working set for a given workload fits into the bigger structure but not into the smaller structure, thereby causing significantly higher miss rates in the smaller structure.

In the case of a write-back cache, there were two distinct mechanisms by which ACE residency time could increase. The first involved increased cache misses in the smaller cache that evicted "clean" or unmodified cache lines, which rendered ACE time un-ACE. The second involved increased cache misses in the smaller cache that evicted "dirty" or modified cache lines, which forced "dirty" data to be written back sooner. Next, we consider each of these cases in detail.

The first case is that increased cache misses in the smaller cache caused the ACE time of "clean" cache lines to become un-ACE. *Figure 2* shows 2 cache lines, one from a small cache (referred to as 1x cache from here onwards) and one from a 2x larger cache (referred to as 2x cache from here onwards). In the 2x cache

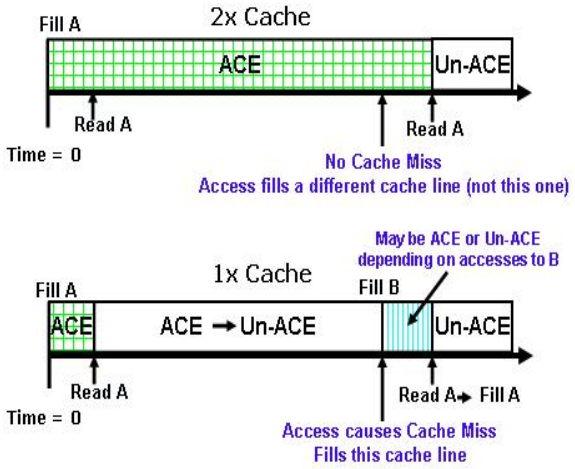

Figure 2. "Clean" cache line scenario

case, we see a fill followed by 2 reads. Assuming that all accesses are made by ACE instructions, the cache line for the 2x cache is ACE from the fill to the last read. In the 1x cache however, we get another cache access to an address which maps to this same cache line (this same access maps to a different cache line in the 2x cache since there are 2x more lines). This access occurs between the 2 read accesses to line A. This cache miss forces an eviction and fill in the 1x cache and, as a result, renders the time from the first read to the eviction un-ACE. The time from the fill of address B to the second read of address A (which misses and re-fills the 1x cache with the old data) could be ACE or un-ACE depending on the accesses to B during this time., Thus, this example demonstrates that the AVF for this cache line can be significantly greater for the 2x cache than the 1x cache.

The second case is specific to a write-back cache. Recall that a write-back L2 cache stores processor writes in the L2 cache in modified or "dirty" state and does not write them back to main memory immediately. In this scenario, depicted in *Figure 3*, we again have 2 cache lines, one belonging to a 2x cache and one to a 1x cache. In this case, there is a fill followed by a write in both caches. After the write, the cache line becomes "dirty" and is ACE from the write until it is written back to a higher level of cache or to main memory. In the 1x cache we again see a cache miss that forces an eviction of the cache line. However, since the line is "dirty" this will result in the data being written back at this time. In the 2x cache the "dirty" line continues to reside in the cache until the end of the program when the cache lines are flushed. As a result, the ACE time and thus the AVF for the "dirty" cache line is significantly larger for the 2x cache than the 1x cache.

3.3 Applying Our Hypothesis to the Target Cache

In our specific case, the cache SER anomaly was observed in both the write-back L2 cache tags (as correctable machine check logs or DUE) as well as the L2 cache data (as logged ECC corrections). The data portion of the cache was protected with ECC, so a bit flip in the data array resulted in an ECC correction, but not a DUE event. In this case, single-bit errors in both "clean" and "dirty" cache lines were correctable.

The tags however were protected only with parity, so a bit flip in the tag array for a "dirty" cache line resulted in a DUE event. Tag parity errors on "clean" cache lines were corrected automatically in the design by invalidating the corrupt cache line. An error in a "dirty" cache line could not be corrected via the same mechanism since the specific cache line had the most up-to-date copy. Thus, a fault in the L2 tag array will only result in an actual error in the case depicted in *Figure 3* since any fault on a "clean" line (*Figure 2*) is recoverable via invalidation.

Interestingly, however, the ACE → un-ACE conversion in the 1x cache (*Figure 3*) can be due to recovery of "clean" cache lines. In other words, if "dirty" data is evicted and replaced by "clean" data in the 1x cache, then the clean cache line ends up holding un-ACE data, instead of ACE causing the DUE AVF to go down.

As we can see, both "clean" and "dirty" data can cause super-linear increases in SER in the L2 cache tag

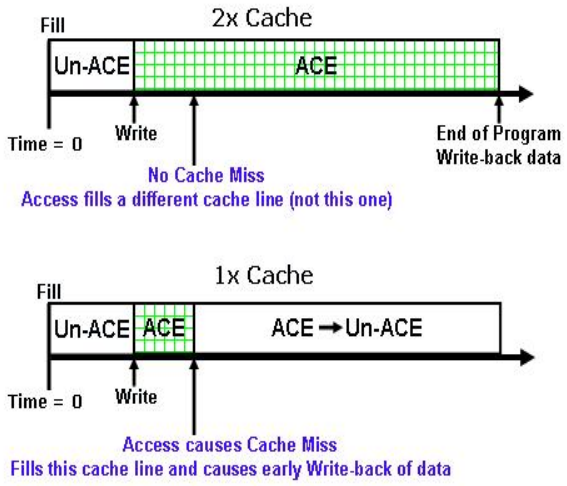

Figure 3. "Dirty" cache line scenario

and data arrays. For the rest of this paper, however, we only focus on the cache tag DUE rates because our proton beam experiments focused on explicitly measuring the DUE rate of the processors under test.

4. Choosing the Benchmarks

As explained in the previous section, we expected the L2 cache tag DUE AVF to increase if the "dirty" residency times increased. One way to indirectly determine this is to observe whether the cache miss rate drops significantly when the cache size is doubled. Specifically, the two processors we were irradiating under a proton beam had 1MB and 2MB L2 cache sizes, so we needed to find benchmarks whose working set did not fit in the 1MB cache, but did fit in the 2MB cache. This should help ensure that the 1x cache has a higher miss rate than the 2x cache. We also needed benchmarks whose working set did not fit into either a 1 MB or a 2 MB cache. Using CMP$im, we generated cache miss profiles for various SPEC benchmarks for a 1 MB and 2MB L2 cache. From these runs we chose 3 specific benchmarks that had the desired profiles: *art* and *swim* from SPEC2000 and *libquantum* from SPEC2006.

Figure 4 shows the miss profiles for these three benchmarks. Number of misses is shown on the X-axis while the Y-axis shows the various cache sizes that we simulated. The thick vertical lines indicate the 2 target cache sizes in each case (1 MB and 2MB). The miss profile for *art* indicates a significant reduction (~6x) in the number of misses when going from a 1MB to a 2MB cache. Thus, we would expect that *art* should show a super-linear SER for the 2MB part than the 1MB part if our hypothesis holds. The miss profiles for *libquantum* and *swim* indicate that there is no change in the number of misses between a 1MB and a 2MB cache. If our hypothesis holds, we do not expect a super-linear SER increase for these two benchmarks.

5. Conducting the Experiment

We took the two target processors—one with 1MB L2 cache and one with 2MB L2 cache in systems configured as per *Table 1*—running the three benchmarks—*art*, *swim*, and *libquantum*—to the Francis H Burr Proton Therapy Center at Massachusetts General Hospital in Boston. *Table 2* shows the results of the proton experiments for the three target benchmarks. As expected, *art* showed a super-linear increase in DUE of 4.25x between the 1 MB and 2 MB parts. Similarly, as expected, the proton measurements for *libquantum* showed a DUE increase of 1.2x from 1 MB to 2 MB representing a small 20% increase. The proton measurements for *swim*, however, indicated a 3.87x super-linear increase in the SER rate, which was contrary to

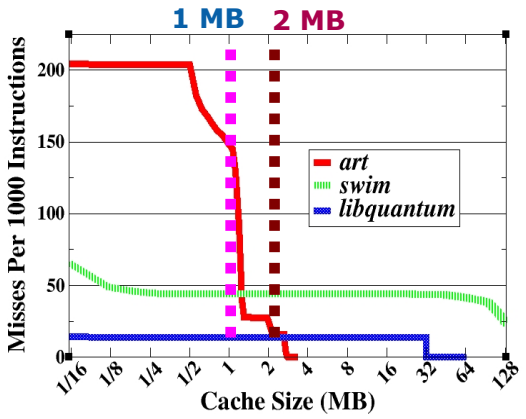

Figure 4. Cache miss profiles for *art*, *swim*, and *libquantum*.

our expectation since *swim*'s cache miss rate did not change between 1 MB and 2 MB L2 caches. We will analyze these results in detail in Section 0.

To achieve statistical significance for these numbers, we ran 10 repetitions of proton beam experiments for each of the 3 benchmarks with each of the 2 processors (30 runs total per processor). This sample size enabled us to detect a DUE MTTF ratio of 4 with 90% confidence, and with greater confidence on MTTF ratios larger than 4.

Figure 5 shows this effect graphically. Since soft errors arise from a random process, there is no correlation observed between the duration of subsequent trials. A graphical demonstration of how confidence intervals for MTTF evolve as more data is collected can be obtained by randomizing the order in which the trial runs were collected, and computing MTTF as a function of number of runs for each of the randomized sets. *Figure 5* shows how we achieved increased statistical confidence for *art* through the funnel-shaped plot. *Figure 5* also shows the importance of collecting the appropriate number of data points. An inappropriately small number of data points can easily lead to incorrect conclusions.

6. Analyzing the Results

To explain the DUE ratios given in *Table 2*, we obtained L2 cache-related metrics, shown in *Table 3*. We used Intel's Vtune™ performance monitoring software to poll the hardware performance counters and configured the counters to count *dirty reads*, *read misses,* and *write-backs*. The simulation data provided *misses per thousand instructions (MPKI)* and *write misses*. This data is given in *Table 3* in the form of 2MB:1MB ratios. A ratio less than 1 represents a

decrease from 1 MB to 2 MB while a number greater than 1 represents an increase. A ratio of 1 represents no change in that statistic between the two processors. Armed with this data, we analyzed the proton beam results in the following subsections.

Note that our goal was not to exactly match the numbers or predict the exact increase in DUE or AVF. Rather, we are only trying to show that AVF can increase super-linearly when the cache size is doubled.

6.1 *Art* and *libquantum*

As expected from simulations, *art* showed a super-linear increase in DUE of 4.25x which was further verified by the Vtune™ data and simulation results. The simulation statistics for *art* from *Table 3* indicate that misses decreased by 6x. More specifically, write misses decreased by 4x when going from 1 MB to 2 MB, which helped increase the "dirty" data residence time in the 2 MB cache. Similarly, Vtune™ data showed that read misses decreased by 3.17x, write-backs decreased by 32.75x and reads to "dirty" cache lines increased by 1.56x from 1 MB to 2 MB. The "dirty" read increase and the significant decrease in write-backs are clear indications that the ACE "dirty" data residency time did increase super-linearly, thereby increasing the AVF in the same way.

Libquantum beam testing results showed a DUE increase of only 1.2x, which is not super-linear, as we expected. The simulation and Vtune™ data for *libquantum* in *Table 3* further confirm this hypothesis. All the statistics remained very close to 1 indicating little to no change in *libquantum's* executions across the two cache sizes.

6.2 The *swim* Conundrum

The proton measurements for *swim* indicated a 3.87x super-linear increase in the DUE rate, but no change in cache miss rates between the 1 and 2 MB L2 caches. This was superficially contradictory to our hypothesis that the DUE went up super-linearly because the "dirty" data (and, hence ACE data) residency time in the cache increased.

Table 2. Accelerated proton beam measurement results for *art*, *swim*, and *libquantum*

Benchmark	DUE SER Ratio (Processor 2x / Processor 1x)
Art	4.25
Swim	3.87
Libquantum	1.2

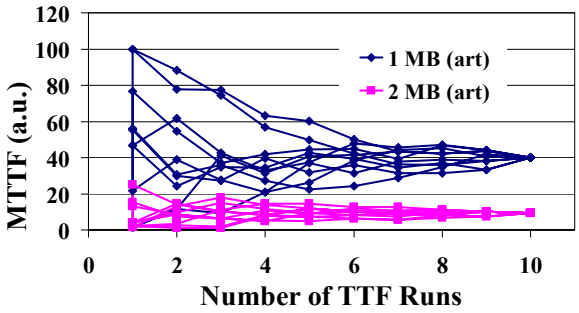

Figure 5. Cumulative TTF runs vs. MTTF (a.u. = arbitrary unit)

Since the cache miss profile as well as the Vtune™ and other simulation statistics from *Table 3* were so similar between *libquantum* and *swim*, we looked for any fundamental differences between the cache access behaviors for these two benchmarks. One difference was in the actual computations performed by each benchmark. *Swim's* main loop is primarily a large matrix operation while *libquantum's* computation is far more irregular. *Swim* is a shallow water wave modeling benchmark which performs a large floating point matrix operation over which it iterates. *Libquantum,* by contrast, is a quantum computer simulation benchmark.

As a result, one might expect *swim* to have a more regular, strided [11] cache access pattern than *libquantum*. *Figure 6* shows data generated from our high-level cache simulation model clearly showing that *swim* has both a regular access pattern (caused by strided accesses) as well as a very regular pattern of "dirty" residency while *libquantum* has far more random cache behavior. In this figure, there are two graphs for each benchmark. The X-axis is time for both graphs. The Y-axis is "Accesses per 1000" for the top graph, which shows how many cache accesses (loads and stores) occurred at a given point in time, and "% Dirty Lines"

Table 3. Simulation and Vtune™ performance counter data for *art*, *swim* and *libquantum*

*Vtune™ **Simulation	***Art*** Ratio: 2MB/1MB	***Libquantum*** Ratio: 2MB/1MB	***Swim*** Ratio: 2MB/1MB
MPKI**	1/6.00	1	1
Dirty Reads*	1.56	1.06	0.98
Write Misses**	1/4.00	1	1
Read Misses*	1/3.17	1/1.14	1/1.16
Write-backs*	1/32.75	1/0.99	1.00

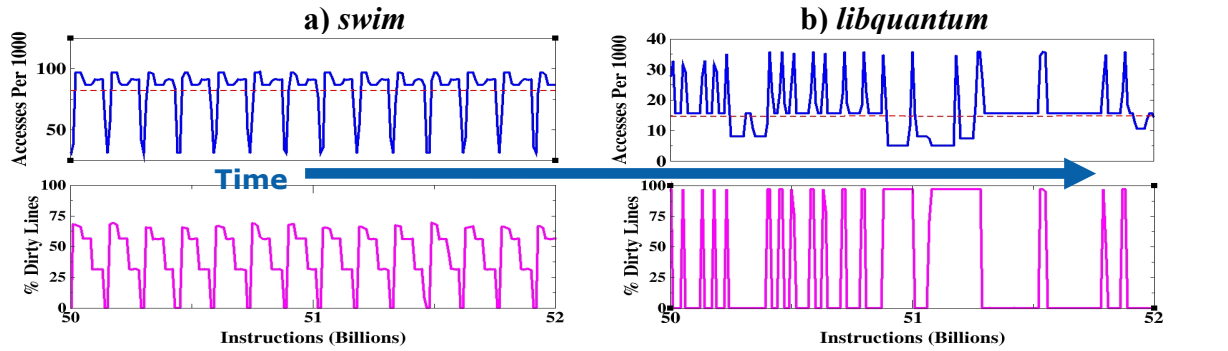

Figure 6. Cache access and "dirty" line utilization patterns for swim and libquantum

for the bottom graph, which is an indication of "dirty" line utilization (what percentage of the cache contains "dirty" data) at a given point in time.

We developed an example to determine whether a strided cache access pattern can cause increased "dirty" residency times while not affecting the number of misses when increasing the cache size. In this example, we assume 2 caches, one with 4 entries (1x cache) and the other with 8 entries (2x cache). We assume a perfectly strided access, meaning that the cache accesses will walk through the entries of the cache. In the simplest case, we assume a single strided write occurs at some regular interval. In this case, once we have simulated for a sufficient period of time, both caches will simply have "dirty" entries all the time and the AVF is 100% for both caches. Hence, the SER rate will increase linearly with cache size while the number of misses will remain constant since all writes miss and the number of write-backs will remain constant since all "dirty" data will be written back at the end of the program.

However, different mixes of reads and writes can result in a significant increase in the AVF for strided cache accesses. For instance, if we take the cache size and stride assumptions from the simple example and change the strided access pattern to be 2 reads followed by 1 write, we see the following behavior as detailed in Figure 8.

- At Time=0 we read entry 1 on both caches.
- At Time=1 we read entry 2 on both caches.
- At Time=2 we write entry 1 on both caches. Assuming the caches started with all entries invalid, we now have 2 read misses and 1 write hit for both caches. The write causes the first entry of both caches to become "dirty" and so will begin to accrue "dirty" residency time until it is written back.
- We continue in this fashion until Time=6
- At Time=6 the first read wraps around to the first entry in the 1x cache, invalidating it and forcing it to write-back. However, the first entry in the 2x cache continues to be "dirty" since it will not wrap around for some time yet. At this point in our example, we begin to see the dirty residency times diverge between the two caches while the miss counts remain the same.
- At Time=11, the 2x cache has accumulated a "dirty" residency time almost 3x higher than that of the 1x cache, yet the miss counts remain roughly the same.
- At the end of the program, all dirty lines will be written back so the number of write-backs will be the same.

Here we have shown that a strided access pattern can indeed result in a super-linear increase in the "dirty" residency time of caches. Different access patterns of reads and writes can change the rate of this super-linear increase. However, the number of misses, "dirty" reads, and write-backs can be held very nearly constant between the two caches. This explains how the *swim* benchmark can show a super-linear increase in SER while the indirect indicators of cache line residency continue to show no differences between the 2 cache sizes. In this case, *swim's* strided cache access pattern which has a 4:1 ratio of loads to stores, coupled with the fact that its data set does not fit cleanly into either cache, results in a significant increase in "dirty" cache line residency times. This in turn leads to the significant increase in AVF that causes the 3.87x increase in SER from the 1 MB to the 2 MB cache.

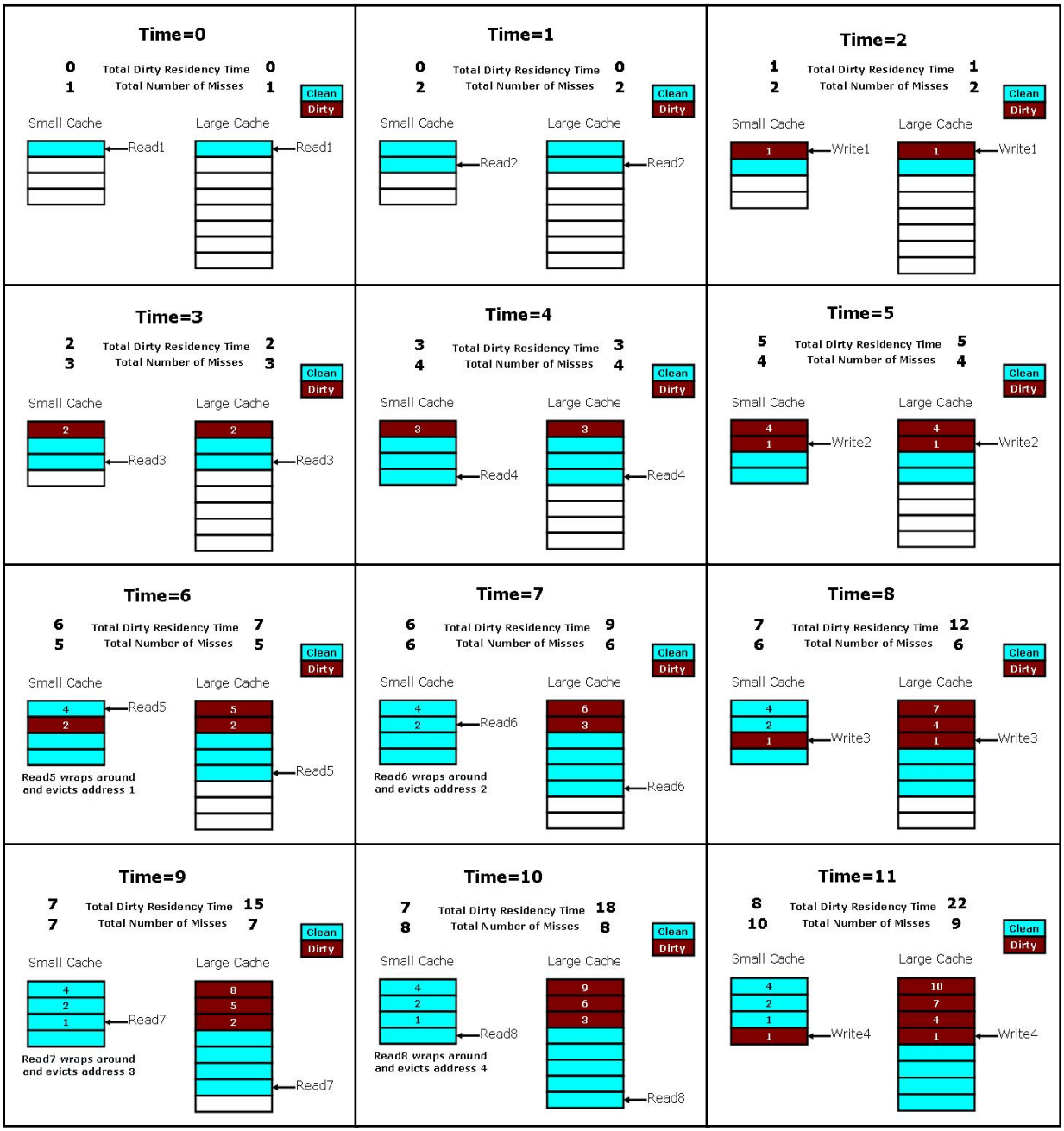

Figure 7. Example of super-linear "dirty" residency increase for strided cache access

6.3 Summary

In order to determine the root cause of the super-linear cache SER when cache sizes double, we employed numerous methods. We started by enumerating all the possible sources of this increase and eliminating all but AVF related effects. Using simulation data we convinced ourselves that size changes in cache arrays could indeed result in significant changes in AVF when running the same workload. Using the simulation studies we formulated the hypothesis that increases in "dirty" cache line residency times were responsible for the cache tag AVF increase. We employed a high-level cache model to choose benchmarks that we could use for accelerated proton beam testing on actual processors.

Armed with these benchmarks, we took two processors with 1 MB and 2 MB of L2 cache respectively and irradiated them using a proton beam while running several iterations of each of the three chosen benchmarks.

The proton test results on all three chosen benchmarks reinforced the hypothesis that increases in "dirty" residency times of cache lines led to the super-linear DUE SER increase. We identified two mechanisms by which this happens: longer residency times due to lower miss rates such as in *art* (with the opposite effect seen in *libquantum*), and strided cache access patterns causing increased "dirty" residency times such as in *swim*.

7. Related work

In this section, we review some related work in the areas of AVF sensitivity to structure sizes and SER measurements. Both Biswas et al. [3] and Cai et al. [4] noticed that the AVF of a single workload can vary significantly as structure sizes change. However neither explored in detail why or how this happens. Biswas, et al. noticed that, even though the AVF of a single workload can vary due to structure size, the average AVF across numerous workloads tends to stay fairly constant across structure sizes. They did not pursue the matter any further than that observation. Cai et al. noted that a single workload AVF can increase as a structure's size increases due to the working set size fitting into the structure. They did not explain the mechanism by which this occurs. Additionally, neither Biswas et al. nor Cai et al. attempted to measure the error rates of actual systems to corroborate their simulation data.

Recently, Sanda, et al. [18] described their methodology and results for SDC measurement of the Power6™ processor using accelerated beam testing. There were two main differences between Sanda, et al.'s work and the work presented here. First, Sanda, et al.'s measurements targeted SDC, not DUE as in our study. DUE errors are easier to measure since they result in a machine check log, whereas SDC errors do not result in machine check logs and it can take a long time until the error becomes user-visible (for instance in the output of a program) and can be caused by a variety of unprotected structures on a chip. Second, Sanda, et al. did not provide any data on how structure size changes affect the error rate.

8. Discussion

We have shown that increases in structure size can indeed result in a non-linear increase in SER caused by increasing AVFs, but what can we do to account for this? Such SER variation can make it very difficult for system users to accurately predict the effect on error rates when migrating to new systems with larger caches.

While such upgrades are easy to characterize for performance and power, characterizing for SER in the face of architectural and workload dependent variation can prove very difficult. In our particular case, one solution would be to protect the L2 cache tags with ECC rather than parity. While this would not eliminate the underlying AVF variability, it would reduce the incidence of cache DUE to nearly zero since all single-bit errors would become correctable.

Another possible solution that addresses the underlying AVF variability is to periodically flush the caches. Biswas, et al. [3] proposed periodic cache flushing as a way to reduce the AVF of structures such as TLBs and caches, showing significant reductions in AVF for a minimal performance loss.

9. Conclusions

We discovered DUE rates in write-back caches with parity-protected tags can increase super-linearly as the cache size doubles. In this paper, we set out to prove that Architectural Vulnerability Factor or AVF increases brought on by increases in residency time of "dirty" cache lines was the root cause of this phenomenon.

We investigated what appeared to be a DUE SER anomaly on the tags for large cache processors using several techniques. We used two different simulation models to develop a hypothesis to explain how AVF could cause this behavior. We then designed an experiment, choosing three specific SPEC benchmarks that exhibited cache miss profiles conducive to proving or disproving our hypothesis and measured the DUE error rates on two processors (with 1 MB and 2 MB of L2 cache, respectively) when running the chosen benchmarks. The results of our accelerated proton beam system measurements along with performance counter data proved that our hypothesis did indeed hold. We saw a 4.25x DUE SER increase for *art*, a 3.87x increase for *swim*, and a 1.2x increase for *libquantum*.

We identified two ways in which the "dirty" residency times could increase. One was by observing the miss rates. As the data set for a workload fits into the larger cache, the relative miss rates decrease dramatically, thereby increasing the AVF and causing super-linear increases in the DUE rate. This mechanism accounted for the super-linear DUE increase measured in *art* as well as the lack of super-linear increase seen in *libquantum*.

The second involved strided cache accesses. A workload that exhibited regular, strided cache access patterns could cause a significant increase in the "dirty"

residency times of the cache without being reflected in either the miss rates or the write-back rates. Nevertheless, this would have the same effect of increasing the cache AVF, resulting in a super-linear increase in the DUE SER. This second mechanism accounted for the super-linear DUE increase observed for *swim*.

We gleaned two important insights from these results. First, we must be careful about how we ascertain the SDC and DUE AVFs of a processor. We cannot simply assume that the AVF per bit stays constant as structure sizes increase across designs. Finally, we must consider ways to reduce the dirty data residency time, such as periodic flushing, thereby reducing the SDC or DUE rates.

10. REFERENCES

[1] "Test Method for Beam Accelerated Soft Error Rate," JEDEC JESD89-3A.

[2] "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices", JEDEC Test Standard No. 89A, Sep. 2006.

[3] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S.S. Mukherjee, R. Rangan, "Computing Architectural Vulnerability Factors for Address-Based Structures", *32nd International Symposium on Computer Architecture (ISCA),* June 2005.

[4] D. Bossen, "CMOS Soft Errors and Server Design", *2002 IEEE Int'l Reliability Physics Symposium, Tutorial Notes—Reliability Fundamentals, IEEE Press*, 2002, pp. 121_07.1–121_07.6.

[5] Y. Cai, M. T. Schmitz, A. Ejlali, B. M. Al-Hashimi, and S. M. Reddy. "Cache Size Selection for Performance, Energy and Reliability of Time-Constrained Systems," *Asia and South Pacific Conference on Design and Automation,* January 2006.

[6] J. Emer, P. Ahuja, N. Binkert, E. Borch, R. Espasa, T. Juan, A. Klauser, C.K. Luk, S. Manne, S.S. Mukherjee, H. Patil, S. Wallace, "Asim: A Performance Model Framework," *IEEE Computer*, 35(2):68-76, Feb. 2002.

[7] D. Hiemstra and A. Baril, "Single Event Upset Characterization of the Pentium® MMX and Pentium® II Microprocessors using Proton Irradiation," *IEEE Transactions on Nuclear Science*, Vol. 46, No. 6, pp. 1453-1460, Dec., 1999.

[8] A. Jaleel, R. S. Cohn, C. K. Luk, and B. Jacob. "CMP$im: A Pin-Based On-The-Fly Multi-Core Cache Simulator". *Workshop on Modeling, Benchmarking and Simulation*, 2008.

[9] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Online Estimation of Architectural Vulnerability Factor for Soft Errors," *Proceedings of 35th International Symposium on Computer Architecture (ISCA '08)*, June 2008.

[10] C. K. Luk, R. Cohn, R. Muth, H. Patil, Artur Klauser, G. Lowney, S. Wallace, V. J. Reddy, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," *ACM SIGPLAN conference on Programming language design and implementation*, 2005.

[11] C. K Luk, R. Muth, H. Patil, G. Lowney, R. Cohn, and R. Weiss, Profile-Guided Post-Link Stride Prefetching," *Proceedings of the 2002 International Conference on Supercomputing (ICS'02)*, pages 167-178, June 2002.

[12] S. Mukherjee, "Architecture Design for Soft Errors," Elsevier, Inc. February, 2008.

[13] S.S. Mukherjee, C.T. Weaver, J. Emer, S.K. Reinhardt, T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," *36th Annual International Symposium on Microarchitecture (MICRO)*, December 2003.

[14] H.T Nguyen, Y. Yagil, N. Seifert, M. Reitsma, "Chip-Level Soft Error Estimation Method", *IEEE Transactions on Device and Materials Reliability*, Volume 5, Issue 3, Sept. 2005, pp. 365 – 381

[15] E. Normand, "Extensions of the Burst Generation Rate Method for Wider Application to Proton/Neutron-Induced Single Event Effects," *IEEE Transactions on Nuclear Science*, Vol. 45, pp. 2904-2914, (1998).

[16] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. "Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation." *37th Annual International Symposium on Microarchitecture (MICRO-37),* December 2004.

[17] J.R. Platt, "Strong Inference," *Science Magazine*, Volume 146, Number 3642, October 1964

[18] P. N. Sanda, J. W. Kellington, P. Kudva, R. Kalla, R. B. McBeth, J. Ackaret, R. Lockwood, J. Schumann, and C. R. Jones, "Soft-error resilience of the IBM POWER6 processor," *IBM Journal of Research and Development, Soft Errors in Circuits and Systems*, Volume 52, Number 3, pp. 275-284 (2008).

[19] N. Seifert, B. Gill, K. Foley, P. Relangi, "Multi-Cell Upset Probabilities of 45nm High-k + Metal Gate SRAM Devices in Terrestrial and Space Environments", *Proceedings of the International Reliability Physics Symposium (IRPS),* pp. 181-186, 2008.

[20] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, J. Maiz, "Radiation-Induced Soft Error Rates of Advanced CMOS Bulk Devices", *Proceedings of the IEEE International Physics Symposium*, pp. 217-225, 2006