Soft Error Benchmarking of L2 Caches with PARMA

Jinho Suh, Mehrtash Manoochehri, Murali Annavaram and Michel Dubois
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles
{jinhosuh, mmanooch, annavara}@usc.edu, dubois@paris.usc.edu

ABSTRACT
The amount of charge stored in an SRAM cell shrinks rapidly with each technology generation thus increasingly exposing caches to soft errors. Benchmarking the FIT rate of caches due to soft errors is critical to evaluate the relative merits of a plethora of protection schemes that are being proposed to protect against soft errors. The benchmarking of cache reliability introduces a unique challenge as compared to internal processor storage structures, such as the load/store queue. In the case of internal processor structures the time a data bit resides in the structure is so short that it is generally safe to assume that no more than one soft error strike can occur. Thus the reliability of such structures is overwhelmingly dominated by single bit errors. By contrast, a memory block may reside for millions of cycles in a last level cache. In this case it is important to consider the impact of the spatial and temporal distribution of multiple errors within the lifetime of a cache block in the presence of error protection.

This paper introduces a unified reliability benchmarking framework called PARMA (Precise Analytical Reliability Model for Architecture). PARMA is a rigorous analytical framework that accurately accounts for the distribution of multiple errors to measure the failure rate under any protection scheme. In a single simulation run PARMA provides a precise FIT rate (expected number of failures in one billion hours) measurement for storage structures where the effect of multiple errors cannot be neglected. We have implemented the PARMA framework on top of a cycle-accurate out-of-order processor simulator (sim-outorder) to benchmark L2 cache failure rates for a set of CPU 2000 benchmarks. The effectiveness of three protection schemes are compared in terms of L2 cache FIT rate: parity, word-level Single Error Correcting Double Error Detecting (SECDED) code and block-level SECDED.

Exploiting the accuracy of PARMA, we demonstrate that current techniques to evaluate cache FIT rates in the presence of SECDED, such as accelerated fault injection simulations and first-principle derivations based on Architectural Vulnerability Factor (AVF), can overestimate FIT rates by vast amounts. Based on the insights gained during this research we also introduce a new approximate analytical model that can quickly and more accurately estimate cache FIT rate in the presence of SECDED.

SIGMETRICS'11, June 7–11, 2011, San Jose, California, USA.

Categories and Subject Descriptors
C.4 [**PERFORMANCE OF SYSTEMS**]: *Reliability, availability and serviceability*

General Terms
Measurement, Reliability

Keywords
Soft Error, Reliability, Cache

1. INTRODUCTION
Roughly 50% of chip real estate today is occupied by caches. To reduce static power dissipation, caches are operated at low voltage using techniques such as drowsy supply voltages [8] and sub-threshold voltage operation [7]. Lower supply voltages during drowsy operation, for instance, reduce the amount of charge held in a SRAM cell thereby making it highly susceptible to single event upsets (SEUs) and soft errors. The impact of SEUs is even more pronounced when chips are deployed in space electronic systems where the intensity of cosmic rays is much higher. In space, under the GEO solar flare model, the failure rate rises by 10 orders of magnitude [3]. Therefore, the resilience of a system to soft errors must be considered and measured during the design phase. Based on these measurements, appropriate error protection mechanisms must be added to every component where required in order to meet the system reliability goals.

Existing research on the impact of SEUs often assumes that soft errors resulting from SEUs are Poisson distributed. While SEUs follow a Poisson process, the resulting soft errors may not be Poisson distributed, as previously noted in [12], especially with higher SEU rates. Some specific reasons include: (1) accesses to memory bits are determined by program behavior or microarchitectural status, (2) faults are masked due to timing, logical, architectural, and microarchitectural factors, and (3) error protection codes change the constant Poisson rate into a time-varying rate.

Several recent studies have focused on computing the FIT rate due to SEUs [1][4][13][16][26]. FIT is a commonly used metric that measures the average number of failures in one billion hours. These studies are rooted in AVF (Architectural Vulnerability Factor) analysis which assumes that no more than one soft error strike can occur during the time a data bit resides in a processor structure. This single-bit fault assumption works well for the vulnerability of processor storage buffers such as load/store queues (LSQ) or reorder buffers (ROBs). Indeed, data resides in these structures for very short periods of time and hence the occurrence of more than one SEU during the residence time has, for all practical purposes, a probability of zero. As such these prior AVF-based approaches are preferred for computing the FIT rate of processor buffers. By contrast, in caches, particularly large

last-level caches, a block of data can potentially reside for millions of cycles between two consecutive accesses to it. When residence time is long each byte, word or block becomes vulnerable to more than one single SEU strike. Hence, it is necessary to complement existing approaches to account for the possibility of multiple errors in large last-level caches. Under the single-bit error dominant assumption, which is the state of the art, the amount of protection against multi-bit errors afforded by simple schemes such as Single Error Correcting Double Error Detecting (SECDED) code cannot be estimated. In the presence of multi-bit errors, it is important to consider the cost/benefit tradeoffs of various protection schemes because error protection mechanisms are expensive in terms of silicon area, energy and performance penalty. Given the likely predominance of multiple errors in large caches, this paper focuses on benchmarking L2 cache vulnerability to soft errors.

One common approach to model and estimate the vulnerability of various components is fault injection. Faults are statistically injected into the detailed RTL model or simulation of the system under study [11][20][29][30]. While this framework is conceptually simple and can model any type of error, including multiple errors, it requires an astronomical number of extremely long simulation experiments to obtain statistically meaningful results: With the current SEU rate of the order of 10^{-25} per bit per cycle, no current benchmark that we are aware of is long enough to register even one single SEU during one of its execution. Even if fault injections are accelerated by orders upon orders of magnitude the procedure still requires massive amounts of test time and only produces statistical estimates. Furthermore the artificial acceleration of faults distorts the results of the test, as we will show at the end of this paper, and accurately correcting for these distortions is a daunting problem.

Given these challenges, we introduce PARMA (Precise Analytical Reliability Model for Architecture). PARMA uses a rigorous fault generation model to address temporal multi-bit errors (MBEs), starting with the probability of SEUs on a single-bit and then expounding the probabilities in both temporal and spatial dimensions while taking error protection schemes into consideration. In the temporal dimension PARMA accounts for the probability of multiple SEU strikes to the same bit during the bit's residency in cache while taking into account scenarios such as processor Stores into L1, and writebacks from L1 to L2 and from L2 to memory. In the spatial dimension PARMA derives the probabilities of one or more errors in bytes, words and finally blocks.

Currently PARMA does not handle the case where a single particle hit causes multiple bit errors (spatial MBEs). This is essentially due to the lack of a proven physical fault model on which we can build. Several studies [9][24] report on spatial MBEs in an array of RAM cells. To the best of our knowledge, only one model, which uses compound Poisson process [2], exists for deciding on the interleaving distance of SECDED code to suppress spatial MBEs. A compound Poisson process models multiple Poisson arrivals (or upsets) simultaneously. The test scheme used in this work is to observe every single spatial MBE and to declare whether this spatial MBE causes a failure or not, based on the interleaving distance of SECDED code. However, modeling various geometric upset patterns resulting from interactions of multiple spatial MBEs, temporal MBEs and multiple single bit errors in a single model is an unanswered, challenging. This is an open and important research area, since such a model would enable accurate evaluation of multi-bit error correcting schemes in the presence of spatial MBEs.

The contributions of our paper are as follows:

- We derive an analytical fault generation and propagation model that captures soft error events rigorously in SRAM arrays.
- We introduce PARMA, a unified framework to measure the reliability of SRAM arrays protected by any possible error protection scheme when temporal multi-bit soft errors exist.
- We show how PARMA can compare the reliability of various protection schemes from benchmark executions.
- By exploiting the accuracy of PARMA, we demonstrate that known techniques to evaluate cache FIT rates in the presence of SECDED, such as accelerated fault injection simulations and first-principle derivations based on Architectural Vulnerability Factor (AVF), can overestimate FIT rates by vast amounts.
- We introduce a new approximate analytical model for measuring the FIT rate of caches protected by word-level SECDED codes. Our new approximate model can quickly and more accurately estimate cache FIT rates in the presence of SECDED.

The rest of this paper is organized as follows. Section 2 develops the PARMA framework in detail and applies it to L2 caches to calculate the FIT rates of various error types under various error protection schemes. Section 3 describes our implementation of the framework on top of SimpleScalar [6]. Results are presented in Section 3.2. Comparison of PARMA with previous proposed schemes such as fault injection and AVF-based approaches are discussed in Section 4. The new approximate model is also presented in Section 4.2.2. Section 5 provides an overview of prior work. We conclude in Section 6.

2. THE PARMA MODEL

Not every SEU in an SRAM cell translates into an error. The impact of an SEU in a cache can be masked due to a variety of reasons such as: the cache line is invalid, the corrupted cache line is overwritten before it is read, the cache line is empty, or the block in the cache line is not referenced again. In general errors can be classified into three categories: Recoverable Errors (RE), Detected Unrecoverable Errors (DUE), and Silent Data Corruption errors (SDC).

Recoverable errors do not impact system integrity and are ignored. DUE FIT rates are further classified into TRUE DUE FIT rates and FALSE DUE FIT rates [26]. An SDC or a TRUE DUE happens when a fault affects the architectural state of the processor and finally corrupts the outcome of an execution. In this paper, we use a simple fault propagation model whereby a fault translates into an error when the fault corrupts the architectural state of the processor. This happens when the processor commits an instruction with at least one faulty bit in the instruction itself (e.g., in its opcode), or when a Load returns data from memory with at least one faulty bit and commits the data value to one of its architectural registers. When this happens, we say that the processor has *consumed* a faulty bit. Note that, to *consume* a faulty bit, the processor must commit the instruction or the Load value, not just fetch it. Thus we track cache errors up until instructions retire in the processor before we deem them SDC, TRUE DUE or FALSE DUE.

When a fault cannot be corrected or masked, we call it *failure*. Well-developed fault propagation models exist [4][13] that may track a fault all the way to I/O or some user-perceived stated, but many studies use simple fault propagation models [1][23] similar to ours to make the problem more tractable. This limitation can be lifted in the PARMA framework, but this is beyond the scope of this paper.

In every cycle, we measure the probability of any component failure in a system, assuming that no more than one failure of the same component can occur in the same cycle.

Let's index each processor cycle by j (where $1 \le j \le T_{exe}$). Then $h_{ERR}(j)$, the discrete time failure probability mass at the j^{th} cycle, is defined as:

$$h_{ERR}(j) = \Pr(\text{Type } ERR \text{ failure at } j \mid \text{system survived all faults until } j) \quad (1)$$

This is the *conditional* probability that a failure of type *ERR* (*ERR* can be SDC, TRUE DUE or FALSE DUE) has occurred at the j^{th} cycle, given that the system survived all types of faults up to the j^{th} cycle. Thus the total expected number of failures of type *ERR* observed during the execution time T_{exe} of a benchmark is:

$$H_{ERR}(T_{exe}) = \sum_{j=1}^{T_{exe}} h_{ERR}(j) = E[ERR] \quad (2)$$

$H_{ERR}(T_{exe})/T_{exe}$ is the failure rate for errors of type *ERR*. We can extrapolate the observed failure rate to the more familiar FIT rate, simply by scaling time. Then, the average FIT rate for a set of applications running one after another, independently on a processor is calculated as:

$$\begin{aligned} \overline{FIT_{ERR}} &= \sum_{\forall \text{ benchmark } i} f_i \times FIT_{i,ERR} \\ &= \sum_{\forall \text{ benchmark } i} f_i \times \frac{E_i[ERR]}{(T_{exe})_i \times CyclePeriod} \times 3600 \times 10^9 \\ &= \frac{\sum_{\forall \text{ benchmark } i} E_i[ERR]}{T_{exe} \times CyclePeriod} \times 3600 \times 10^9 \end{aligned} \quad (3)$$

where f_i ($=T_{exe,i}/T_{exe,all}$) is the fraction of time taken by the execution of benchmark i. $FIT_{i,ERR}$ is the FIT rate extrapolated from $E[ERR]$. We will use (3) to report our results in FIT rates.

2.1 Physical Model – Target System

The system considered in this paper is shown in Figure 1. The L2 cache in the dotted frame is the only component vulnerable to soft errors. All other system components are assumed to be totally fault-free (bullet-proof). This approach isolates the contribution of the L2 cache to the overall FIT rates of the entire system.

2.2 Physical Model – SEU Model

PARMA uses one bit of data and one clock cycle as the minimum units of space and time. The probability that one bit is upset in one cycle is denoted p. We make two basic assumptions about the physical fault model: (i) All clock cycles are independent: whether a bit is struck or not at clock cycle i is independent of whether or not it was hit in cycles $k = 1$ to i-1 (This allows us to consider SEUs as a renewal process, so that the probability distribution of SEUs is always the same after every cycle.), and all cache bits are independent: there is no correlation between SEUs affecting any two cache bits.

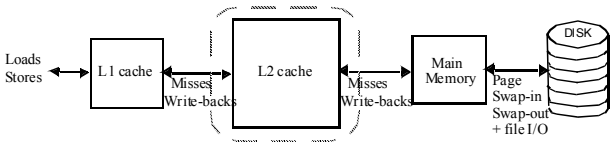


Figure 1. Memory hierarchy model

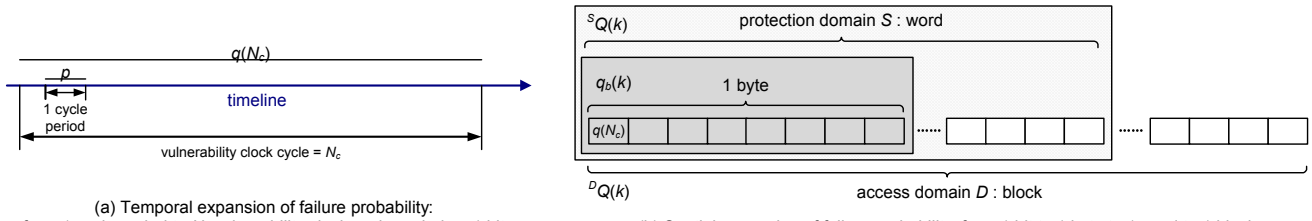
2.3 Vulnerability Clock

In between two consecutive accesses to a memory block in the L2 cache, the block is vulnerable to SEUs. Memory is byte-addressable and therefore PARMA must track the integrity of a memory unit no less than one byte. To this effect, in PARMA, each byte of memory is associated with a *vulnerability clock* which is initialized to zero and which keeps track of the time the byte is exposed to SEUs in L2 while taking into consideration Stores to L1, and writebacks from L1 to L2 and from L2 to memory. In this section, we briefly explain how vulnerability clocks are managed in the PARMA framework.

As soon as a block of data is brought from memory into L2 all the bytes in this block become vulnerable and the vulnerability clock of each byte ticks up in every clock cycle, while the block resides in the L2 cache. At the time when a block is loaded from L2 to L1 there is no knowledge as yet of which bytes are actually going to be consumed by the processor. Hence, we save the snapshot of L2 vulnerability clocks for future use as soon as the L1 copy is filled-in. When the processor accesses L1 and consumes a subset of bytes then we mark the consumed bytes. Eventually when the L1 block is evicted (or when the program finishes) we then compute SDC FIT rates or TRUE DUE FIT rates using information from the saved snapshot of L2 vulnerability cycles and the bytes consumed by the processor. In addition, eviction from L1 can change the vulnerability status of the L2 copy. When a block is loaded in L1 two copies of the same block coexist, one in L1 and one in L2. The copy in L1 is bullet-proof to any strike, but the copy in L2 is still vulnerable. Thus we must keep track of two cases: (i) the block copy in L1 is modified, in which case it will be written back and the vulnerability clocks of every byte of the L2 copy will be reset to zero so that PARMA does not count the same error twice, or (ii) the block copy in L1 is not modified, in which case the block will not be written back to L2. In this latter case, the block copy in L2 remains vulnerable during the entire stay of the block in L1 and the vulnerability clocks of its bytes will not be reset when the block is silently replaced in L1.

When a block is replaced in L2, it is either replaced silently, i.e. the block is not written back to main memory since the block is clean, or it is written back to main memory if it is dirty. If it is replaced silently, then the vulnerability clocks for all the bytes in the block remain unchanged in the main memory as if they had never been loaded in L2 before. If it is written back to main memory because it was modified, then each byte in the L2 block carries its corresponding vulnerability clock with it to main memory but the clocks stop ticking in memory (since memory is also assumed to be bullet-proof) and will restart later on when the block is reloaded in the L2 cache on a miss.

We update failure metrics every time a cache line is accessed. At these times, we calculate the conditional probability of various failure types, called discrete failure probability mass, based on the vulnerability clock in each byte of the block. The first step towards this goal is to calculate the distribution of bit faults in a set of bytes.



(a) Temporal expansion of failure probability: from 1 cycle period to N_c vulnerability clock cycle period on 1 bit

(b) Spatial expansion of failure probability: from 1 bit to 1 byte to 1 word to 1 block

Figure 2. Temporal and spatial expansions of probability calculations when every individual word is protected

2.4 Overview of PARMA Probability Calculations

Figure 2 overviews PARMA's calculations of the failure rates. PARMA starts with the probability p that one bit is flipped in one processor cycle. Then, exploiting the basic assumption that the rate of SEUs is time-independent, PARMA expands the probability p into $q(N_c)$, the probability that a given bit is faulty after N_c vulnerability cycles. This is illustrated in Figure 2(a), and will be explained in Section 2.5.1. Then, exploiting the assumption that SEUs are spatially independent, PARMA expands the probability $q(N_c)$ into $q_b(k)$, the probability that a byte has k faulty bit(s) after N_c vulnerability cycles. This is illustrated in Figure 2(b), and will be explained in Section 2.5.2.

We call a set of bytes that are bundled together as a *domain*. Let's denote an arbitrary domain as A. Then the goal of PARMA is to calculate ${}^{A}Q(k)$, the probability that there are k faulty bit(s) after N_c vulnerability cycles in a domain A, whenever A is accessed. Once we have ${}^{A}Q(k)$, PARMA determines what k should be in order to have an error in that domain, considering the kind of protection scheme covering a domain (for example, an unprotected domain will have one SDC whenever $k > 0$). Throughout the benchmark execution, PARMA keeps measuring the probabilities of having errors in the domain. Finally, based on the number of errors expected whenever the domain is accessed, PARMA accumulates the expected numbers of faults using ${}^{A}Q(k)$s measured during the benchmark execution.

Let's call a set of bytes that are accessed together an *access domain* (denoted D), and a set of bytes that are protected together by the same code under any specific protection scheme a *protection domain* (denoted S). Access domain and protection domain can be the same or one can be a subset of the other. For example the cache block is the access domain when the L2 cache writes back to main memory since the transfer happens at the block -level, but the protection domain is a word if a word-level SECDED code is used. In this case, one faulty bit will be corrected and two faulty bits will be detected within a word. In general, the protection domain can be a word, a block, or a set of bytes that are protected together as in interleaved SECDED [17]. In the following we assume that the access domain is an L2 cache block and that the protection domain is a subset of the access domain.

Figure 2(b) illustrates how PARMA calculates ${}^{S}Q(k)$, the probability that the set of bytes in a protection domain S has k faulty bit(s) after N_c vulnerability cycles, from probability $q_b(k)$, by exploiting the spatial-independence assumption further. This expansion will be explained in Section 2.5.3. In the example of Figure 2(b), the protection domain is one word.

2.5 Fault Generation Model: Probability Distribution of Faults in a Set of Bytes

2.5.1 Temporal Expansion from 1 to N_c Vulnerability Clock Cycles in a Bit

We first calculate the probability of i SEUs on one cache bit in N_c vulnerability clock cycles.

$$P_i(N_c) = \binom{N_c}{i} p^i (1-p)^{N_c - i}, i = 0,\ldots, N_c \quad (4)$$

This result is based on the fact that, in any cycle, one SEU can cause a bit to flip, with probability p. A bit can flip i times up to N_c times in N_c vulnerability cycles, and bit flips may happen in any one set of i cycles amongst the N_c vulnerability cycles.

With (4), we can calculate the probability q that a given cache bit will be faulty after N_c vulnerability cycles, knowing that a bit is faulty if it is flipped an odd number of times during N_c cycles.

$$q(N_c) = \sum_{i=0}^{\lfloor N_c/2 \rfloor} P_{2i+1}(N_c) \quad (5)$$

2.5.2 Spatial Expansion from 1 Bit to 1 Byte

Given the value of $q(N_c)$ we calculate the probabilities $q_b(k)$ that a byte residing in an L2 cache line will experience k bit errors in N_c vulnerability cycles, where k = 0 to 8 (the number of errors can be from 0 to 8). We first choose any k out of 8 bits and then compute the probability of errors in these k bits while the remaining (8-k) bits are fault-free.

$$q_b(k) = \binom{8}{k} q(N_c)^k (1 - q(N_c))^{8-k}, k = 0,\ldots,8 \quad (6)$$

To simplify the notation, $q_b(k)$ is an implicit function of N_c.

2.5.3 Spatial Expansion from 1 Byte to the Whole Protection Domain

We can now calculate the probability distribution of k bit faults (k = 0 to $8N_S$) in a protection domain S of N_S bytes. To do this, we index each byte by j, j = 1 to N_S. We use the notation $\{j\} \in S$ to sweep all bytes in protection domain S. A vulnerability clock with value N_c is associated with each byte and is implicit in the notation $q_{b,j}(k)$. The probability of k faulty bits within the protection domain can be expressed as a function of all $q_{b,j}(k)$'s as follows:

$${}^{S}Q_m(k) = \sum_{(\sum l_j = k)} \prod_{\{j\} \in S} q_{b,j}(l_j) \quad (7)$$

where m is an optional index characterizing the protection domain in the access domain. In (7), we choose all the cases in which S

Table 1. Errors as a function of k and N_S

	No Parity	1-bit Parity		SECDED			
	SDC	TRUE DUE	SDC	TRUE DUE		SDC	
				word-level	block-level	word-level	block-level
D: Access Domain	Block	Block	Block	Blk containing *M* words	Block	Blk containing *M* words	Block
S: Protection Domain	N/A	Block	Block	Word	Block	Word	Block
Faulty bits	≥1 in *C*	∀ odd in *S*, ≥1 in *C*	∀ even >0 in *S*, ≥1 in *C*	2 in any S_m, ≥1 in that C_m	2 in *S*, ≥1 in *C*	≥3 in any S_m, ≥1 in that C_m	≥3 in *S*, ≥1 in *C*
Notation	${}^{B}E^{NP,SDC}$	${}^{B}E^{P1B,TRUE\_DUE}$	${}^{B}E^{P1B,SDC}$	${}^{W}E^{SW,TRUE\_DUE}$	${}^{B}E^{SB,TRUE\_DUE}$	${}^{W}E^{SW,SDC}$	${}^{B}E^{SB,SDC}$

has k faulty bits among N_S bytes and compose the probability accordingly. m is omitted in the notation if the access domain holds only one protection domain. We do not need to calculate the probability distributions at the granularity of an access domain, as error types such as DUE or SDC are calculated for each protection domain. Because errors in different protection domains are independent of each other, they are additive. For instance, if L2 blocks are protected by parity then the summation of (7) over all odd k values and all cache blocks gives the L2 DUE. This calculation is performed whenever a block is fetched from L2 to L1. We now show how to classify errors into SDC and TRUE DUE based on how the data is accessed in L1.

2.6 Fault Propagation Model: SDC FIT Rates and DUE FIT Rates in Each Error Protection Scheme

2.6.1 Probability Calculations of Generic SDC Errors and TRUE DUE Errors

Even if a corrupted L2 cache block reaches the first level cache, the execution may still be correct. Indeed, if no bit in the block is *consumed* during the sojourn of the block in L1, then faults in an L2 cache block cause no error and, if it was detected as an L2 DUE, it becomes a FALSE DUE. The first time a faulty bit is *consumed* from the block residing in L1, the bit fault turns into either an SDC or a TRUE DUE. If the processor stores a value in an L1 byte before the byte is read then that byte becomes error-free. We need to consider all the above scenarios to compute TRUE DUE and SDC.

Let's denote the set of consumed bytes in the L1 copy of a protection domain as C. Then the complement set $\overline{C}$ contains all the *unconsumed* bytes in the same domain. C and $\overline{C}$ are illustrated in Figure 3. $\overline{C}$ is the set of unconsumed bytes when the L1 copy is evicted. The union of C and $\overline{C}$ is the protection domain. We use the same notation as in (7). The probability of having k faulty bits *only* in $\overline{C}$, but having no faulty bit in C is obtained as:

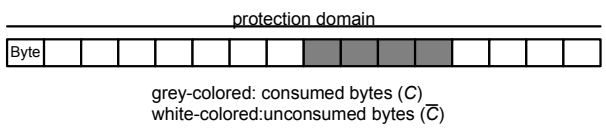


Figure 3. C : A set of consumed bytes in a protection domain

$$ {}^{C_m}Q_m(0) \times {}^{\overline{C}_m}Q_m(k) \tag{8} $$

where m is an optional index for a protection domain in the access domain. When the protection domain is the same as the access domain, we drop the optional index m from (8). (8) simply states that C has no fault and $\overline{C}$ has k faults. $\overline{C}$ is not known until the block is replaced in L1, or until the end of the execution, whichever comes first. (8) is used to calculate SDC and TRUE DUE FIT rates in the following subsections.

2.6.2 Protection Schemes

We have shown the generality of the approach with any granularity of protection and access domains. In the balance of this paper, we restrict the protection domain to either a word (W) or a block (B), and the access domain to a block. We consider four schemes: no error detection or correction (NP), 1 bit parity (P1B), SECDED per block (SB) and SECDED per word (SW). Table 1 summarizes SDC, TRUE DUE and FALSE DUE classifications in different protection schemes. The number of faulty bits necessary to diagnose DUE or SDC are shown in Table 1 for all scenarios. For example, when the L2 cache is protected with word-level SECDED, it encounters an SDC when (i) three or more faulty bits are in the protection domains (word) when an L2 block is accessed, and (ii) the processor *consumes* at least one bit among these faulty bits. In the following subsections, we show how to calculate SDC FIT rates and DUE FIT rates using (7) and (8).

2.6.3 SDC Errors

If an L2 block is copied to L1 with at least one undetected faulty bit, then one SDC is counted for all cases when any one of the faulty bit(s) is consumed. As shown in Table 1 (row "faulty bits"), the SDC FIT rate depends on the protection scheme in L2. To compute the probability of an SDC, we simply subtract the probability of having undetected faulty bit(s) in $\overline{C}$ only, from the probability of having undetected faulty bit(s) in the entire block. Thus, the expected number of SDCs under no-protection (NP) is as follows:

$$ {}^{B}E^{NP,SDC} = \sum_{k=1}^{8N_B} {}^{B}Q(k) - {}^{C}Q(0) \times \sum_{i=1}^{8N_{\overline{C}}} {}^{\overline{C}}Q(i) \tag{9} $$

The first summation term is the probability of any number of faults in the block and the second term is the probability of no

fault in C multiplied by the probability of any number of faults in $\overline{C}$. Similarly, with single error detection per block (denoted as P1B), all odd numbers of error are detected but even numbers of error go undetected. So the expected number of SDCs is:

$$ {}^{B}E^{P1B,SDC} = \sum_{\forall even\ k>0}^{8N_B} {}^{B}Q(k) - {}^{C}Q(0) \times \sum_{\forall even\ i>0}^{8N_{\overline{C}}} {}^{\overline{C}}Q(i) \quad (10) $$

Consider now the case of an L2 cache with SECDED per block (denoted as SB). The expected numbers of SDCs is:

$$ {}^{B}E^{SB,SDC} = \sum_{k=3}^{8N_B} {}^{B}Q(k) - {}^{C}Q(0) \times \sum_{i=3}^{8N_{\overline{C}}} {}^{\overline{C}}Q(i) \quad (11) $$

Finally, consider the case of an L2 cache with SECDED per word (denoted as SW). Let N_W be the number of bytes in a word and M be the number of words in a block. Up to M faulty bits can be corrected in a block and we need to include all M words whenever a block is accessed. Let's index each word in the block by m, where m = 1 to M. The expected number of SDCs with word-level SECDED (denoted as SW) is:

$$ \begin{aligned} {}^{B}E^{SW,SDC} &= \sum_{m=1}^{M} {}^{W}E_m^{SW,SDC} \\ &= \sum_{m=1}^{M} \sum_{k=3}^{8N_W} {}^{W}Q_m(k) - \sum_{m=1}^{M} \left[{}^{C_m}Q_m(0) \times \sum_{i=3}^{8N_{\overline{C_m}}} {}^{\overline{C_m}}Q_m(i) \right] \end{aligned} \quad (12) $$

Whenever an L2 cache line is filled from memory, an L2 block is accessed due to an L1 miss, or an L1 block is evicted (discarded or written back to L2), (9)-(12) are invoked to yield the expected number of SDC faults since the last access to the block. These expected numbers of failures are accumulated until the simulation ends.

2.6.4 DUE Errors

Unlike for SDC errors, DUE FIT rates are further classified as TRUE DUE FIT rates and FALSE DUE FIT rates. As shown in Table 1, DUE FIT rates are classified differently in different protection schemes. DUEs are raised if and as soon as they are detected – i.e., when an L2 block is fetched if it is protected by a protection code with error detection capabilities. Just as for SDC FIT rates, PARMA counts them as TRUE DUE FIT rates if at least one faulty bit is consumed by the processor after the DUE has been raised. FALSE DUE FIT rates are computed as the probability that all faulty bits fall in $\overline{C}$ only. TRUE DUE FIT rates are obtained by subtracting the probability of FALSE DUE from the total probability that the faulty bits fall anywhere in the protection domain. There is no DUE under NP. With P1B, DUE FIT rates are calculated as:

$$ {}^{B}E^{P1B,FALSE\_DUE} = {}^{C}Q(0) \times \sum_{\forall odd\ i}^{8N_{\overline{C}}} {}^{\overline{C}}Q(i) \quad (13) $$

$$ {}^{B}E^{P1B,TRUE\_DUE} = \sum_{\forall odd\ k}^{8N_B} {}^{B}Q(k) - {}^{C}Q(0) \times \sum_{\forall odd\ i}^{8N_{\overline{C}}} {}^{\overline{C}}Q(i) \quad (14) $$

With block-level SECDED, the expected number of DUEs is:

$$ {}^{B}E^{SB,FALSE\_DUE} = {}^{C}Q(0) \times {}^{\overline{C}}Q(2) \quad (15) $$

$$ {}^{B}E^{SB,TRUE\_DUE} = {}^{B}Q(2) - {}^{C}Q(0) \times {}^{\overline{C}}Q(2) \quad (16) $$

With word-level SECDED, the expected number of DUEs is:

$$ \begin{aligned} {}^{B}E^{SW,FALSE\_DUE} &= \sum_{m=1}^{M} {}^{W}E_m^{SW,FALSE\_DUE} \\ &= \sum_{m=1}^{M} \left[{}^{C_m}Q_m(0) \times {}^{\overline{C_m}}Q_m(2) \right] \end{aligned} \quad (17) $$

$$ \begin{aligned} {}^{B}E^{SW,TRUE\_DUE} &= \sum_{m=1}^{M} {}^{W}E_m^{SW,TRUE\_DUE} \\ &= \sum_{m=1}^{M} {}^{W}Q_m(2) - \sum_{m=1}^{M} \left[{}^{C_m}Q_m(0) \times {}^{\overline{C_m}}Q_m(2) \right] \end{aligned} \quad (18) $$

3. PARMA SIMULATIONS

3.1 Parameters in PARMA Simulations

We have implemented PARMA on top of the SimpleScalar/Alpha simulator [6] to measure FIT rates for various benchmarks. The PARMA model starts with a probability p representing the probability that any one bit is flipped during one clock period. To calculate p, we use the Poisson rate λ projected by ITRS 2007 [22], which is 1,150 SEU FIT rates (equivalent to SER in 10^9 hours) for a 1Mbit SRAM in 65nm technology. The target processor frequency is 3GHz, so the value of p is 1.0155×10^{-25} from the Poisson probability mass function by summing the probabilities of all the odd numbers of SEU arrivals during a cycle.

In our simulations we add the time to decode the protection check bits to the latency of a block access. 1-bit parity causes little performance penalty and hence the access latency with 1-bit parity is same as no protection scheme. On the other hand, both space (number of check bits) and time (decoding delays) overheads are incurred for SECDED. Naseer, et al. [18] reported the decoder delays for various protection codes in a 90nm technology. We have extrapolated the reported SECDED decoder delays from [18] and scaled it to our 65nm technology. In summary, 4-byte word-level SECDED has 7 check bits with 0.827ns decoder delay. Similarly, 32-byte block-level SECDED has 11 check bits overhead with 1.244ns decoder delay. The resulting L2 access latencies as calculated by Cacti [25] for different protection schemes are shown in Table 2.

Table 2. Cache hierarchy parameters

Cache	Size	Associativity	Latency [cycles]		
IL1	16KB	1-way	2		
DL1	16KB	4-way	3		
UL2	256KB	8-way	NP/P1B: 10	SW (4B): 13	SB (32B): 14

The target processor simulated in our 65nm technology is a 4-wide out-of-order processor with 64-entry ROB, 32-entry LSQ, and McFarling's hybrid branch predictor. The processor runs at 3GHz with 150 cycles of latency to off-chip main memory. L1-I, L1-D and unified L2 caches all have 32-byte lines. All the caches are non-blocking, write-back caches. All the cache parameters shown in Table 2 are obtained from Cacti 5 [25]. The delay for fetching and decoding ECC check bits is included in the L2 latencies.

Table 3. FIT rates in various protection schemes

(a) NP_ SDC: no-protection/SDC
P1B_TRUE_DUE:1bit parity/TRUE DUE
(b) P1B_FALSE_DUE:1bit parity/FALSE DUE
(c) P1B_ SDC: 1bit parity/SDC
(d) SB_TRUE_DUE: block-level 1bit SECDED/TRUE DUE
(e) SB_FALSE_DUE: block-level 1bit SECDED /FALSE DUE
(f) SB_SDC: block-level 1bit SECDED /SDC
(g) SW_TRUE_DUE: word-level 1bit SECDED /TRUE DUE
(h) SW_FALSE_DUE: word-level 1bit SECDED /TRUE DUE
(i) SW_SDC: word-level 1bit SECDED /SDC
(j) AVF_SDC: SDC calculated by AVF method
(k) %_Diff: Percentage difference of SDC (PARMA/AVF)

Bench	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
ammp	320.32	419.27	2.50E-14	2.53E-14	1.53E-14	1.32E-29	1.99E-15	2.92E-15	1.02E-31	320.32	0.000%
art	48.76	16.74	1.22E-16	1.22E-16	3.70E-17	3.89E-34	1.03E-17	9.01E-18	3.21E-36	48.76	0.000%
crafty	429.47	716.45	1.99E-14	2.34E-14	4.74E-14	4.24E-30	1.85E-15	6.41E-15	2.83E-32	429.47	0.000%
eon	382.25	298.23	1.45E-14	1.63E-14	6.03E-15	2.72E-30	1.69E-15	9.77E-16	3.57E-32	382.25	0.000%
facerec	98.08	0.59	9.80E-17	9.79E-17	1.37E-18	8.88E-35	1.18E-17	2.45E-19	1.22E-36	98.08	0.000%
galgel	60.35	77.61	9.52E-17	9.52E-17	8.53E-17	3.54E-34	8.15E-18	1.38E-17	2.72E-36	60.35	0.000%
gap	138.59	22.27	3.32E-16	3.94E-16	5.26E-17	2.34E-33	4.30E-17	8.80E-18	2.59E-35	138.59	0.000%
gcc	349.11	229.96	3.76E-15	4.86E-15	3.25E-15	1.89E-31	4.65E-16	4.68E-16	1.86E-33	349.11	0.000%
gzip	547.53	1115.56	1.05E-14	1.17E-14	9.56E-15	2.86E-31	1.30E-15	1.24E-15	3.67E-33	547.53	0.000%
mcf	14.71	14.43	1.28E-17	1.28E-17	3.23E-17	2.93E-35	1.13E-18	4.35E-18	1.89E-37	14.71	0.000%
mesa	460.52	112.19	4.50E-15	5.03E-15	8.57E-16	4.01E-32	5.44E-16	1.52E-16	4.73E-34	460.52	0.000%
parser	138.54	380.34	1.86E-15	2.00E-15	4.12E-15	4.59E-32	1.47E-16	5.82E-16	2.97E-34	138.54	0.000%
perlbmk	100.82	315.37	2.74E-15	2.97E-15	8.85E-15	6.96E-32	2.36E-16	1.17E-15	4.46E-34	100.82	0.000%
sixtrack	76.24	7.92	3.75E-16	3.91E-16	1.39E-16	9.70E-33	4.26E-17	2.14E-17	1.12E-34	76.24	0.000%
twolf	193.40	419.25	1.52E-15	1.56E-15	2.88E-15	1.45E-32	1.24E-16	4.11E-16	1.06E-34	193.40	0.000%
vortex	831.26	324.74	8.57E-15	9.63E-15	2.80E-15	1.70E-31	1.04E-15	4.31E-16	1.93E-33	831.26	0.000%
vpr	184.31	369.25	2.16E-15	2.18E-15	3.27E-15	3.04E-32	1.83E-16	4.79E-16	2.45E-34	184.31	0.000%
wupwise	146.69	130.00	1.99E-15	2.02E-15	7.28E-16	1.75E-32	1.63E-16	1.70E-16	1.42E-34	146.69	0.000%
Avg	155.66	217.17	2.53E-15	3.45E-15	3.59E-15	8.34E-31	2.25E-16	4.07E-16	2.92E-33	155.66	0.000%

The benchmarks are execution samples of 100 million committed instructions obtained from 18 SPEC2K benchmarks compiled for the alpha ISA and shown in Table 3. Every sample was selected using SimPoint [19], which is a well-know method for selecting the most representative sample sequence of instructions to simulate.

3.2 Simulation Results

Table 3 shows the FIT rates extrapolated from the failure rates measured in our simulations. The average FIT rates in the last row are derived from (3). Without any error protection the SDC FIT rate (Table 3(a)) is still very high, varying from 15 to 832, which is not acceptable for most manufacturers today. With 1-bit parity, SDC FIT rates are in effect turned into TRUE DUE FIT rates. The SDC FIT rate with 1-bit parity is 16 orders of magnitude smaller than the SDC FIT rate with no protection. Hence, 1-bit parity is more than sufficient to satisfy the stringent SDC FIT rate requirements of most manufacturers. However the DUE FIT rate may not be within the reliability budget for an L2 cache. By upgrading to block-level SECDED (column (d) and (e) in Table 3) or word-level SECDED (column (g) and (h) in Table 3), the average DUE FIT rates drop by a factor of 5.30×10^{16} and 5.90×10^{17} respectively. Due to different memory footprints, and different memory behaviors in each benchmark, the FIT rate varies widely amongst programs, as shown in Table 3. Note that the SDC FIT rates calculated by PARMA match AVF FIT rates down to 10 decimal points, giving 0% differences in all cases in Table 3(column (k)). This is expected.

One distinct advantage of word-level SECDED over block-level SECDED is that word-level SECDED can correct more words and thus yields lower DUE FIT rates than block-level SECDED. The numbers show that word-level SECDED capable of correcting one faulty bit in every four-byte word is on average about 11.14 times stronger than block-level SECDED capable of correcting one faulty bit in every 32-byte block. However, word-level SECDED has 21.9% bit overhead for a 256KB cache. Because the block-level SECDED DUE is already very tiny ($10^{-15} \sim 10^{-17}$), reducing the failure rate again by an order of magnitude may not be a critical reliability gain, although using more silicon with word-level SECDED as compared to block-level SECDED might need stronger justification. If the reduction of the failure rate is the sole purpose of a (stronger) word-level SECDED, investing more in silicon may not be justified.

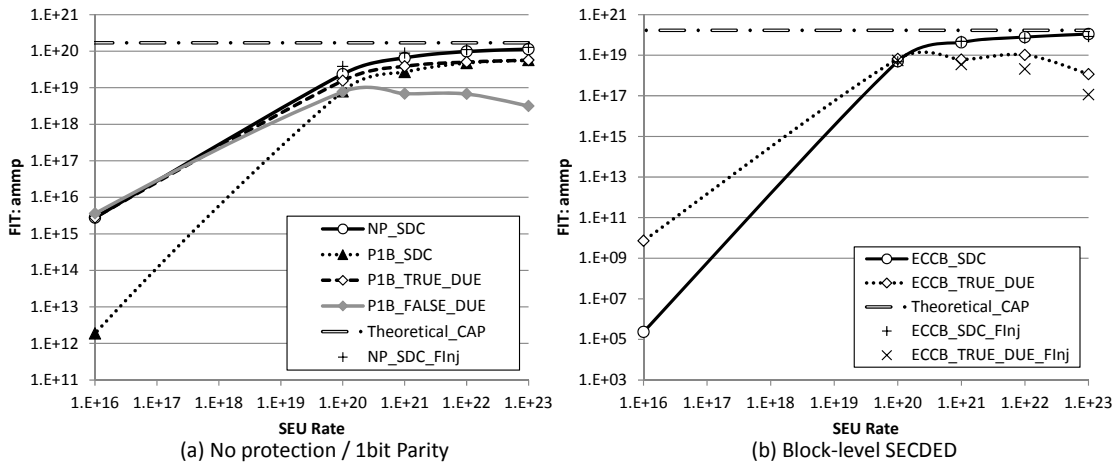


Figure 4. FIT rates under different SEU rates

4. COMPARISON WITH PREVIOUS MODELS

Current popular models to estimate FIT rates in the presence of error protection codes include accelerated fault injection simulations and approximate analytical models. In this section we demonstrate that previous models can vastly overestimate FIT rates for a variety of reasons. Therefore PARMA is a necessary tool to reach accurate design decisions.

4.1 Accelerated Fault Injection Simulations

Fault injection simulations are common for reliability studies. They have the advantage to handle various fault scenarios in a fairly simple framework. For example, the effectiveness of error correction codes can be measured with fault injection simulation. Unfortunately fault injection simulations require greatly accelerated SEU rates in order to benchmark the vulnerability of caches because the benchmarks' execution times are extremely short compared to the expected time between two SEUs in the real world. In order to observe at least several SEUs that are not architecturally masked during one benchmark execution, one need to raise the SEU rate to the order of 10^{20} SEUs per 10^9 hours, an astronomical 17 orders of magnitude acceleration. 10^{20} SEUs corresponds roughly to p=10^{-8} and generates about 10 SEUs during a 1 billion cycle simulation run, and fewer than 10 errors when accounting for architectural masking effect. Accelerating injection rates is problematic because it causes distortions in the results and no credible methodology exists to scale the results to the real SEU rate. This is true both for quantitative *and* qualitative design explorations.

Figure 4 shows the FIT rates measured by accurate PARMA simulations for the SimPoint of the SPEC ammp benchmark for a range of accelerated SEU rates, which includes the fault injection rates range compatible with feasible simulations (i.e., SEU rate >> 10^{20}). PARMA measurements of SDC FIT rates and TRUE DUE FIT rates under 1-bit Parity and block-level SECDED in this SEU range are shown in Figure 4(a) and (b). Several fault injection test results are also displayed in Figure 4 with suffix 'FInj'. The horizontal line labeled 'Theoretical_CAP' shows the maximum possible failure rates obtained when every access to any L2 block with a non-zero vulnerability clock is counted as one failure with probability one. With highly accelerated SEU rates, FIT rate measurements from fault injection simulations converge to this cap. As shown in the figure, under very high SEU rates, SDC FIT rates catch up with DUE FIT rates under Parity to a point that they become equal, which is a highly unrealistic result in reality.

The distortions are even more serious under block-level ECC (Figure 4(b)). We observe that in the lower SEU rate range TRUE DUE FIT rates dominate SDC FIT rates (as is expected in the real world). However, as the SEU rate increases, an inversion occurs, in which SDC FIT rates dominate TRUE DUE FIT rates, both a non-intuitive and incorrect observation. The basic reason for these distortions can be illustrated with a simple example. Consider the case that a block is protected with 1-bit parity. This block will experience SDCs only when an even number of bits are flipped, and all odd numbers of bit flips will turn into DUE. Since 1-bit flips are the most common case one would expect that SDC FIT rates would be much lower than TRUE DUE rates. However, with highly accelerated SEU rates, the probability of having an even number of faulty bits becomes almost the same as the probability of having an odd number of faulty bits. Hence, the SDC FIT rate and DUE FIT rate converge. Similarly, in the case of SECDED the probability of having more than two faulty bits overwhelms the probability of having two faulty bits and hence SDC and TRUE DUE FIT rates converge even with SECDED protection.

The main lesson to learn from Figure 4 is that FIT rate distortions are high in the high SEU rate range compatible with fault injection rates typically used in simulations. With SEU rates higher than 10^{20} SEUs per 10^9 hours, the measured FIT is limited by the theoretical cap, and the non-linear effects in the relation between SDC FIT rates or DUE FIT rates and SEU rates lead to incorrect *qualitative* conclusions. These observations show the importance of an accurate tool such as PARMA to evaluate the relative strength of various error protection schemes.

4.2 Approximate Analytical Models for ECC-Protected Caches

PARMA provides accurate FIT rate measurements. With PARMA we can now validate prior approaches taken to compare the strength of various error protection codes. We have just shown that fault injection simulations can distort the observations. We now evaluate the effectiveness of existing approximate analytical

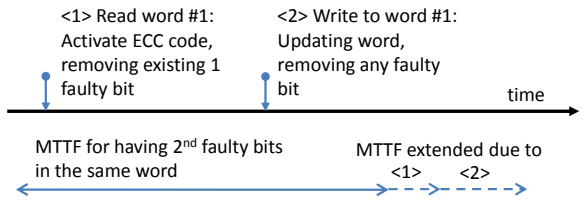


Figure 5. MTTF for SECDED protected cache [17][21] affected by cache accesses

models to compute the FIT rates of caches protected by SECDED, and then propose a better one.

Approximate analytical models have been proposed in the past to evaluate the Mean Time To Failure (MTTF) and FIT rates of caches protected by SECDED [17][21]. We will now compare the FIT rates measured by PARMA with FIT rates computed by these approximate models and show why these prior approaches overestimate FIT rates significantly. The main reason for this vast discrepancy is that these existing approximate models overlook the fact that, between two successive SEU strikes in a word, several fault mitigating events can occur. Every time a word is accessed, error correction is invoked and a single fault can be corrected multiple times, thus prolonging the MTTF by vast amounts of time (see Figure 5). Therefore these approximate models underestimate the MTTF and overestimate the FIT rates. Based on this insight we have developed a more rigorous approximate model for cache failures under SECDED protection. This new model predicts FIT rates that are much closer to the accurate values computed by PARMA than existing approximate models.

4.2.1 Previous Approximate Models

The calculation of FIT using AVF is as follows [5][12]:

$$\text{Soft Error Rate}_{\text{component}} = \text{AVF}_{\text{component}} \times \text{intrinsic Soft Error Rate}_{\text{component}} \quad (19)$$

To apply (19) in the case of a cache where words are protected by SECDED, we need to find first the intrinsic SER (Soft Error Rate) of a word-level SECDED protected cache. To compute the intrinsic SER for a word-level SECDED protected cache it is necessary to estimate the expected time for having two faulty bits in a word. Models for such estimation were proposed in [17] and [21] in the context of cache scrubbing. There the goal was to determine the optimal cache scrubbing interval such that only a maximum of one fault is expected to occur between scrubbing intervals so that scrubbing provides a guard mechanism against the occurrence of SDC faults. These models, in effect, calculate the intrinsic SER for caches protected by SECDED.

These models calculate the expected time of a second SEU strike given that there has been a first SEU strike already. The most critical information that was overlooked in these studies is that in between any two SEU strikes single-bit faults will be corrected by SECDED if there is at least one access to the cache in between the two strikes. Figure 5 illustrates how cache accesses affect the MTTF. The first read activates SECDED for the word and the following write overwrites the word. The effect in these two cases is to improve (reduce) the intrinsic SER.

In order to evaluate how cache accesses and corrections affect the MTTF under SECDED protection we now compare PARMA results with results obtained by the approximate model in [21]. Note that the authors in [17] compared their model with the closed form MTTF formula in [21] and concluded that both models have very similar outcomes. Hence, our comparison results with [21] and the resulting conclusions also hold well against [17].

In [21] the authors provided a closed-form MTTF analytical model for caches protected by word-level SECDED as:

$$\text{MTTF} = \frac{1}{L}\sqrt{\frac{\pi}{2M}} \quad (20)$$

where L is the soft error rate in the protection domain and M is the total number of protection domains in the cache.

Table 4 DUE FIT rates in word-level SECDED protected cache

Name	AVF	AVFxFIT from (20)	FIT from Eq (23)	FIT from PARMA
ammp	40.977%	2.9374	8.4182E-14	4.9114E-15
art	2.849%	0.2042	4.5179E-16	1.93476E-17
crafty	61.078%	4.3784	3.3463E-14	8.25685E-15
eon	99.049%	7.1003	1.3441E-13	2.67121E-15
facerec	4.319%	0.3096	1.3138E-15	1.20444E-17
galgel	6.010%	0.4308	7.0577E-16	2.19513E-17
gap	7.118%	0.5103	4.5248E-16	5.18293E-17
gcc	27.658%	1.9827	7.7612E-15	9.33375E-16
gzip	83.466%	5.9832	6.9763E-14	2.53328E-15
mcf	1.267%	0.0908	2.9364E-16	5.47892E-18
mesa	30.070%	2.1555	1.6881E-14	6.96557E-16
parser	22.983%	1.6475	1.8796E-14	7.28453E-16
perlbmk	31.621%	2.2667	2.9209E-14	1.41053E-15
sixtrack	3.916%	0.2807	5.9788E-16	6.39658E-17
twolf	26.750%	1.9176	2.2392E-14	5.35443E-16
vortex	53.171%	3.8115	3.6437E-14	1.4704E-15
vpr	24.232%	1.7371	4.0074E-14	6.61992E-16
wupwise	12.183%	0.8733	1.1242E-14	3.33145E-16
Average	27.333%	2.1454	2.8246E-14	6.31707E-16

With (20), we calculate the FIT rate of our target cache with word-level SECDED and we multiply it by the AVF ([1][4]) we measured on our target system executing the target benchmarks. Table 4 shows the results (3rd column).

We observe that the model in [21] (and [17] as well) overestimates the FIT rate of the cache. The PARMA FIT estimates (5th column) are 16 *orders of magnitude* smaller. Tens of millions of L2 cache accesses are made during the execution, each of which may correct single faults in words, thus prolonging the MTTF.

4.2.2 A New Approximate Model

PARMA is not just a benchmarking tool; it also provides deeper insights into the reliability of cache structures. Prior analytical models overestimate FIT rates by not considering how intermediate accesses may activate ECC to clean single-bit faults before a second SEU strike. In this section our goal is to introduce

a new approximate analytical model using AVF methodology [4] to estimate the DUE FIT rate of SECDED caches while taking into account ECC activation effects as well as the architectural masking effects.

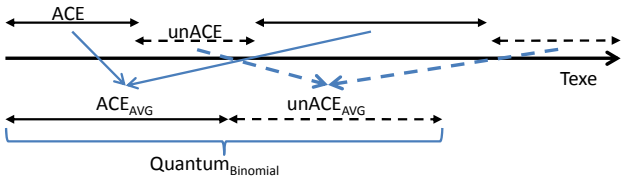


Figure 6. Alternance between ACE and unACE intervals

The L2 cache AVF is computed as:

$$\mathrm{AVF}_{\mathrm{Cache}} = \frac{1}{n \times \mathrm{T}_{\mathrm{exe}}} \times \sum_{\forall n \text{ words in a cache}} \mathrm{T}_{\mathrm{ACE},\, n} = \frac{\mathrm{T}_{\mathrm{ACE\_cache}}}{n \times \mathrm{T}_{\mathrm{exe}}} \qquad (21)$$

Thus, from $\mathrm{T}_{\mathrm{ACE\_cache}} = \mathrm{AVF}_{\mathrm{Cache}} \times n \times \mathrm{T}_{\mathrm{exe}}$, we compute the average ACE cycles for a single word in that cache, called $\mathrm{T}_{\mathrm{ACE\_word}}$, by dividing $\mathrm{T}_{\mathrm{ACE\_cache}}$ by the total number of accesses to n words in L2.

$\mathrm{T}_{\mathrm{ACE\_word}}$ is the average number of cycles between two accesses to a word. Since ECC is activated on each access $\mathrm{T}_{\mathrm{ACE\_word}}$ is also the average interval when SECDED may correct a single faulty bit in a word. In order to compute the DUE FIT rate with word-level SECDED our goal now is to calculate the probability that *exactly two* temporal faults on the same word happen during $\mathrm{T}_{\mathrm{ACE\_word}}$. After $\mathrm{T}_{\mathrm{ACE\_word}}$ there are only two outcomes: failure or survival. Hence, we calculate the probability of having a failure in a word using a geometric distribution where each word access becomes a Bernoulli trial. During $\mathrm{T}_{\mathrm{ACE\_word}}$ the SER rate is $\lambda_{\mathrm{ACE\_word}}$ (which can be easily computed from ITRS2007 SEU FIT rates).

Using the Poisson probability mass function, we can get the probability that a word has *exactly two* SEUs (i.e., the probability of a DUE with SECDED) during $\mathrm{T}_{\mathrm{ACE\_word}}$. This probability is given by:

$$P_{DUE} = \frac{(\lambda_{\mathrm{ACE\_word}})^2}{2!} e^{-\lambda_{\mathrm{ACE\_word}}} \qquad (22)$$

Then from the geometric distribution, the expected number of attempts to see a DUE in a word under SECDED is $1/P_{DUE}$. Note that the duration of each attempt here is not $\mathrm{T}_{\mathrm{ACE\_word}}$, rather it is $\mathrm{T}_{\mathrm{ACE\_word}}/\mathrm{AVF}_{\mathrm{Cache}}$ to correctly account for the total benchmark execution time, which also includes unACE time. This effect is illustrated in Figure 6, where each binomial attempt should include two different intervals: average ACE time and unACE time. We can then compute $\mathrm{MTTF}_{\mathrm{SW}}$ as:

$$\mathrm{MTTF}_{\mathrm{SW}} = \frac{1}{P_{DUE}} \times \frac{\mathrm{T}_{\mathrm{word}}(\mathrm{ACE})}{\mathrm{AVF}_{\mathrm{Cache}}} [\mathrm{sec}] \qquad (23)$$

We calculated the $\mathrm{MTTF}_{\mathrm{SW}}$ and converted it to $\mathrm{FIT}_{\mathrm{SW}}$ and show it in the 4th column of Table 4. As can be seen, the new approximate model that takes the SECDED correction effects into account yields much better estimates of the DUE FIT rates compared to prior approaches such as [21]. However, the approximate method still overestimates the DUE FIT rates as compared to PARMA by 1 to 2 orders of magnitude. This residual error comes from various sources. For instance, $\mathrm{T}_{\mathrm{ACE\_word}}$ is the average over time and words instead of the instantaneous ACE time per each word. Because the right side of (23) is a convex function, the average of the function values calculated for each interval is greater or equal to the function value calculated for the average of all intervals, meaning that the estimated MTTF from (23) can be smaller than the accurate estimate.

To the best of our knowledge, PARMA is the *only* framework that can accurately measure the SDC FIT rates or DUE FIT rates of SRAM arrays that are protected by various protection schemes, by correctly addressing the non-linear effects caused by error correction.

5. RELATED WORK

Recognizing the drawbacks of fault injection simulations, researchers have proposed methods to accelerate reliability estimation, particularly for processor internal buffers such as ROBs and LSQs. In this regard, two approaches are particularly relevant: SoftArch [13] and AVF [16]. These techniques use a simple fault generation model assuming that all SEUs are converted to faulty bits; thus soft errors are Poisson distributed. This approximated generation model [12] is valid in environments dominated by single-bit errors since the expected residence time of bits in the target structures is extremely short and hence the likelihood of multiple errors to a word during its vulnerability window is negligibly small. Both approaches focus on measuring SDC FIT rates without error protection scheme. PARMA is a complementary approach targeting large memory structures where the vulnerability window may extend to millions of cycles and hence demands more extensive analytical models. PARMA models soft errors with a binomial process which can account for all possible spatial or temporal patterns of faults over a period of time. This detailed model allows PARMA to consider and quantify the effect of multi-bit error protection schemes.

Studies in [17][21] developed approximate analytical models to estimate the expected time for two-bit errors in a word, in order to decide on an effective scrubbing rate. These methods consider fault rates closer to *intrinsic* rates as mentioned in the previous section. The expected time that any word may have *exactly* two temporal faulty bits without considering masking effects conservatively estimates the scrubbing interval so as to prevent the second fault from occurring. But in the presence of error protection schemes caches can have non-trivial derating factors as reported in [4] and as confirmed by PARMA. Resultantly, we can relax the scrubbing rate significantly based on the much lower DUE FIT rates obtained with PARMA. Further scrubbing rate relaxation may therefore be possible.

Under low operating voltages, gates behave erratically due to process variations and their failure rate soars by nine orders of magnitude [7]. In [27], the authors showed that the probability of having retention errors increases by multiple orders of magnitude when reducing the refresh rates of eDRAM caches to save power. PARMA is applicable to these techniques, although we only have demonstrated PARMA to measure soft error rates in SRAM structures.

The study in [5] showed that the DUE FIT rate of write-back caches increases superlinearly when the cache size is doubled due to the parity-protected TAG vulnerability on dirty lines. In their work, the authors introduced the notion of DUE AVF which is different from the well-known (SDC) AVF. The DUE AVF is empirically measured from real machine tests by reading error logs on a real system exposed to accelerated beam tests. We

believe that PARMA can be extended to include reliability effects on TAG and state bits but this requires significant future research.

6. CONCLUSION AND FUTURE WORK

The cache reliability problem caused by soft errors has become a cause for significant concern leading to the introduction of sophisticated error protection schemes. Cache FIT rates must be accurately measured in order to evaluate and compare the merits of various protection schemes. PARMA enumerates all possible temporal and spatial fault patterns to address temporal MBEs and derives accurate expected failure rates. The rigorous, fundamental approach adopted in PARMA is potentially applicable to the reliability benchmarking of any memory structure with any protection scheme. PARMA models provide an accurate quantitative basis to trade-off various design targets such as reliability, power and performance. Furthermore, PARMA can measure the reliability of caches protected by such schemes as CPPC [15] that are hard to model using classical approaches.

PARMA can work as a standard to check the accuracy of popular techniques used up to now to evaluate the reliability of caches. We have shown that non-linear effects distort the observations obtained from highly accelerated fault injection simulations. Hence the results from highly accelerated reliability simulations cannot be trusted in actual design decisions, even from a purely qualitative viewpoint. Using PARMA we have also shown that prior approximate analytical models to evaluate the FIT rates of memories with error correction codes were totally inaccurate. Using the insights gained from the comparison of the PARMA model with prior approximate analytical models we have introduced a new approximate analytical model based on a refined AVF methodology to estimate the DUE FIT rate of SECDED protected caches. The FIT rates obtained with our new model are closer to PARMA FIT rates although they are still off by one or two orders of magnitude. The advantage of the new approximate model over PARMA models is that it can be evaluated much faster than PARMA and thus is worth considering for rapid comparisons of FIT rates with different protection schemes.

In our simulator, sim-outorder is augmented with a Binary Search Tree (BST) to keep track of the entire memory footprint of a program. In this structure, the simulator records the vulnerability clock at the byte level. As a result, the memory overhead of PARMA is about 35x the total simulated memory footprint. Additionally the computation of probabilities at each cache access has a complexity of $O(n^3)$, where n is the number of bits in the access domain. As a result of the lengthy, random searches in the large data structure keeping track of vulnerability clocks and of the complex floating-point operations computing probabilities, the sim-outorder/PARMA simulation is, on the average, slower than the basic sim-outorder simulation by a factor of about 25x for 100M SimPoint simulations.

We are currently exploring sampling techniques to speedup PARMA using cache set sampling [14] and access interval sampling. In access interval sampling, we select the intervals between cache accesses that we keep track of at random. This reduces both the size of the data structure keeping track of vulnerability clocks and the number of probability calculations. It reduces memory pressure and speeds up the simulations. Preliminary results based on set sampling only show great promise. By sampling 10% of the cache sets we have been able to cut the slowdown factor down to seven, with an acceptable average loss of accuracy of 6% on the FIT rate across the benchmarks we have considered.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. CSR-0615428 and CCF-0834798.

8. REFERENCES

[1] H. Asadi, V. Sridharan, M.B. Tahoori, and D. Kaeli. Vulnerability analysis of L2 cache elements to single event upsets. In *Proceedings of the Design, Automation and Test in Europe,* 1276-1281, 2006.

[2] S. Baeg, S. Wen, R. Wong, SRAM Interleaving Distance Selection With a Soft Error Failure Model, *Nuclear Science, IEEE Transactions on* , vol.56, no.4, pp.2111-2118, Aug. 2009

[3] M.A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A.F. Witulski, J. Sondeen, S.D. Stansberry, J. Draper, L.W. Massengill, J.N. Damoulakis. Models and Algorithmic Limits for an ECC-Based Approach to Hardening Sub-100-nm SRAMs. In *IEEE Transactions on Nuclear Science*, 54(4), 935-945, 2007.

[4] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. Mukherjee, R Rangan, Computing Architectural Vulnerability Factors for Address-Based Structures, In *Proceedings of the 32nd International Symposium on Computer Architecture*, 532-543, 2005

[5] Arijit Biswas, Charles Recchia, Shubhendu S. Mukherjee, Vinod Ambrose, Leo Chan, Aamer Jaleel, Mike Plaster, and Norbert Seifert, Explaining Cache SER Anomaly Using Relative DUE AVF Measurement, In *Proceedings of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2010

[6] D. Burger and T. M. Austin. The SimpleScalar Tool Set Version 2.0. Technical Report 1342, CS Department, University of Wisconsin-Madison.

[7] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings of the 36th International Symposium on Microarchitecture*, pages 7-18, 2003.

[8] K. Flautner, N.S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings of the 29th International Symposium on Computer Architecture*, 148-157, 2002.

[9] E. Ibe, S.S. Chung, S. Wen, H Yamaguchi, Y Yahagi, H Kameyama, S Yamamoto, T Akioka, "Spreading Diversity in Multi-cell Neutron-Induced Upsets with Device Scaling," *Custom Integrated Circuits Conference*, 2006. CICC '06. IEEE , vol., no., pp.437-444, 10-13 Sept. 2006

[10] J.W. Kellington, R. McBeth, P. Sanda, and R.N. Kalla. IBM POWER6 Processor Soft Error Tolerance Analysis Using Proton Irradiation. In *Proceedings of the 3rd IEEE Workshop on Silicon Errors in Logic System Effects*, 2007.

[11] M. Li, P. Ramachandran, R.U. Karpuzcu, S.K.S Hari, S. Adve. Accurate Microarchitecture-Level Fault Modeling for Studying Hardware Faults. In *Proceedings of the*

International Conference on High Performance Computer Architecture, 105-116, 2009.

[12] X. Li, S. Adve, P. Bose, and J.A. Rivers. Architecture-Level Soft Error Analysis: Examining the Limits of Common Assumptions. In *Proceedings of the International Conference on Dependable Systems and Networks*, 266-275, 2007.

[13] X. Li, S. Adve, P. Bose, and J.A. Rivers. SoftArch: An Architecture Level Tool for Modeling and Analyzing Soft Errors. In *Proceedings of the International Conference on Dependable Systems and Networks*, 496-505, 2005.

[14] Liu, L.; Peir, J.K.; , Cache sampling by sets, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.1, no.2, pp.98-105, Jun 1993

[15] M. Manoochehri, M. Annavaram, M. Dubois. CPPC: Correctable Parity Protected Cache, In *Proceedings of the 38th International Symposium on Computer Architecture*, 2011

[16] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T.Austin. A systematic methodology to calculate the architectural vulnerability factors for a high-performance microprocessor. In *Proceedings of the 36th International Symposium on Microarchitecture*, pages 29-40, 2003.

[17] S. S. Mukherjee, J. Emer, T. Fossum, and S. K. Reinhardt. Cache Scrubbing in Microprocessors: Myth or Necessity? In *Proceedings of the 10th IEEE Pacific Rim Symposium on Dependable Computing*, 37-42, 2004.

[18] R. Naseer, Y. Boulghassoul, J. Draper, S. DasGupta, A. Witulski. Critical Charge Characterization for Soft Error Rate Modeling in 90nm SRAM. In *Proceedings of the IEEE Symposium on Circuits and Systems*, 1879-1882, 2007.

[19] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder. Using SimPoint for Accurate and Efficient Simulation. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 318-319, 2003.

[20] M. Rebaudengo, M. S. Reorda, and M. Violante. An Accurate Analysis of the Effects of Soft Errors in the Instruction and Date Caches of a Pipelined Microprocessor. In *Proceedings of the Design, Automation and Test in Europe*, 602-607, 2003.

[21] A.M.Saleh, J.J.Serrano, and J.H.Patel. Reliability of Scrubbing Recovery Techniques for Memory Systems. In *IEEE Transactions on Reliability*, 39(1), 114-122, 1990.

[22] Semiconductor Industries Association, International Technology Roadmap for Semiconductors.

[23] Sridharan, V., Asadi, H., Tahoori, M. B., and Kaeli, D. 2006. Reducing Data Cache Susceptibility to Soft Errors. *IEEE Trans. Dependable Secur. Comput*. 3, 4 Oct. 2006, 353-364.

[24] Tosaka, Y., Satoh, S., Itakura, T., Suzuki, K., Sugii, T., Ehara, H., Woffinden, G.A., Cosmic ray neutron-induced soft errors in sub-half micron CMOS circuits, *Electron Device Letters*, IEEE , vol.18, no.3, pp.99-101, Mar 1997

[25] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. Cacti 5.1. HP Report HPL-2008-20.

[26] C. Weaver, J. Emer, S.S. Mukherjee, and S.K. Reinhardt. Techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor. In *Proceedings of the International Symposium on Computer Architecture*, 264-274, 2004.

[27] Wilkerson, C., Alameldeen, A. R., Chishti, Z., Wu, W., Somasekhar, D., and Lu, S. 2010. Reducing cache power with low-cost, multi-bit error-correcting codes. In *Proceedings of the 37th Annual international Symposium on Computer Architecture*. ISCA '10., 83-93.

[28] Wunderlich, R. E., Wenisch, T. F., Falsafi, B., and Hoe, J. C. "SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling." In *Proceedings of the 30th Annual international Symposium on Computer Architecture*. ISCA '03., 84-97.

[29] B. Zandian, M. Annavaram. Cross-layer Resilience Using Wearout Aware Design Flow. *Dependable Systems and Networks* (DSN), 2011.

[30] B. Zandian, W. Dweik, S. Kang, T. Punihaole, and M. Annavaram. WearMon: Reliability Monitoring Using Adaptive Critical Path Testing. *Dependable Systems and Networks* (DSN), pages 151-160, 2010.