MACAU: A Markov Model for Reliability Evaluations of Caches Under Single-bit and Multi-bit Upsets

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

Abstract

Due to the growing trend that a Single Event Upset (SEU) can cause spatial Multi-Bit Upsets (MBUs), the effects of spatial MBUs has recently become an important yet very challenging issue, especially in large, last-level caches (LLCs) protected by protection codes. In the presence of spatial MBUs, the strength of the protection codes becomes a critical design issue. Developing a reliability model that includes the cumulative effects of overlapping SBUs, temporal MBUs and spatial MBUs is a very challenging problem, especially when protection codes are active.

In this paper, we introduce a new framework called MACAU. MACAU is based on a Markov chain model and can compute the intrinsic MTTFs of scrubbed caches as well as benchmark caches protected by various codes. MACAU is the first framework that quantifies the failure rates of caches due to the combined effects of SBUs, temporal MBUs and spatial MBUs.

1. Introduction

Caches occupy more than half of the chip real estate in today's microprocessors and their reliability is therefore a critical design issue. The charge stored in a memory cell such as an SRAM cell decreases with every process generation. As a result, memories become more and more susceptible to random, transient errors called *soft errors*. The vulnerability of a memory cell to soft errors increases further when caches operate at lower voltage using techniques such as drowsy supply voltages [8] and sub-threshold voltage operation [7] to reduce static power dissipation. A system's soft error reliability must be measured during its design phase to determine appropriate error protection mechanisms for every component. Otherwise, building a system that meets reliability specifications is a guessing game.

Soft errors are usually due to neutron or alpha particle strikes. A single strike of such particles causes an event called Single Event Upset (SEU) if the event disturbs the content of the memory. An SEU can flip one bit cell (SBU: Single-Bit Upset) or multiple bit cells (MBU: Multi-Bit Upset). While the energy transferred by a particle strike remains the same over time, memory cells become geometrically smaller and hold less charge as technology advances, making them rapidly more vulnerable to MBUs. A *spatial* MBU is an MBU resulting from an SEU. By contrast, an MBU resulting from multiple SEUs over time is called a *temporal* MBU. The silicon industry projects that, starting from year 2015, all SRAM arrays will be mostly affected by spatial MBUs [25]. Therefore system designers must prevent spatial MBUs from corrupting correct system operation [2][3][18][19]. Many studies have characterized and modeled spatial MBUs [9][12][14][17][21][22][23].

Circuit designers usually concentrate on measuring the *intrinsic FIT* (Failures-in-Time of 10^9 hours) *rate* or *intrinsic MTTF* (Mean-Time-to-Failure), assuming that *all* the memory cells in the structure are *always* critical for correct execution. The intrinsic MTTF is the expected time until the memory cells fail due to particle hit(s) when all the cells are holding information critical to the computation and are not accessed until they fail. This measurement results in an over-estimation of the vulnerability of a memory system because during a program execution not all bits are critical to correctness, and moreover every access to them may activate a protection code and correct the faulty bit(s). Still, this intrinsic MTTF gives a first, rough estimate of the vulnerability of memory structures and thus is a widely accepted quantification of their reliability. Intrinsic MTTFs are independent of the workload and thus particularly relevant to designers of general-purpose systems.

Reliability benchmarking yields better estimates of the number of bits critical to execution correctness by factoring in a de-rating factor called *Architectural Vulnerability Factor* (AVF) [5]. Recently, a rigorous model called PARMA [26] which includes the effects

of SBUs and temporal MBUs was proposed in order to correct the rough approximations inherent in classical approaches based on AVF analysis [1][5]. With PARMA, a designer can benchmark the vulnerability of caches protected by codes such as SECDED or parity. Such reliability benchmarking helps designers decide on protection codes amongst multiple design options. To the best of our knowledge, no study has been published on benchmarking soft-error reliability when SBUs, spatial MBUs and temporal MBUs coexist.

In this paper, we propose a model called MACAU (a MArkovian model for reliability evaluations of CAches under single-bit and multiple-bit Upsets). MACAU can compute the intrinsic FIT rate or the intrinsic MTTF, as well as benchmark the soft-error reliability of caches with various protection codes. The contributions of our paper are as follows.

- We introduce MACAU, a model that measures the reliability of caches protected by various protection codes when SBUs, temporal MBUs and spatial MBUs coexist.
- We demonstrate that MACAU can compute the intrinsic MTTF of SEC-protected caches under SBUs and of DEC-protected caches under SBUs and 2BUs (2-Bit Upsets) and we compare its results to previously proposed models [21][23]. However MACAU goes beyond these models to compute the intrinsic MTTF when SBUs, temporal MBUs and various spatial MBUs coexist, which previous models cannot compute.
- We demonstrate how MACAU benchmarks the FIT rates of caches with various protection codes on a set of benchmark programs.

The rest of the paper is organized as follows. We review the related work in Section 2. In Section 3, we define the terminology and expose the MACAU model in detail under a physical model of SEUs. We also show how MACAU computes the intrinsic MTTF and benchmarks caches with various protection schemes when there are at most two SEUs. Then in Section 4, we show the intrinsic MTTFs and benchmarking results of MACAU and verify them with previous state-of-the-art methods. In Section 5, we discuss how to deal with several SEUs. We conclude in Section 6.

2. Related work

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 the simulation of the system under study [13]. While this framework is conceptually simple and can model any type of error, including spatial MBUs, it requires an astronomical number of extremely long simulation experiments to obtain statistically meaningful results. Accelerating the fault injection rate is a possible solution to avoid such long and expensive simulations. However large quantitative and qualitative distortions due to simulation acceleration were reported in [26].

Several recent studies have focused on computing the FIT rate due to SBUs [1][5]. These studies are based on AVF analysis and assume that no more than one SBU can hit a set of bits during the time it resides in a processor structure and are widely accepted for benchmarking internal processor storage buffers such as load/store queues (LSQs) or reorder buffers (ROBs). The probability of temporal MBUs in caches, especially large last-level caches, is much higher than in processor buffers since blocks can reside in cache for millions of cycles between two consecutive accesses to them. PARMA [26] models the effect of temporal MBUs. It benchmarks ECC-protected caches and demonstrates how designers can address the cost/benefit tradeoffs of various protection schemes. The effects of temporal MBUs were also studied in [18][23] but only for estimating the intrinsic MTTF on SEC-protected caches under SBUs or on DEC-protected caches under up to two SBUs. By ignoring the effect of activating protection code and correcting faulty bit(s) whenever an access is made to the cache, these approaches cannot benchmark caches.

Due to the growing impact of spatial MBUs, many recent studies including [2][3][17][21][22] estimate the intrinsic MTTF of spatial MBUs in SRAM structures. The study in [2] uses a compound Poisson process [22] to model spatial MBUs and to decide on the interleaving distance of SECDED code to suppress spatial MBUs, but this model cannot benchmark caches. In [17] and [21], it is reported that the intrinsic MTTF in the presence of spatial MBUs can be approximated by the model in [23], which computes the intrinsic MTTF under SBUs only. This is clearly an approximation, but these observations show that state-of-the-art soft-error benchmarking frameworks like PARMA are credible even in the presence of spatial MBUs.

Studies in [9][12][14] based on beam-injection experiments report how spatial MBUs affect real chips. Such data is critical to model the complex fault patterns of spatial MBUs. A recent study in [14] observed that patterns of spatial MBUs in SRAM arrays built with a 40nm deep-n-well process are highly affected by the placement of N and P wells. In most cases, spatial MBUs are clustered as a connected group and are parallel to wells. When the wells are placed in the bitline (vertical) direction, at most two

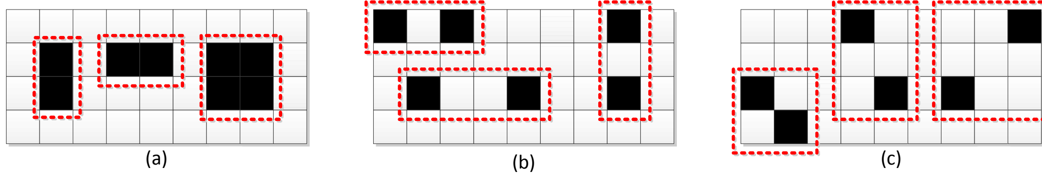

Figure 1. Patterns of spatial MBUs in an SEU

bit-upsets are observed in the wordline (horizontal) direction while up to four upsets are observed in the bitline direction. Similar observations were made for 65nm bulk SRAMs in [27].

Integrating various geometric fault patterns resulting from interactions of SBUs, spatial MBUs and temporal MBUs in a single model is an unanswered, challenging problem so far. MACAU is a unified framework that models SBUs and MBUs at the same time. It not only measures the intrinsic MTTF of caches with various protection schemes with or without scrubbing, but it also benchmarks the FIT rate of such caches under real workloads.

3. The MACAU model

Here we explain the MACAU model in detail. We start by defining terms and explaining how to compute reliability metrics in Section 3.1. In Section 3.2, we show the cache configurations and the physical model of SEUs, especially the kind of patterns of spatial MBUs considered in MACAU. Then we describe the MACAU model in Section 3.4, with the assumptions made in Section 3.3.

3.1 Setups

PARMA [26] demonstrates a well-defined framework to compute the FIT rates of memory systems in benchmarks by counting the expected number of errors in actual program executions. We adopt the same setup in MACAU. If the fault in an SRAM cell propagates to an outer scope (for example, if a faulty bit in the L2 cache is copied to the L1 cache and is not detected or is detected but is uncorrectable), then the fault becomes an error. In the case of a detected but uncorrectable error, a fatal exception is raised and the system halts. In the case of an undetected error, the system may crash due to various reasons. When such an event happens, we call this event a *failure*. Failures can be categorized further into TRUE DUE (Detected Unrecoverable Error), FALSE DUE or SDC (Silent Data Corruption) error. Among all failures, if a failed bit is *consumed* by the processor as a committed instruction or as a committed operand [26], the result is a TRUE DUE or an SDC. An L2 block is copied to L1 in block granularity but the processor accesses L1 in word granularity. Thus finding TRUE DUEs or SDCs of L2 blocks requires tracking accesses until L1 blocks are evicted, as was done in [26].

In order to benchmark reliability in a system, we measure the probability of any component failure in a system in every cycle, assuming that no more than one failure of the same component may occur in the same cycle. The MACAU framework can benchmark the reliability of memories, as well as estimate their intrinsic MTTF, which does not involve benchmark programs and assumes that all cached data are critical.

Let's index each processor cycle by j (where $1 \leq j \leq$

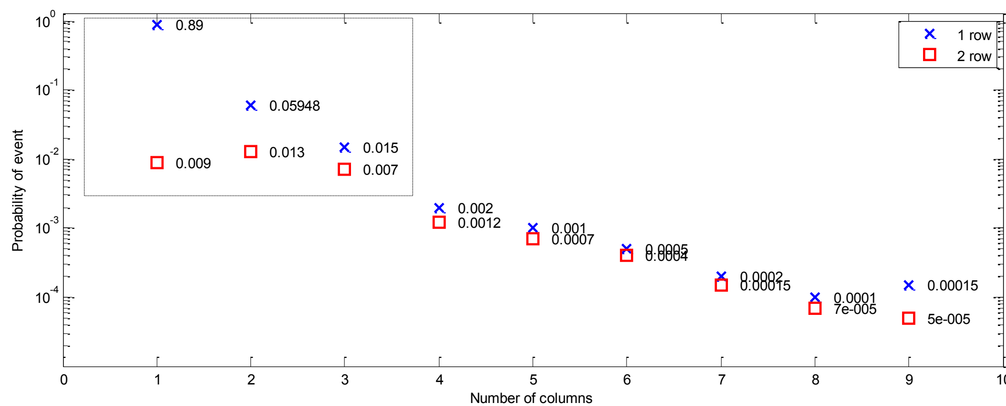

Figure 2. The probability distribution of MBUs for omni-directional galactic cosmic rays [27] in the cache layout shown in Figure 3(b).

T_{exe}). Then $h_{ERR}(j)$, the discrete time failure probability mass at the jth cycle, is defined as:

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

h_{ERR} is the *conditional* probability (also called "hazard mass") that a failure of type *ERR* (*ERR* can be SDC, TRUE DUE or FALSE DUE) has occurred at the jth cycle, given that the system survived all types of failures up to the jth 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 by simply scaling time. Then, the average FIT rate for a set of applications running one after another independently on a processor is calculated as:

$$\overline{FIT_{ERR}} = \sum_{\forall \text{workload } w} f_w \times FIT_{w,ERR} = \sum_{\forall \text{workload } w} f_w \times \frac{E_w[ERR] \times 3600 \times 10^9}{T_{\text{exe},w} \times CyclePeriod} \quad (3)$$

where f_w ($=T_{exe,w}/T_{exe,all}$) is the fraction of time taken by the execution of workload w. $FIT_{w,ERR}$ is the FIT rate extrapolated from $E_w[ERR]$. We will use (3) to report our benchmarking results in FIT rates in Section 4.3. One important observation is that accumulating the expected number of failures while continuing the benchmark is equivalent to classical survival analysis where failed components are replaced. This is consistent with the nature of transient faults.

We adopt two terms from [26].

Vulnerability clock cycles (VCCs): The vulnerability clock cycles of a bit are the processor clock cycles during which the bit resides in the target memory structure to benchmark (i.e., a cache) between two accesses to it. If a block stays for *J* cycles in the target cache before eviction after its last access and then is reloaded later in the target cache, these *J* cycles are VCCs since it is possible that a particle strike during the *J* cycles and the faulty bits are *consumed* later. VCC tracking is equivalent to lifetime analysis for ACE cycles [5].

Protection domain (PD): The protection domain is the set of bits covered by the protection code. For example, if odd parity is calculated over the block, bits in the entire block form the PD. If a SECDED code protects a word, the PD is the word. In the balance of this paper, the protection domain is a word of 32 bits.

3.2 Physical model

3.2.1 SEU model

In MACAU one clock cycle is the minimum unit of time. The probability that a PD (a word of 32 bits) has an SEU in one cycle is denoted by $p_{SEU,PD}$. All clock cycles are independent: whether a PD is struck or not at clock cycle *j* is independent of whether or not it was hit in any previous cycles from 1 to *j*-1. This means that SEUs are a renewal process (in fact a Poisson process) so that the probability distribution of SEUs is always the same after every cycle.

In order to compute p_{SEU_PD}, the SEU rate obtained from the ITRS roadmap [25] is first scaled for the PD and for one clock cycle. The intrinsic SEU rate is 1,150 SEUs per 10^9 hours for 1Mbit SRAM array [25]. Throughout this paper the PD is a word and we use the value p_{SEU_PD} = 3.2496E-24 for a 3GHz processor.

3.2.2 Spatial MBU model

Several experimental observations have been reported recently on the distribution of spatial MBUs in real chips from beam injection experiments [9][12][14]. In Figure 1, each white cell shows undisturbed bit cells and black cells show flipped bit cells due to particle hits. Each dotted rectangle includes a spatial MBU caused by an SEU. One noticeable observation from recent studies is that spatial MBUs are usually compact, i.e. faults are confined to a contiguous rectangle. Referring to Figure 1, the fault pattern in (a) happens most of the time while the fault patterns in (b) and (c) are observed very rarely [9][14]. As a result, in general [27], the dimension of a spatial MBU event is specified as follows:

$$\text{MBU Dimension} = N_{row} \times N_{column} \quad (4)$$

where N_{row} and N_{column} are the horizontal and vertical dimensions of the pattern of bit cells flipped due to a spatial MBU event. For example, the leftmost event in Figure 1(a) is a 2×1 spatial MBU. Rows and columns in (4) usually point to wordline and bitline directions respectively. Note that MACAU does not ignore cases such as in Figure 1(b) or (c). How to address such cases will be discussed in Section 3.3.

In this paper, we adopt the probability distribution of MBUs reported in [27], which is shown in Figure 2. In this figure the probability of occurrence is attached to each data point. At most two rows and up to nine columns are affected by an SEU. However, because the probability of having more than four columns is very small (below 1E-3), we concentrate on MBU patterns included in the dotted square in Figure 2, for

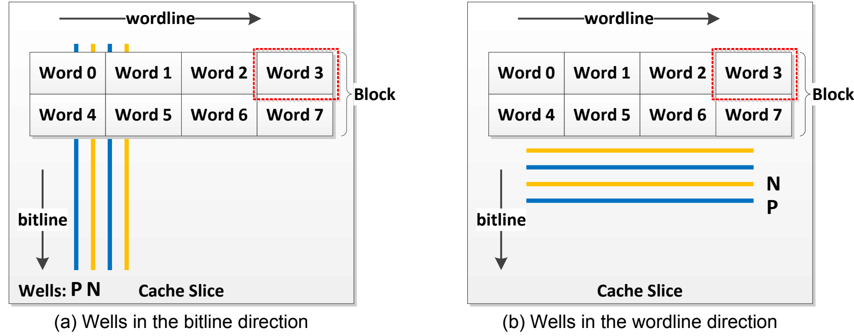

Figure 3. Cache configurations. Each dotted rectangle shows a protection domain

the purpose of demonstrating MACAU. We address spatial MBUs of 1×1, 1×2, 1×3, 2×1, 2×2 and 2×3 only in this paper. Distribution is weighted for those patterns such that the sum of their probability equals one. However, as explained in Section 3.4, MACAU can deal with other geometries.

3.2.3 Cache configuration

Figure 3 shows the cache configuration throughout this paper. The cache is divided into several slices to reduce wordline capacitance. In each slice a word in the block is protected by a protection code such as odd parity, SECDED, DECTED or TECQED. Because spatial MBUs may overlap across multiple rows, one single SEU can affect two PDs. The same is true of two different words in the same row, i.e. word 0 and word 1 if the MBU that flips multiple columns happens to hit the border between those two words.

In this configuration, N- and P-wells can be placed in the bitline direction as in Figure 3(a) or in the wordline direction as in Figure 3(b). A study in [14] reports that spatial MBUs are usually observed in the direction of the wells because parasitic bipolar transistors contribute mostly to MBUs and only NPN transistors turn on in deep-N-well processes. At most two flips have been observed in a spatial MBU in the direction perpendicular to wells. At most two bit flips are observed in wordline direction in Figure 3(a) or in bitline direction in Figure 3(b). Our SEU model focusses on caches laid out like in Figure 3(b).

3.3 Modeling spatial MBUs

In this section we expose the complexity of a rigorous spatial MBU model and the relaxations to make the model feasible.

3.3.1 Complexity due to spatial MBU patterns

Figure 4 shows what the resulting patterns may be after two spatial MBUs overlap. Dotted rectangles frame the PD (word) in the figure. Faulty bits are darkened and fault patterns are circled. Spatial MBUs can happen inside the PD (as in fault patterns 4, 5, 6) or across PDs (as in fault patterns 1, 2, 3, 7). At first an SEU (such as fault patterns 1 to 4) happens as shown in Figure 4(a). Then a second SEU (such as fault patterns 5 to 7) strikes the same PD as shown in Figure 4(b). The faulty bit patterns resulting from the superimposition of the two MBUs are shown in Figure 4(c). We make several observations.

i. Overlap of MBU footprints: If two spatial MBUs overlap, the bit cells flipped by the first SEU can be flipped back to their correct value. Thus two spatial MBUs may end up having less faulty bits than the sum of their footprints.
ii. Two overlaps in the same PD: For example in PD#1, the second bit is first flipped by SEU 1. Then, due to SEU 5, this second bit is flipped back to a correct state. However, the first and third bits remain faulty. The resultant faulty bit pattern in PD#1 now has two disjoint SBUs.
iii. Vertical overlaps across PDs: If a spatial MBU overlaps vertically across PDs as in faults 1, 2, and 7, it can be considered as two spatial MBUs, each one in a different PD. For example, SEU 1 can be counted as two SBUs, one in PD#1 and one in PD#5, and SEU 7 can be counted as two spatial 1×3 MBUs in PD#4 and PD#8.
iv. Horizontal overlaps across PDs: This is essentially the same case as the above case (iii) but is a little more subtle. SEU 3 is harder to deal with as it can be divided into one spatial MBU of 1×2 in PD#3 and one SBU in PD#4. We call this an *edge effect*.

Major challenges in modeling spatial MBUs come from cases (ii) and (iv) above. In the following section we discuss how we deal with such cases.

3.3.2 Simplifying the spatial MBU problem

3.3.2.1 Case: second SEU to the same word

SEUs happen very rarely in a realistic environment. ITRS reports that the SEU rate for an 1Mbit SRAM array is around 1,150 ~ 1,300 SEUs per 10^9 hours [25]. This suggests that, during the time a system is running an application, a PD is extremely unlikely to be hit by

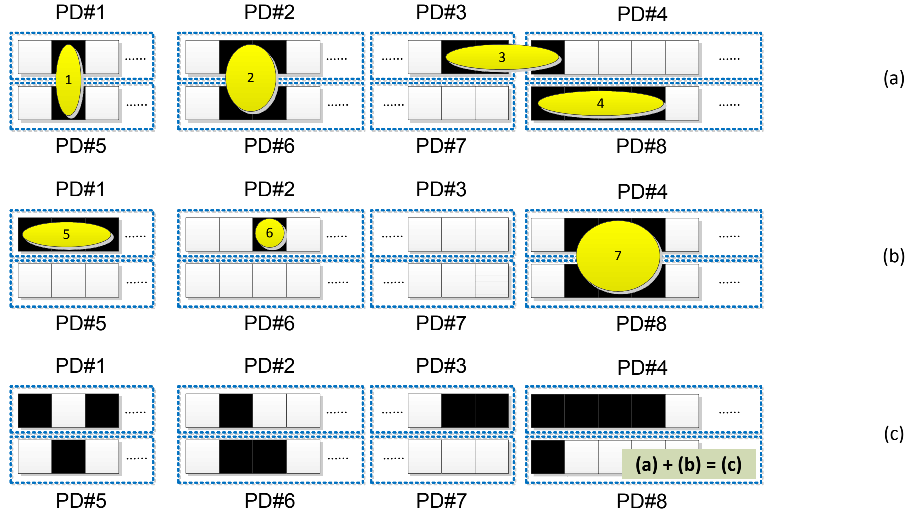

Figure 4. Overlapping effects of spatial MBUs

more than one or two SEUs. As an example, using the Poisson probability mass function, the probability that any one word is hit three times (three SEUs) during one billion cycles in a 3GHz processor is 5.7190E-45.

As discussed before, most spatial MBUs are contiguous, rectangular-shaped clusters. Therefore, if a word has k faulty bits when the second SEU hits it, the pre-existing MBU pattern due to the first SEU is most likely $1\times k$ and we are reduced to calculating the probabilities of various overlapping scenarios between the new SEU and any of k existing contiguous faulty bits.

3.3.2.2 Case: edge effect

Usually a PD is much wider than most frequent spatial MBUs (up to 3 bit per SEU as shown in Figure 2)). If a spatial 3BU spans over two PDs horizontally, it would increase the vulnerability of up to two bits sitting next to the borders of PDs. Therefore, its impact is insignificant since the probability of having such cases is around 6% (= ${}_{2}\mathrm{C}_{1}/{}_{32+2-2}\mathrm{C}_{1}$) and the increased vulnerability due to such 6% of events is capped by at most two bits among 32 bits in a PD. Therefore, we simply ignore the edge effect exposed in case (iv) in Section 3.3.1.

3.3.2.3 Case: non-contiguous spatial MBUs

Since SEUs happen very rarely during the execution of a typical program, most words experience at most one SEU. Today's data on MBUs ([9][12][14]) reports that non-contiguous patterns such as in Figures 1(b) and (c) are extremely rare.

However if non-contiguous patterns are possible, MACAU can still address non-contiguous patterns. Since non-contiguous patterns are much rarer than contiguous patterns, we can replace patterns in Figures 1(b) and (c) by those of Figure 1(a) by considering that all the bits inside the entire dotted rectangles in Figures 1(b) and (c) are flipped. As a result, we simplify dealing with extremely rare non-contiguous patterns in the model, with minimal overestimation.

3.3.2.4 Limitations of MACAU

In previous subsections we simplified the problems associated with spatial MBUs. We made two strong assumptions: 1) SEUs happen very rarely in a given domain during the execution time of typical programs, and 2) spatial MBUs have contiguous patterns most of the time. Both assumptions agree with the observations made in chips built with current-generation technology. The MACAU model should be revised in future if new data disagrees with one of these assumptions. However, we believe these two assumptions will remain valid even with future technology because of the following two reasons: 1) even in space under worst-case GEO flare, the SEU rate increases by 10 orders of magnitude [3], which increases the probability of one SEU during one cycle from 10^{-25} to 10^{-15} (which is still extremely small) and 2) spatial MBUs happen because of small cell pitch and the parasitic bipolar transistors in wells; by comparison the location of the diffusion area that collects charges is not significant [24], meaning that non-contiguous patterns are not likely to appear.

3.4 The MACAU model

In this section we explain the MACAU model in detail.

3.4.1 The Markov chain

A transition in the Markov chain occurs in every processor cycle. The state names in the Markov chain are the number of faulty bit in the PD. Hence with a 32-bit word, there are 33 states in the Markov chain.

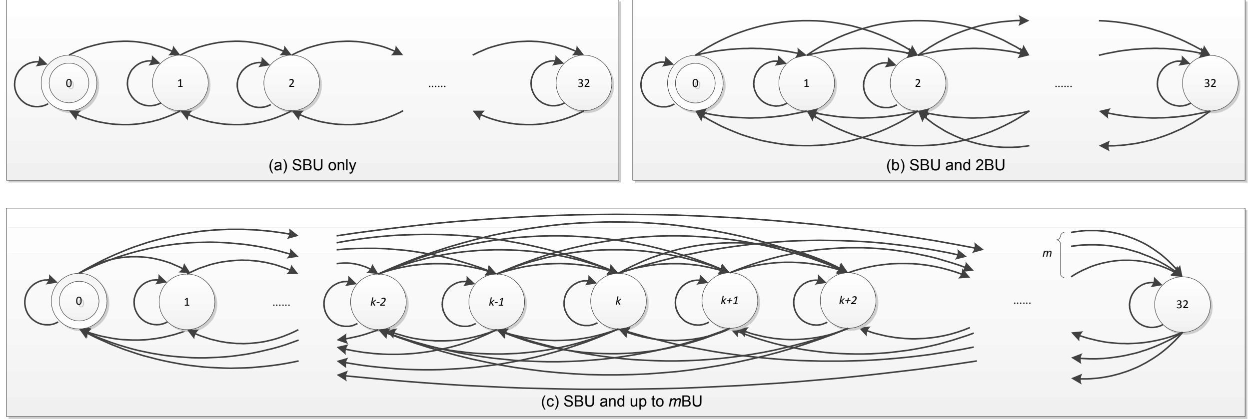

Figure 5. Markov chains representing the effect of SBUs and MBUs in a 32-bit word

Once the Markov chain is built, we can express the probability of having k faulty bit in the PD by computing the transition probability from state 0 to state k after t VCCs. Figures 5(a) and (b) show the transitions of the Markov chain for the number of faulty bits in a 32-bit word PD in the presence of SBUs only and in the presence of SBUs plus 2BUs respectively. In Figure 5(c), we show the generalization of the Markov chain for SEUs with up to m bit upsets. The figure shows the transitions between sets of states $\mathbf{M}=\{k-m, k-m+1, \ldots, k-1, k, k+1, \ldots, k+m-1, k+m\}$. The transition probabilities between a state k and any state (among a total of $2m+1$ states) in $\mathbf{M}$ are specified in a transition matrix $\mathbf{T}$. In general, if an SEU may flip up to m bits in the PD, each row and column of $\mathbf{T}$ has at most $2m+1$ nonzero elements.

A Markov state is *transient* if the probability of *not* returning to it after departing from it is nonzero. A state is called recurrent if it is not transient, meaning it will eventually be revisited. One special case of a recurrent state is an absorbing state. A state is an absorbing state when *no* more transition to other states is possible once that state is visited.

3.4.2 Transition matrix T

Matrix $\mathbf{T}$ contains the state transition probabilities in every processor clock cycle. To build the matrix we start with $p_{\mathrm{SEU_PD}}$, the probability of an SEU in a word in a cycle. Every SEU has some probability of causing one SBU or one spatial MBU with various patterns. The probability distribution is shown in Figure 2, inside the dotted square. We build a probability matrix $\mathbf{D}$ for events confined to the dotted square as:

$$\mathbf{D} = \begin{Vmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \end{Vmatrix} \tag{5}$$

where $d_{a,b}$ is the probability of an SEU being a $a\times b$ spatial MBU. The maximum dimension is 2×3 in the fault pattern. Note that matrix $\mathbf{D}$ could have higher dimension in general. Without loss of generality, we limit the dimension to 2×3 for demonstration purposes.

As discussed in Section 3.3, we can take into account the effect of spatial, compact MBUs with two rows by increasing single-row SEU probabilities. Thus we build the probability of having a 1BU, a 2BU or a 3BU inside the PD (a word) in one cycle from $p_{\mathrm{SEU_PD}}$ and $\mathbf{D}$ as follows.

$$\mathbf{P} = \begin{Vmatrix} p_{1\mathrm{BU}} \\ p_{2\mathrm{BU}} \\ p_{3\mathrm{BU}} \end{Vmatrix} = \begin{Vmatrix} p_{\mathrm{SEU_PD}} \times (d_{1,1} + 2 \times d_{2,1}) \\ p_{\mathrm{SEU_PD}} \times (d_{1,2} + 2 \times d_{2,2}) \\ p_{\mathrm{SEU_PD}} \times (d_{1,3} + 2 \times d_{2,3}) \end{Vmatrix} \tag{6}$$

$d_{2,b}$ (b = 1, 2, 3) is multiplied by two in (6) since two-row spatial MBUs flip bits in two vertically adjacent PDs. In general, if a spatial MBU spans up to z rows, $z\times d_{z,b}$ is added to $d_{1,b}$ (b = 1, 2, …). Matrix $\mathbf{P}$ gives the probabilities of having a 1×1, a 1×2, or a 1×3 MBU in a single word in any processor cycle. In the current MACAU setup we only consider $1\times k$ MBUs in a PD.

We now compute the probabilities of various patterns when two spatial MBUs overlap. Let o be the number of overlapping bits when a spatial qBU hits a word with k flipped bits. Both patterns are made of contiguous bits and a PD has a total of N bits. We compute the probabilities in three cases:

i. If $0 < o = q$: There are a total of ${}_{(k-q+1)}\mathrm{C}_1$ cases because the second qBU must fall into the $1\times k$ contiguous flipped bit pattern. In this case the fault size is reduced by q bits. The number of cases that a qBU falls into N contiguous bits is ${}_{(N-q+1)}\mathrm{C}_1$. The probability of such cases is:

$$p_{ovl(o|q)} = p_{ovl(q|q)} = \frac{{}_{(k-q+1)}\mathrm{C}_1}{{}_{(N-q+1)}\mathrm{C}_1} = \frac{k-q+1}{N-q+1} \tag{7}$$

ii. If $0 < o < q$: For a given o there are only two possible cases because the qBU must cross the boundary at the head or tail of the $1\times k$ contiguous

flipped bit pattern, regardless of the value of o. The probability of such case is:

$$p_{ovl(o|q)} = \frac{2}{_{(N-q+1)}C_1} = \frac{2}{N-q+1} \quad (8)$$

iii. If o = 0: There is no overlap so the second SEU falls on non-flipped bits only. In this case the fault size is increased by q. The probability of such cases can be obtained from (7) and (8):

$$p_{ovl(o|q)} = p_{ovl(0|q)} = 1 - \sum_{l=1}^{q} p_{ovl(l|q)} \quad (9)$$

Let's now build the transition matrix **T** from the matrix **P** and probabilities $p_{ovl(o|q)}$. Among k faulty bits before the qBU arrives, k-o bits remain faulty as o bits flipped back to their correct values. Among the q bits that a qBU flips, q-o bits become faulty so a total of k+q-$2o$ bits are faulty after the arrival of the second SEU (qBU). The transition distance is $d = q-2o$. d is odd iff q is odd regardless of k or o. Likewise, d is even iff q is even regardless of k or o. Thus in the calculation of each element of matrix **T** we need to distinguish between the cases where the transition distance is odd or even.

We start with a $(N+1)\times(N+1)$ zero matrix and then fill the nonzero elements according to (10).

1) For k = 0 to $2m$+1 and $|d| \le m$:

$$\begin{cases} \text{state } k \text{ to } k+d \ (|d|>0): \\ \text{if } k=0, \\ T_{k,k+d} = p_{d\text{BU}} \\ \text{else if } d \text{ is even}, \\ T_{k,k+d} = \sum_{\substack{\forall \text{even } q \text{ such that} \\ |d| \le q \le \min(m,2k+d)}} p_{q\text{BU}} \times p_{ovl\left(\frac{q-d}{2}|q\right)} \\ \text{else if } d \text{ is odd}, \\ T_{k,k+d} = \sum_{\substack{\forall \text{odd } q \text{ such that} \\ |d| \le q \le \min(m,2k+d)}} p_{q\text{BU}} \times p_{ovl\left(\frac{q-d}{2}|q\right)} \end{cases} \quad (10)$$

2) Then for $\forall k$:

$$\begin{cases} \text{state } k \text{ to } k: \\ T_{k,k} = 1 - \sum_{\forall l \text{ such that } 0 \le l \le N \text{ and } l \ne k} T_{k,l} \end{cases}$$

where m is the maximum number of flipped bits in an SEU. $T_{k,k}$ is the probability that no SEU occurs during the cycle or, if an SEU occurs, the overlap results in the same number of faulty bits.

The structure of the $(N+1)\times(N+1)$ matrix **T** is:

$$\mathbf{T} = \begin{Vmatrix} \mathbf{t}_{(2m+1)\times(2m+1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(N-2m)\times(N-2m)} \end{Vmatrix} \quad (11)$$

where **t** is a band matrix and **I** is the identity matrix. Note that the **T** is already in a canonical form.

Note that (10) is correct if we observe at most two SEUs as discussed in Section 3.3.2. The structure of the transition matrix **T** takes this into account too. It is not possible to reach state $2m$+1 from state 0 in two SEUs and therefore states with more than $2m$+1 faults are never visited.

3.4.3 Intrinsic MTTF

With **T**, we can compute the intrinsic MTTF of a protected word (PD) by computing the expected first-passage transition time. The expected first-passage transition time is the expected number of transitions from state u to state v given that the chain has started from state u. The calculation of the expected first-passage transition time is a well-known problem [16].

3.4.3.1 Intrinsic MTTF without cache scrubbing

In a SEC-protected word, any state $k > 1$ is a failure state. Similarly in a DEC and TEC-protected word, any state $k > 2$ and $k > 3$ respectively is a failure state. Because we measure MTTF from the time a word is clean, the intrinsic MTTF is the expected first-passage transition time from state 0 to 2 or above, 3 or above and 4 or above for SEC, DEC and TEC-protected words respectively.

In [11], the computation of the expected first-passage transition time for a finite transient Markov chain is derived by adding absorbing states. Converting a state k to an absorbing state in the transition matrix **T** is done by setting $T_{k,k} = 1$ and $T_{k,l} = 0$ if $l \ne k$.

Submatrix **T'** is obtained from **T** by removing all columns and rows that correspond to absorbing states, as shown below.

$$\text{SEC on word: } \mathbf{T}' = \begin{Vmatrix} T_{0,0} & T_{0,1} \\ T_{1,0} & T_{1,1} \end{Vmatrix}$$

$$\text{DEC on word: } \mathbf{T}' = \begin{Vmatrix} T_{0,0} & T_{0,1} & T_{0,2} \\ T_{1,0} & T_{1,1} & T_{1,2} \\ T_{2,0} & T_{2,1} & T_{2,2} \end{Vmatrix} \quad (12)$$

$$\text{TEC on word: } \mathbf{T}' = \begin{Vmatrix} T_{0,0} & T_{0,1} & T_{0,2} & T_{0,3} \\ T_{1,0} & T_{1,1} & T_{1,2} & T_{1,3} \\ T_{2,0} & T_{2,1} & T_{2,2} & T_{3,3} \\ T_{3,0} & T_{3,1} & T_{3,2} & T_{3,3} \end{Vmatrix}$$

With an absorbing Markov chain **T**, the probability that a state remains in a transient state monotonically decreases, and therefore submatrix $(\mathbf{T}')^t \to 0$ as $t \to \infty$. That is, this **T'** is a transient matrix.

The matrix **I-T'** has an inverse matrix **N** and $\mathbf{N}=\mathbf{I}+\mathbf{T}'+(\mathbf{T}')^2+\ldots$. Because $(\mathbf{T}')^t \to 0$ as $t \to \infty$, **N** is the summation of transition probabilities that does not grow to infinity. $N_{u,v}$ is the expected number of times

Table 1 Examples of h_{ERR} calculation of various protection schemes from $\mathbf{S}(t)$

Protection	No protection	Odd parity		SECDED		DECTED		TECQED	
ERR	SDC	DUE	SDC	DUE	SDC	DUE	SDC	DUE	SDC
h_{ERR}	$\sum_{\forall k, k>0} S_{0,k}$	$\sum_{\forall \text{odd } k>0} S_{0,k}$	$\sum_{\forall \text{even } k>0} S_{0,k}$	$S_{0,2}$	$\sum_{\forall k, k>2} S_{0,k}$	$S_{0,3}$	$\sum_{\forall k, k>3} S_{0,i}$	$S_{0,4}$	$\sum_{\forall k, k>4} S_{0,k}$

the chain is in state v given that it starts in state u before the chain is absorbed. Matrix **N** is called a fundamental matrix of **T**' and is:

$$\mathbf{N} = \left(\mathbf{I} - \mathbf{T}^{'}\right)^{-1} \quad (13)$$

The expected first-passage transition time, i.e. the expected time before the chain is absorbed, from any state can be computed using **N** by summing all the expected number of times a chain stays in transient states before being absorbed:

$$\boldsymbol{f} = \mathbf{N} \cdot \boldsymbol{w} \quad (14)$$

where $\boldsymbol{w}$ is a column vector filled with 1's.

$\boldsymbol{f}$ is a column vector and the uppermost element f_0 gives the expected first-passage transition time of the word if the chain started in the clean state. This measures the time the chain takes until it reaches an absorbing (failing) state when it starts in clean state 0. Thus, we get the intrinsic MTTF of the word in the absence of scrubbing by multiplying the clock cycle period and f_0.

3.4.3.2 Intrinsic MTTF with cache scrubbing

MACAU can model stochastic scrubbing. Let's say that L is the mean scrubbing interval. We can include the scrubbing effect in **T** in the Markov chain as follows:

$k = 1, \ldots, v$ (code corrects up to v bits);

$$k \text{ to } 0\text{: } T_{k,0} = T_{k,0} + \frac{1}{L}$$

$$k \text{ to } k\text{: } T_{k,k} = 1 - \sum_{\forall l \text{ such that } 0 \le l \le N \text{ and } l \ne k} T_{k,l} \quad (15)$$

For example, if DEC code is used, up to two faulty bits are correctable by the code. Therefore, $v = 2$ and $1/L$ is added to $T_{1,0}$ and $T_{2,0}$. This means that in addition to the possibilities that a second SEU of size 1×1 or 1×2 exactly hits the already flipped 1 or 2 bits to correct the faulty bit(s), scrubbing can explicitly correct them. Then $T_{k,k}$ should be recalculated so that all the outgoing probabilities from any state in transition matrix **T** sum to 1. After recalculating matrix **T**, the same procedure of (12) to (14) is applied to get f_0 which then is multiplied by the processor clock cycle period to get the intrinsic MTTF.

MACAU models stochastic scrubbing. However it is applicable to deterministic scrubbing as well since it has been shown in [23] that the intrinsinc MTTF with deterministic scrubbing is twice longer than the intrinsic MTTF with stochastic scrubbing with the same average scrubbing interval.

3.4.4 Reliability benchmarking

Using the transition matrix **T**, we calculate $h_{ERR}(j)$ at j^{th} cycle from (1). If a word whose VCC = t is accessed at cycle j, we compute $h_{ERR}(j)$ by matrix power calculations $\mathbf{S}(t) = \mathbf{T}^t$. $S_{u,v}$ is the transition probability from state u to v in t cycles.

$h_{ERR}(j)$, the rate of failure of type ERR, is computed from $\mathbf{S}(t)$. For example, if the 32-bit word is not protected at all, the sum of the transition probabilities from state 0 to 1, 2, …, and 32 gives the $h_{\text{SDC}}(j)$ of that word. Table 1 shows how MACAU computes $h_{ERR}(j)$ when a word is protected by various schemes. Note that TRUE DUE or SDC failures happen only when the word is *consumed* by the processor, as explained in Section 3.1 and in [26].

Matrix power calculations $\mathbf{S}(t) = \mathbf{T}^t$ are the major computation overhead as t can be more than a million cycles at times [26]. The brute-force computation of $\mathbf{T}^t$ requires $O(t)$ matrix multiplications. We use a well-known *square-and-multiply* method [10] to reduce the number of matrix multiplications to $O(\log_2 t)$.

4. Simulations and results

In this section, we first use MACAU to compute the intrinsic MTTFs of various caches. Computed intrinsic MTTFs are compared to results obtained from previous state-of-the art models. Then, we use MACAU to benchmark various L2 caches.

4.1 Simulation setup

The target processor designed for a 65nm technology is a 4-wide out-of-order processor with a 64-entry ROB, 32-entry Load-Store queue, 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. L1-I is a 32KB direct mapped cache with 2

Table 2 Intrinsic MTTF in years with and without scrubbing

SEUs	Protection on a word	Model	32b-word			
			No scrub	Once/year	Once/month	Once/day
SBUs only	SEC	MACAU	6.715E+06	1.092E+13	1.329E+14	3.986E+15
		Saleh	6.245E+06	1.058E+13	1.287E+14	3.862E+15
1BU+2BU (0.5:0.5)	DEC	MACAU	8.012E+06	1.593E+13	1.938E+14	5.813E+15
		Reviriego	7.211E+06	1.411E+13	1.716E+14	5.149E+15
D in (5)	DEC	MACAU	9.700E+06	1.153E+08	1.153E+08	1.153E+08
		Reviriego	8.748E+06	N/A	N/A	N/A
	TEC	MACAU	1.330E+07	1.815E+14	2.209E+15	6.626E+16
		Reviriego	1.700E+07	1.839E+14	2.238E+15	6.713E+16

cycles hit latency. L1-D is a 32KB 4-way set-associative cache with 3 cycles hit latency. The unified L2 is a 2MB 8-way set-associative cache with 20 cycles hit latency. All the caches are non-blocking and write-back. All cache parameters are obtained from Cacti 5 [29]. In these simulations, we do not add the extra latency of SECDED, DECTED or TECQED codes to the L2 cache access latency. Off-chip DRAM latency is computed from DDR2 specification. *Only* L2 is vulnerable and therefore we track ACE cycles only for L2. The benchmarks are 100M SimPoints [28] of randomly selected 20 SPEC CPU2K programs with reference inputs.

4.2 MACAU intrinsic MTTF results

In this section, we show the intrinsic MTTF calculated by MACAU using the procedure of (12) to (14) (and (15) if scrubbing is used). In order to verify that MACAU calculates the intrinsic MTTF correctly, we compare MACAU's results with Saleh's [23] and Reviriego's [21] models.

In [23], Saleh et al. model the intrinsic MTTF of a word-level SEC-protected cache in the presence of SBUs for SEC-protected caches without and with scrubbing by closed-form equations (16) and (17):

Without scrubbing:

$$\mathrm{MTTF} = \frac{1}{\lambda_{\mathrm{word}}} \times \sqrt{\frac{\pi}{2 \times M}} \tag{16}$$

With stochastic scrubbing at avg interval L:

$$\mathrm{MTTF} = \frac{1}{M \times L \times \lambda_{\mathrm{word}}{}^{2}} \tag{17}$$

where λ_{word}, M and L are the SEU rate of a word (λ_{word} is equivalent to our $p_{\mathrm{SEU,PD}}$), the number of words in the cache and the scrubbing interval respectively. In [21], Reviriego et al. find that the intrinsic MTTF of a word-level DEC-protected cache under up to 2BUs is obtained by replacing λ_{word} by λ'_{word} in (16) and (17).

$$\lambda'_{\mathrm{word}} = \lambda_{\mathrm{word}} \times \sqrt{p2 \times (2 - p2)} \tag{18}$$

where $p2$ is the fraction of 2BUs among all SEUs.

Table 2 shows the intrinsic MTTFs calculated with MACAU and with Saleh's and Reviriego's models. As can be seen in the table, the intrinsic MTTFs (without scrubbing) computed by MACAU broadly agree with Saleh's (case of SBUs only) or Reviriego's (2BUs) models. The advantage of MACAU is it is a universal model for the complex cases of mixes of SBUs and spatial MBUs with multiple dimensions.

Results for scrubbing are also shown in the table for three scrubbing intervals of once per year, once per month and once per day. MACAU computes intrinsic MTTFs that agree with Saleh's or Reviriego's model for SBUs and SBUs plus 2BUs respectively.

One interesting observation is that Reviriego's model can also compute the intrinsic MTTF of DEC-protected caches (although their model only applies to caches with codes correcting *m* errors in the presence of SEUs flipping up to *m* bits) when SEUs include the complex mixture of SBUs and MBUs defined by matrix **D**. However, Reviriego's model cannot correctly compute the intrinsic MTTFs of the DEC-protected cache when up to 3BUs are existent (see shaded cells in Table 2). Because of 3BUs, caches protected by DEC codes fail more frequently. Reviriego's simple model fails to quantify them since this situation goes beyond the capability of the model by breaking the basic assumption that the protection codes used are powerful enough to correct all the faults due to a single SEU. MACAU and Reviriego are again in broad agreement for MBUs given by matrix **D** under TEC code.

4.3 MACAU benchmarking results

In this section, we show benchmarking results for 2MB caches protected by various schemes (no-protection, odd-parity, SECDED, DECTED and TECQED) obtained with MACAU. Table 3 summarizes the benchmarking results for the 2MB cache that is

Table 3 Average benchmarking results when SBUs and spatial MBUs coexist (FITs)

		No protection	Odd-parity	SECDED	DECTED	TECQED
DUE	TRUE	--	1217.840	110.872	37.614	7.3947E-16
	FALSE	--	2448.644	222.923	75.629	1.3724E-15
SDC		1328.711	110.872	37.614	8.0925E-16	6.9784E-17

unprotected or protected by odd-parity, SECDED, DECTED or TECQED code when SBUs and spatial MBUs coexist as in Figure 2. Averages of the benchmarking results were obtained using (3). Modified sim-outorder [6] was used for benchmarking.

We compared MACAU's benchmarking results with PARMA's results for cases with SBUs only, since PARMA can only model SBUs and temporal MBUs. Among 20 results, six have at most a 0.015% difference in FIT rates for the 2MB SECDED protected caches. FIT rate results for other benchmarks match down to five digits below the decimal point. We believe the differences are due to floating-point rounding errors. We used IEEE double precision representation for floating-point numbers in the simulations.

In general, when 3BUs exist it is commonly believed that the TECQED code or some equivalently strong protection code should be used to mask failures. However, the results in Table 3 suggest that, although there are 3BUs among the spatial MBUs, DECTED code can suppress DUEs and SDCs efficiently. If strong protection schemes like TECQED do not meet area or performance budgets, lowering the protection strength to DECTED may be acceptable given the typical reliability budget of today's chips as the one in [4]. With the MACAU framework, designers can benchmark caches to choose the best protection scheme among many design choices when SBUs and spatial MBUs coexist.

5. Extending MACAU to several SEUs

The current version of MACAU assumes no more than two SEUs on the same word between two accesses to it. One important assumption in MACAU is that one SEU flips bits in a contiguous $1 \times k$ rectangular pattern. Thus after one SEU to a word, the fault pattern is contiguous. However, after two SEUs the resulting fault pattern in the word may be disconnected because of overlap. Thus the state of the fault cannot simply be represented by the number of faulty bits, a basic premise of MACAU's Markov chain model. However, we can easily extend MACAU to more than two SEUs by making the approximation that, if the word has k faulty bits when a new SEU hits, the fault pattern is always $1 \times k$, regardless of the number of SEUs that led to this pattern.

We extended the MACAU model to deal with more than two SEUs by allowing errors that are practically unobservable. The FIT rate results we obtained by using the matrix **T** in (10) and the complex matrix **T** built after this extension match down to 10 decimal points.

6. Conclusion

In this paper, we demonstrate a new soft-error benchmarking framework called MACAU. MACAU is a Markov chain model for the soft failures of memory structures when an SEU can be an SBU or a spatial MBU. Given current experimental data on the shape and probabilities of spatial MBUs, MACAU assumes that spatial MBU patterns are compact and that at most two SEUs can affect a word. However with some approximations (all overestimations of FIT) , MACAU can cover situations with several spatial MBUs.

MACAU can calculate the intrinsic MTTF of caches with and without scrubbing and it can also realistically benchmark the soft-error reliability of caches for specific programs. Therefore, MACAU can be used by circuit designers and computer architects alike to quantify and qualify the reliability of caches during the design process. To the best of our knowledge, MACAU is the *only* framework that addresses the effect of SBUs and spatial MBUs together to calculate intrinsic MTTFs and benchmark various cache designs.

Currently MACAU is developed for word-level protection schemes. Several recent studies [15][26] suggest that increasing the size of the protection domain is preferable because SEU rates are currently very low in realistic environments. Modeling TAG vulnerability is another challenging topic. How to deal with cache TAGs, PDs larger than a word or edge effects will be part of our future work.

7. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. CNS-0834798, CNS-0834799 and CCF-0954211.

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 Conference on Design, Automation and Test in Europe*. 1276-1281, Mar 2006.

[2] S. Baeg, S. Wen, R. Wong, "SRAM Interleaving Distance Selection With a Soft Error Failure Model," In *IEEE Transactions on Nuclear Science*, 56(4), 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] Bossen, D. C. "CMOS Soft Errors and Server Design. IEEE 2002 Reliability Physics Tutorial Notes," Reliability Fundamentals 121 (2002), 07-1.

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

[6] D. Burger and T. M. Austin. The SimpleScalar Tool Set Version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin--Madison, May 1997.

[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*, 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] G. Georgakos, P. Huber, M. Ostermayr, E. Amirante, F. Ruckerbauer, "Investigation of Increased Multi-Bit Failure Rate Due to Neutron Induced SEU in Advanced Embedded SRAMs," 2007 *IEEE Symposium on VLSI Circuits*. 2007-01-01;80-81.

[10] D. M. Gordon, "A Survey of Fast Exponentiation Methods," *Journal of Algorithms*, 129-146, 1998.

[11] C. H. Grinstead, J. L. Snell, "Introduction to Probability, 2nd Edition," American Mathematical Society, 452-461

[12] 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, 437-444, Sep. 2006

[13] 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.

[14] Mahatme, N.; Bhuva, B.; Fang, Y.; Oates, A.; , "Analysis of multiple cell upsets due to neutrons in SRAMs for a Deep-N-well process," *Reliability Physics Symposium (IRPS), 2011 IEEE International*. SE.7.1-6, Apr 2011

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

[16] Meyer C.D. Jr. (1978). "An alternative expression for the mean first passage time matrix," *Linear Algebra Appl.*, 22, 41-47.

[17] Zhu Ming; Xiao Li Yi; Liu Chang; Zhang Jian Wei; , "Reliability of Memories Protected by Multibit Error Correction Codes Against MBUs," In *IEEE Transactions on Nuclear Science*, 58 (1), 289-295, Feb. 2011

[18] 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.

[19] 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.

[20] OpenMP, https://computing.llnl.gov/tutorials/openMP/

[21] Reviriego, P.; Maestro, J.A., "Study of the Effects of Multibit Error Correction Codes on the Reliability of Memories in the Presence of MBUs," In *IEEE Transactions on Device and Materials Reliability*, 9(1), 31-39, Mar 2009

[22] M. Sahinoglu. "Compound-Poisson Software Reliability Model," *IEEE Trans. Softw. Eng*. 18, 624-630, Jul 1992

[23] 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.

[24] Seifert, N.; Gill, B.; Foley, K.; Relangi, P.; "Multi-cell upset probabilities of 45nm high-k + metal gate SRAM devices in terrestrial and space environments," In *Proceedings of the IEEE International Reliability Physics Symposium,*181-186, 2008

[25] Semiconductor Industries Association. International Technology Roadmap for Semiconductors. 2007.

[26] J. Suh, M. Manoochehri, M. Annavaram, M. Dubois. "Soft error benchmarking of L2 caches with PARMA," *In Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems* (SIGMETRICS '11).

[27] Tipton, A.D.; Pellish, J.A.; Hutson, J.M.; Baumann, R.; Deng, X.; Marshall, A.; Xapsos, M.A.; Kim, H.S.; Friendlich, M.R.; Campola, M.J.; Seidleck, C.M.; LaBel, K.A.; Mendenhall, M.H.; Reed, R.A.; Schrimpf, R.D.; Weller, R.A.; Black, J.D.; , "Device-Orientation Effects on Multiple-Bit Upset in 65 nm SRAMs," In *IEEE Transactions on Nuclear Science*, 55(6), 2880-2885, Dec. 2008

[28] T. Sherwood, E. Perelman, G. Hamerly and B. Calder. "Automatically Characterizing Large Scale Program Behavior," In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 45-57 Oct 2002.

[29] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. Cacti 5.1. Technical Report HPL-2008-20, Hewlett-Packard Development Company, Apr 2008.