Calculating Architectural Vulnerability Factors for Spatial Multi-Bit Transient Faults*

Mark Wilkening, Vilas Sridharan†, Si Li‡, Fritz Previlon, Sudhanva Gurumurthi§ and David R. Kaeli
ECE Department, Northeastern University, Boston, MA, USA,
wilkening.m@husky.neu.edu, previlon@ece.neu.edu, kaeli@ece.neu.edu
†*RAS Architecture, Advanced Micro Devices, Inc., Boxborough, MA, USA, vilas.sridharan@amd.com*
‡*ECE Department, Georgia Institute of Technology, Atlanta, GA, USA, sli@gatech.edu*
§*AMD Research, Advanced Micro Devices, Inc., Boxborough, MA, USA, sudhanva.gurumurthi@amd.com*

***Abstract*—Reliability is an important design constraint in modern microprocessors, and one of the fundamental reliability challenges is combating the effects of transient faults. This requires extensive analysis, including significant fault modeling to allow architects to make informed reliability tradeoffs. Recent data shows that multi-bit transient faults are becoming more common, increasing from 0.5% of SRAM faults in 180nm to 3.9% in 22nm, and are predicted to be even more prevalent in smaller technology nodes. Therefore, accurately modeling the effects of multi-bit transient faults is increasingly important to the microprocessor design process.**

Architecture vulnerability factor (AVF) analysis is a method to model the effects of single-bit transient faults. In this paper, we propose a method to calculate AVFs for spatial multi-bit transient faults (MB-AVFs) and provide insights that can help reduce the impact of these faults. First, we describe a novel multi-bit AVF analysis approach for detected, uncorrected errors (DUEs) and show how to measure DUE MB-AVFs in a performance simulator. We then extend our approach to measure SDC MB-AVFs. We find that MB-AVFs are not derivable from single-bit AVFs. We also find that larger fault modes have higher MB-AVFs. Finally, we present a case study on using MB-AVF analysis to optimize processor design, yielding SDC reductions of 86% in a GPU vector register file.

***Keywords*-reliability; soft errors; fault tolerance;**

I. Introduction

Reliability is a critical design constraint for all microprocessors. In particular, studies have shown particle-induced transient faults in SRAM to be the dominant contributor to microprocessor faults [5]. These faults arise from high-energy particle strikes –such as neutrons from cosmic radiation– which deposit charge onto transistors as they pass through a silicon device. A sufficient amount of charge has the potential to invert the state of a logic device and generate a temporary fault. There have been several documented cases of transient faults occurring in production systems on both CPUs and GPUs [16] [27] [30] [34].

In order to prevent excessive failures from transient faults, microprocessor vendors set a desired failure rate target for each design. Vendors then design for the best power and performance under this constraint. Significant analysis is required to validate the design against this target, and improve the efficiency of the protection used to achieve this failure rate. Common techniques include accelerated testing of new technologies [17], pre-silicon validation and efficiency analyses such as statistical fault injection and architectural vulnerability factor modeling [11] [25], and post-silicon validation such as field-data collection and beam testing [12] [34]. By using all of these techniques together, a vendor can design microprocessors to meet stringent reliability standards.

Recently, multiple-bit transient faults have emerged as an increasingly important challenge in SRAM [17]. Due to technology scaling, high-energy particles are able to deposit charge onto multiple SRAM devices, resulting in faults in multiple adjacent bits. This is called a *spatial* multi-bit fault. Table I (reproduced from Ibe et al. [17]) shows that in a 180nm process, less than 0.6% of faults affected more than one bit along an SRAM wordline, while in a 22nm process, 3.6% of all faults affected multiple bits along a wordline. This increase in multi-bit fault rate is projected to continue despite the introduction of technologies such as FinFET transistors, which reduces the overall rate of transient faults, but does not slow the trend toward larger multi-bit faults [29].

An architect can choose to employ many techniques to protect against multi-bit faults. For instance, stronger error-correcting codes (ECCs) such as double-error-correction, triple-error-detection (DEC-TED) ECCs can be used, and *bit interleaving*, which ensures that physically adjacent bits are protected by different ECCs, can be used in conjunction with ECCs [24]. However, these techniques have power, area, and performance costs. For instance, implementing DEC-TED ECC on a 128 bit data word requires 17 check bits - a 13% overhead - whereas implementing single-error-correction, double-error-detection (SEC-DED) ECC requires only 7% overhead (9 check bits). DEC-TED ECC also requires a deeper XOR tree to decode, resulting in increased latency. Similarly, a physically-interleaved array requires

*A portion of this work was performed while Mark Wilkening and Si Li were co-ops in AMD Research. David R. Kaeli, Fritz Previlon, and Mark Wilkening were supported in part by NSF CISE grant SHF-1017439.

Design Rule (nm)	Total	Bit width of multi-bit fault			
		2	3	4-8	>8
180	0.5	0.5	0.0	0.0	<0.1
130	1.2	0.8	0.2	0.2	<0.1
90	1.9	1.5	0.2	0.2	<0.1
65	2.4	2.1	0.1	0.0	<0.1
45	2.3	1.9	0.2	0.1	<0.1
32	3.1	2.6	0.2	0.3	<0.1
22	3.9	3.0	0.2	0.3	0.1

Table I: Percent ratio of multi-bit faults to total faults from Ibe et al. [17]. Multi-bit faults are 3.9% of all faults in 22nm and both rate and width are increasing with decreased feature sizes.

multiple columns to be activated and multiplexed in order to read out a single bit, resulting in increased dynamic power consumption, area overhead, and latency [4]. Therefore, an architect must understand the required level of multi-bit protection: excessive protection will unnecessarily increase power and area, and reduce performance, but inadequate protection will result in an unreliable design.

Single bit transient faults have been a known challenge for some time, and there are well developed methods for characterizing and determining the impact of such faults. One method is to measure architectural vulnerability factors (AVFs) through architecturally-correct execution (ACE) analysis [25]. Measuring AVFs allows an architect to prioritize hardware structures for protection and, when combined with transient fault rates from accelerated testing, to bound the overall soft error rate of a chip. Unfortunately, while multi-bit fault rates can be measured via accelerated testing, methods to allow architects to quantify the impact of multi-bit faults are lacking.

In this paper, we propose a method to quantify architectural vulnerability factors for spatial multi-bit faults (MB-AVFs). To measure MB-AVFs, we develop a novel extension to architecturally-correct execution (ACE) analysis in a performance simulator. We measure MB-AVFs for detected, unrecoverable errors (DUE MB-AVF) as well as those for silent data corruptions (SDC MB-AVF). MB-AVF analysis allows an architect to understand the impact of architectural decisions on a design's failure rate due to multi-bit faults, and MB-AVF values derived from ACE analysis can be used in conjunction with fault rates from accelerated testing to calculate the soft error rate of a chip from all transient faults.

The novel contributions of this paper are as follows:

- We introduce a method to quantify and calculate the MB-AVF of any hardware structure. This method applies to both DUE MB-AVF and SDC MB-AVF.
- We present a precise model to measure DUE MB-AVFs using ACE analysis in performance simulation.
- We extend this model to approximate SDC MB-AVFs using ACE analysis, and use a fault injection study to determine that this approximation has negligible error.
- We show how the inherently-parallel nature of a GPU can be exploited to reduce the SDC MB-AVF of the vector general-purpose register file (VGPR), while minimizing area cost dedicated to reliability.

Our work has several key findings that are applicable to the design of any processor:

- We show that MB-AVFs are not derivable analytically from single-bit AVFs; MB-AVFs can vary independently of single-bit AVFs and must be measured through simulation.
- We find MB-AVFs that range from one to M times single-bit AVFs, where M is the number of bits in the multi-bit fault pattern in question, and that larger fault modes have larger MB-AVFs, offsetting to some extent the reduced rate of these larger faults (shown in Table I).
- We find that many multi-bit faults that designers conservatively assume cause SDCs actually cause DUEs.
- We identify a property of hardware structures called *ACE locality*, or the tendency of ACE bits to cluster together, that can be used to guide design. Structures that exhibit high ACE locality have lower MB-AVFs. Therefore, for example, *logical* interleaving, where each data word is split into multiple interleaved check words, can have MB-AVFs many times lower than that of *physical* interleaving, where each data word is interleaved with other data words.

The rest of the paper is organized as follows. Section II presents background and terminology related to transient faults, AVF, and ACE analysis. Section III provides an overview of related work. Section IV presents our model for multi-bit DUE AVF. Section V explains our method for measuring multi-bit AVF using ACE analysis. This is followed by a description of the insights obtained in section VI. Then, section VII extends this model to multi-bit SDC AVF. Finally, we present a case study for using the information obtained through our model for practical design decisions in section VIII, and conclude in section IX.

II. Background and Terminology

A. Definitions and Concepts

We define a *fault* as an undesired state change in hardware. In this work we limit ourselves to the analysis of *transient* faults, or faults which cause a temporary state change (versus stuck-at, or *permanent*, faults). If a transient fault flips a bit in a hardware structure, that bit can be overwritten to remove the fault. Faults affecting multiple (2 or more) bits in a hardware structure are called *multi-bit faults*. Any fault (single or multi-bit) results in all affected bits flipping to the opposite state.

There are three major classes of transient fault that can occur due to particle strikes. A *single-bit fault* (SBF) occurs given a single strike affecting a single bit. A *spatial multi-bit fault* (sMBF) occurs given a single strike affecting multiple bits at the same instance in time. This is in contrast to a *temporal multi-bit fault* (tMBF), in which multiple strikes

(spaced in time) lead to multiple flipped bits the next time a set of bits is read. Both spatial and temporal multi-bit faults can be arbitrary shapes and sizes.

We define an *error* as an incorrect result in program output. Errors can also be further classified based on their impact on the system. An undetected error can cause *silent data corruption* (SDC), which causes a program to behave incorrectly without being detected by the hardware. Errors which are detected but are not corrected are called *detected uncorrected errors* (DUE). Error detection is not ideal. Some errors are detected early, and will not cause data corruption if ignored. These errors are named *false DUEs*, while detected errors which would result in SDC if ignored are called *true DUEs* [42].

B. Architectural Vulnerability Factors

Not all faults become errors. The fraction of faults in a hardware structure that become errors is called the structure's *architectural vulnerability factor*, or AVF [25]. The AVF of a hardware structure H containing B_H bits over a period of N cycles can be expressed as follows:

$$AVF_H = \frac{\sum_{n=0}^{N} [\text{ACE bits in H at cycle n}]}{B_H \times N} \qquad (1)$$

AVFs are measured by identifying whether state in a system is required for architecturally correct execution. State in the system required for architecturally correct execution is named *ACE*, and any fault in this state will result in incorrect execution (an error). All other state is *unACE* state, and faults in this state will have no effect on correct execution. The process of identifying ACE and unACE state is called *ACE Analysis*. ACE analysis conservatively assumes that all state is ACE and then systematically proves state unACE, resulting in the computation of an upper bound estimate for the AVF of the system. While ACE analysis has certain limitations (e.g., [23]), the technique is widely used and has immense practical value to industry [9] [18].

C. Existing Multi-bit Remediation Techniques

Existing processors use a variety of techniques to protect against multi-bit transient faults. Techniques such as redundant multi-threading can detect most multi-bit faults (e.g., [19] [43]). Many error correcting codes (e.g., DEC-TED ECCs) can detect and correct multi-bit faults, and cyclic redundancy codes (CRCs) can provide robust multi-bit error detection [22].

A technique known as *bit interleaving* is commonly used in conjunction with parity or ECC [24]. *Logical interleaving* increases the number of ECC words per data word, and assigns physically-adjacent bits to different ECC words. This increases the area overhead of ECC since each data word contains multiple ECCs. *Physical interleaving*, on the other hand, assigns physically-adjacent bits to different data words, thus ensuring adjacent bits are protected by different ECCs. Interleaving can be done between 2 data words (*x2 interleaving*), 4 data words (*x4 interleaving*), and so on.

III. Related Work

There has been a large body of research on AVF modeling and analysis, starting with the works by Mukherjee et al., Weaver et al., and Biswas et al., who coined the term and developed the initial methodology [7] [25] [42]. Adve et al. demonstrated limits to AVF analysis, showing that assumptions of independence break down at very high fault rates or in very large structures [23]. Wang et al. compare ACE analysis to fault injection and find ACE analysis to be conservative [41]. This analysis was rebutted, and the effects pointed out by Wang were incorporated into an ACE analysis infrastructure [8]. Sridharan and Kaeli extended the notion of vulnerability to allow independent measurement of the vulnerability of programs and hardware [32] [33]. There have also been several publications that estimate AVF using architectural or microarchitectural state [26] [40] and that measure the AVF of GPUs [13] [38].

A few prior studies explore the impact of multi-bit faults in SRAMs and logic. George et al. modeled spatial MBFs using a fault injection methodology [15]. George et al. examined interleaving as a potential remedy to sMBFs in combinational logic [14], and Szafaryn et al. examined the overheads associated with multi-bit fault protection [37]. Suh et al. introduced the PARMA framework to compute the MTTF of caches from single-bit and temporal multi-bit faults [36]. Suh et al. then proposed the MACAU framework to compute the intrinsic MTTF of hardware structures under the cumulative effects of overlapping SBFs, temporal MBFs and spatial MBFs using Markov chains [35].

Our work differs from the work by George et al. in much the same way that the original ACE analysis methodology differed from prior fault injection frameworks [21]. Our work also differs from MACAU in several ways. First, MACAU computes MTTFs and not AVFs. Therefore, MACAU is useful for estimating the soft error rate of a final product, including all architecture and technology effects, but it is difficult to separate the effects of single-bit and multi-bit faults, and to isolate architectural from technology-driven effects. MB-AVF analysis is specifically designed to allow this type of architectural analysis, irrespective of process technology choices. Second, although MACAU supports a wide variety of error correcting codes, it cannot evaluate important design points such as bit interleaving: every multi-bit fault in a structure with interleaving affects multiple ECC words, a case which MACAU ignores in order to simplify the needed math. Finally, MACAU does not integrate with existing industrial AVF infrastructures that support sophisticated masking analysis (e.g., [9] [18]).

IV. MB-AVF for Detected Uncorrected Errors

In this section, we provide an overview of our modeling strategy to compute AVFs for multi-bit faults. We begin by

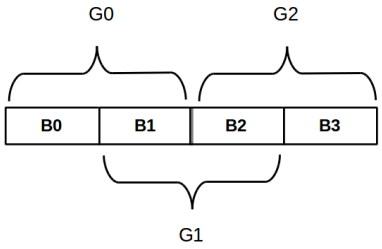


Figure 1: A fault mode is a specific multi-bit fault pattern, and a fault group is a set of bits in a structure that match this pattern. For example, a 2x1 multi-bit fault mode has 3 unique fault groups (G0 through G2) in this 4x1 SRAM array (B0 through B3).

introducing some basic concepts and assumptions present in our modeling approach, and then show through an example how to compute multi-bit AVF from single-bit AVF.

A. Fault Modes and Fault Groups

The term *multi-bit fault* can refer to the failure of any number of bits in any arbitrary geometry. In this work we define the term *fault mode* to refer to a particular multi-bit fault geometry (i.e., a specific pattern of bit flips). For example, we refer to a particle strike affecting three consecutive bits within one row as a *three-by-one* spatial multi-bit fault, where three-by-one is the fault mode. Different fault modes will appear with different raw fault rates, and can be analyzed separately using different vulnerability factors to determine separate error rates for each mode.

In this work, we will refer to an individual multi-bit fault of a particular fault mode as occurring on a particular *fault group* in a particular cycle. We define a fault group as the set of bits satisfying the geometry of a particular fault mode, on which a multi-bit fault of that mode may occur. For example, Figure 1 presents all possible fault groups of a 2x1 fault mode on a 4x1 bit structure.

B. Spatial vs. Temporal Multi-bit Faults

In this work we constrain ourselves to modeling only spatial multi-bit faults. To justify our focus on sMBFs, we must confirm that sMBFs are a greater threat than tMBFs.

Mean-times-to-failure (MTTFs) for tMBFs can be calculated using the methodology developed by Saleh et al. [28]. Recent data shows that for sMBFs in 22nm technology, 0.1% of all neutron strikes affect more than 8 bits along a wordline [17]. Figure 2 shows that at this sMBF rate, and across a variety of realistic raw fault rates [31], the MTTF of a 32MB cache from sMBFs is lower than the MTTF from tMBFs even when assuming infinite cache lifetimes (i.e. data lasts forever and is never replaced). When limiting cache line lifetime to 100 years, the MTTFs from tMBFs increase by several orders of magnitude.

Furthermore, as cell spacing decreases in future technologies, the percentage of particle strikes that cause sMBFs is likely to increase. Figure 2 shows that a 5% rate of sMBFs

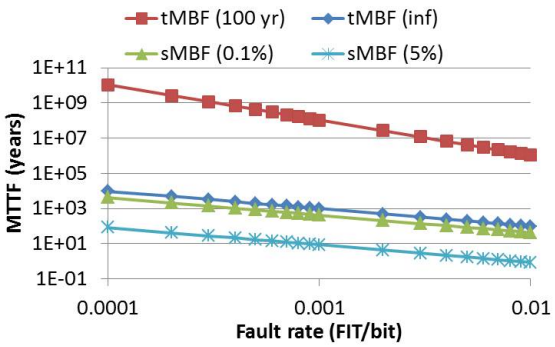


Figure 2: Mean-time-to-failure (MTTF) of a 32MB cache from tMBFs and sMBFs at various raw fault rates. Realistic rates of sMBFs result in MTTFs that are 6-8 orders of magnitude lower than MTTFs from tMBFs, even when assuming an average cache lifetime of 100 years.

will decrease the MTTF by two orders of magnitude relative to a 0.1% rate, further exacerbating the problem of sMBFs.

From this analysis, we conclude that sMBFs are significantly more of a threat to silicon reliability than tMBFs, and that modeling and remediation efforts should focus on sMBFs rather than tMBFs. For the rest of this paper, the term multi-bit fault refers solely to a spatial multi-bit fault, and multi-bit AVF refers to AVF of spatial multi-bit faults.

C. Calculating Spatial Multi-bit AVF

We define the multi-bit AVF (MB-AVF) of a hardware structure as the probability that a fault of a given multi-bit fault mode in the structure will result in a visible error in the final output of a program. MB-AVF is determined through ACE analysis, where *fault groups* required for correct execution are considered ACE groups, and fault groups that are not required for correct execution are considered unACE. The MB-AVF of a hardware structure is the fraction of *fault groups* in the structure that are ACE. MB-AVF is computed similarly to single-bit AVF, but considers fault groups in place of individual bits.

MB-AVF values are determined independently for each fault mode. This is consistent with how multi-bit fault rates are measured; a typical accelerated testing campaign measures an overall fault rate, as well as a fault rate per fault mode [17]. For a hardware structure H containing $G_{H,M}$ unique fault groups of mode M, the structure's MB-AVF for fault mode M over a period of N cycles can be expressed as follows:

$$MB\text{-}AVF_{H,M} = \frac{\sum_{n=0}^{N} [\text{ACE groups in H at cycle n}]}{G_{H,M} \times N} \quad (2)$$

D. Differences Between Single- and Multi-bit AVF

Now that we have defined MB-AVF, it is worth asking how much MB-AVF can differ from traditional single-bit AVF (SB-AVF), and whether it is possible to derive MB-AVF analytically from SB-AVF. Let us consider a single

fault group of a fault mode with M bits. If all bits in this group are ACE, the SB-AVF and MB-AVF of this fault group in a single cycle are:

$$SB\text{-}AVF = \frac{M \times 1}{1 \times M} = 100\%$$

$$MB\text{-}AVF_M = \frac{1}{1} = 100\% = 1 \times SB\text{-}AVF$$

However, if only one bit is ACE, then the single-bit and multi-bit AVFs of this group in a single cycle are:

$$SB\text{-}AVF = \frac{1 \times 1}{1 \times M} = \frac{100}{M}\%$$

$$MB\text{-}AVF_M = \frac{1}{1} = 100\% = M \times SB\text{-}AVF$$

Therefore, from first principles, we can see that the MB-AVF can vary between 1x and Mx SB-AVF, where M is the number of bits in the fault mode. Moreover, this multiplier is determined by the pattern of ACE times among bits in the fault group. If all bits are ACE in exactly the same cycles, then the multi-bit AVF equals the single-bit AVF. On the other hand, if only one bit is ACE in each cycle, then the multi-bit AVF is Mx the single-bit AVF.

The pattern of adjacent ACE and unACE times depends on the interaction between the physical layout of a hardware structure and workload behavior and thus is not easily derivable analytically. Furthermore, the fact that MB-AVF can be Mx single-bit AVF implies that using single-bit AVF to approximate multi-bit AVF may significantly underestimate a design's SER from multi-bit faults.

E. Calculating Soft Error Rates

Given raw fault rates for each fault mode (e.g., from accelerated testing) measured in failures per billion hours (FIT), and MB-AVFs for a hardware structure, we can multiply the raw fault rate and MB-AVF for each fault mode to get a soft error rate from that particular fault mode for that structure. If we sum these over all fault modes (including single-bit faults) we can derive a total SER for that structure. For a hardware structure H considering M fault modes with G_M fault groups in the structure, over a period of N cycles, the soft error rate of the structure is given by the following equation:

$$SER_H = \sum_{m=0}^{M} \sum_{g=0}^{G_M} \sum_{n=0}^{N} FIT_m \times MB\text{-}AVF_{m,g,n} \qquad (3)$$

By summing SER_H over all structures we can calculate the overall soft error rate of a chip from all single- and multi-bit transient faults.

V. Measuring Multi-bit AVF

This section describes a method to compute DUE MB-AVFs using a performance simulator. One method to compute MB-AVFs would be to perform a fault injection campaign for each fault mode. However, fault injection suffers from well-known problems that spurred development of ACE analysis, a method to identify bits necessary for Architecturally Correct Execution (ACE bits) using a performance simulator [25]. The problems with fault injection are exacerbated by the need to quantify MB-AVFs for multiple fault modes. Therefore, we focus on measuring MB-AVFs by extending a typical performance simulator-based ACE analysis infrastructure.

In this section, we focus on calculating DUE MB-AVF. Section VII extends this model to estimate SDC MB-AVF.

A. ACE Groups and Protection Domains

For this model, a fault group is considered ACE if any of the bits within the group are considered ACE. A group with no ACE bits is an unACE group. If a flip in any bit in the group will cause a DUE, then flipping all bits will also cause a DUE. Therefore, the ACEness of group G containing B bits at cycle C can be expressed by the following equation:

$$ACE_{G,C} = \bigcup_{b=0}^{B} ACE_{b,C} \qquad (4)$$

This model of multi-bit ACE analysis is simple and can be computed easily from single-bit ACE analysis.

Another extremely important aspect of characterizing the impact of multi-bit faults is the effect of *error protection schemes* such as parity or error-correcting codes (ECCs). In order to accurately model an error protection scheme, a framework must be constructed to define the scheme. We define error protection schemes in our model through the definition of *protection domains*, and their interactions with multi-bit faults. A protection domain consists of a region of data protected by a single element of the protection scheme; this corresponds to a single parity or ECC word. Each protection scheme can define the size of its protection domains, as well as the location(s) of the domains. This allows for a flexible incorporation of different ECCs and interleaving schemes. A protection scheme must also define the number of faults present in a single domain that the scheme can detect or correct. This defines the level of protection of the scheme and how it will interact with multi-bit faults to create different types of errors.

B. Multi-bit ACE Analysis

In order to calculate MB-AVF through ACE analysis, the ACEness of fault groups must be determined. With the addition of protection domains, Equation 4 is no longer valid because individual bits in a group are no longer independent due to the interaction of multi-bit faults and

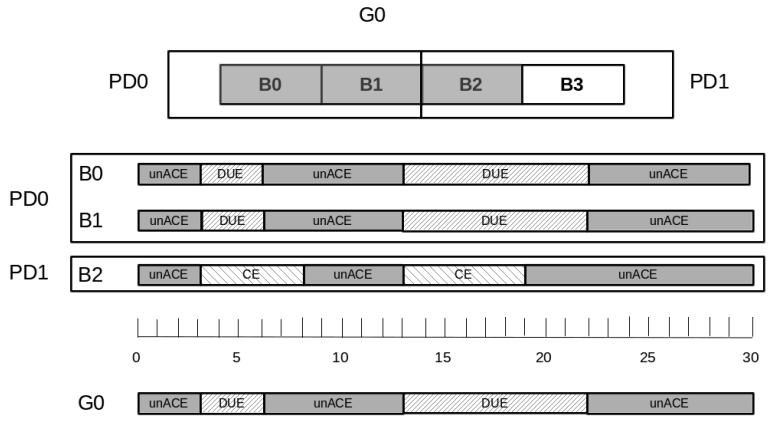


Figure 3: MB-AVF ACE Analysis of a 3x1 Fault over two SEC-DED protection domains. A fault that affects B0-B2 is DUE ACE in PD0 and corrected (unACE) in PD1. If a fault is DUE ACE in either PD0 or PD1, fault group G0 is DUE ACE.

protection domains. When considering protection schemes, a fault group can contain bits from more than one protection domain. We refer to a set of bits from a single protection domain within a fault group as an *overlapped region*.

Two measurements must be considered when determining the effect of a protection domain overlapped with a fault group: the ACEness of the overlapped region and the action taken by the protection domain upon observing the fault.

For generality, the ACEness of an overlapped region R is the union of the ACEness of all bits B in the region, as shown in equation 5:

$$ACE_{R,C} = \bigcup_{b=0}^{B} ACE_{b,C} \tag{5}$$

In practice, a protection domain is read as a complete unit, and the DUE ACEness of all bits in an overlapped region will always be the same.

The action taken by the protection domain determines if the fault is corrected, detected, or undetected upon being observed. This action is a direct specification of the protection scheme used. For example, with a simple model of SEC-DED ECC, an overlapped region of size 1 will result in a corrected error. If the size of the overlapped region is exactly 2, the error is detected but uncorrected.

An overlapped region R is DUE-ACE in cycle C if the region is ACE in cycle C and the region reacts as detected with the protection domain. This calculation is expressed in equation 6:

$$DUEACE_{R,C} = ACE_{R,C} \wedge Detected_R \tag{6}$$

The DUE ACEness of a fault group is then calculated as the union of the DUE ACEness of all overlapped regions in the fault group. The ACEness of group G containing R overlapped regions at some cycle C can be expressed by the following equation:

$$DUEACE_{G,C} = \bigcup_{r=0}^{R} DUEACE_{r,C} \tag{7}$$

Take the group shown in Figure 3 for example. This figure depicts a 3x1 fault group (G0) overlapping two SECDED ECC protection domains (PD0, PD1) over a period of 30 cycles. To calculate the DUE MB-AVF of these bits, ACE analysis must be used to calculate the ACEness of this group over this 30 cycle period. We first assume we have already performed single-bit ACE analysis to determine ACE and unACE cycles for each individual bit. We then must analyze each overlapped region between the fault group and any protection domains separately.

Consider the overlapped region from PD0, which is a SECDED ECC protection domain containing two bits from the fault group. If a 3x1 multi-bit fault occurs, this protection domain will detect an error upon reading the data. Because of this we can classify the ACE time in bits B0 and B1 as DUE ACE. Considering the overlapped region from PD1, which contains one bit from the fault group, we can determine the protection domain will correct all errors upon reading the data. We therefore classify all ACE time as corrected. Because the DUE ACEness of a fault group is the union of the DUE ACEness of its overlapped regions, cycles with any DUE ACE regions are considered DUE ACE for the group.

C. Computational Complexity

An important aspect in considering the usefulness and practicality of this model is the computational complexity. The model is similar in computational complexity to single-bit ACE analysis. For multi-bit analysis, bits in the design must be organized into overlapped regions, and as events in the bits in these regions are resolved, analysis accounting for the interactions of the bits in the sets may resolve the ACEness of the sets. Overlapped regions and their reactions can be calculated statically. In this manner, multiple fault

modes of arbitrary complexity may be statically set up before simulation, and dynamically evaluated as events are resolved through single-bit ACE analysis.

VI. Insights From Multi-bit AVF

In this section, we present results from measuring multi-bit AVFs in our processor model. For ease of description, most results in this section use parity protection. Similar trends were observed when using ECC protection unless otherwise noted. We usually report MB-AVFs normalized to single-bit AVF (SB-AVF) in order to highlight trends across benchmarks with vastly different absolute AVF values.

A. Experimental Setup

Graphic processing units (GPUs) and accelerated processing units (APUs) are now used in scientific and mission-critical applications. Therefore, the reliability of these devices is paramount. We use the gem5 simulator to model an accelerated processing unit (APU) consisting of an x86 CPU and integrated GPU with 4 compute units and 1 memory channel [6]. We augment the simulator with an AVF measurement infrastructure similar to those described in the literature [25] [33]. We measure AVF in the GPU L1 and L2 caches. Each compute unit has a 16KB L1 cache and the integrated GPU has a 256KB L2 cache per memory channel. Both caches use a 64-byte cache line but allow byte reads and writes. These structure sizes are consistent with state-of-the-art designs, and a high-end GPU may have dozens of compute units and up to a dozen memory channels [1].

The workloads for our experiments come from the Rodinia benchmark suite, the AMD OpenCL sample suite, and the Mantevo benchmark suite [2] [3] [10]. All workloads are evaluated to completion.

The basic AVF measurement is conducted in two phases: an event-tracking phase followed by an analysis phase. The event-tracking phase collects the time when key events that can potentially affect the ACEness of the bits in the structure occur. The analysis phase calculates the AVF by analyzing the collected event times. The AVF infrastructure considers program-level effects such as first-level and transitive dynamic-dead instructions and logic masking. We measure AVFs only during portions of the workload that use the GPU.

Our model supports fault modes with arbitrary geometries, including contiguous and non-contiguous fault modes of any size in any structure including SRAMs and datapath latches. For clarity of description, this paper focuses on the most common multi-bit fault modes, contiguous Mx1 faults in SRAMs [17]. These faults are also the most problematic for conventional ECCs that protect data stored along a word-line.

B. Differences between Single-bit and Multi-bit AVF

As discussed in Section IV-D, MB-AVF can vary between 1x and Mx SB-AVF, where M is the number of bits in the fault mode. Figure 4 plots the 2x1 MB-AVF of the L1 cache with parity and three different x2 interleaving schemes: logical interleaving, *way-physical* interleaving, which interleaves lines from different ways in the same set, and *index-physical* interleaving, which interleaves lines from adjacent indices.

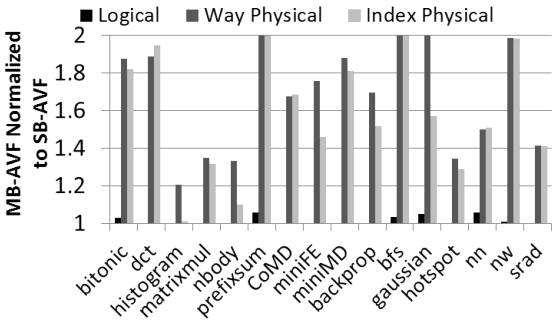
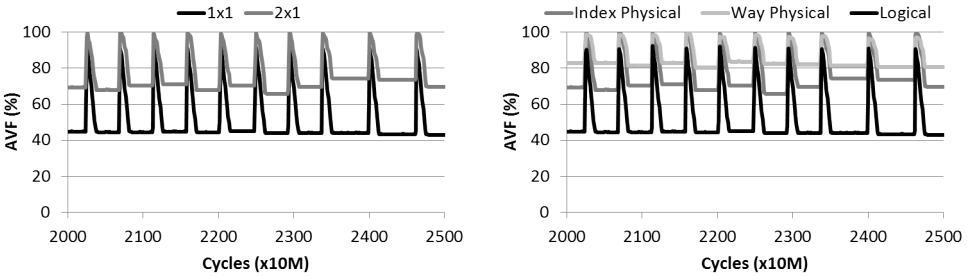


Figure 4: DUE MB-AVF for a 2x1 fault in the L1 cache varies between 1x and 2x the single-bit AVF, depending on both interleaving style and workload access pattern. Logical interleaving, which incurs additional area overhead, consistently has the lowest MB-AVF.

We draw three main conclusions from Figure 4. First, as expected from first principles, the 2x1 MB-AVF varies between 1x and 2x the single-bit AVF in the structure, but there is substantial variation in this ratio among workloads. Second, we see that logical interleaving consistently yields MB-AVFs very close to the theoretical minimum. This is because bits in the same cache line are often written and read close together in time. Therefore, these bits are more likely to be both ACE (necessary for correct program execution) or both unACE than bits from different cache lines. We refer to this property as *ACE locality*; Figure 4 shows that data in the same cache line has higher ACE locality than data in different cache lines. This effect is due to the union effect of multi-bit ACEness: if you physically interleave bits from different cache lines, the odds are high that most multi-bit faults will hit at least one ACE bit, causing the entire fault group to be ACE. Conversely, interleaving bits from the same cache line means that some multi-bit faults will be unACE when they strike a cache line that is entirely unACE.

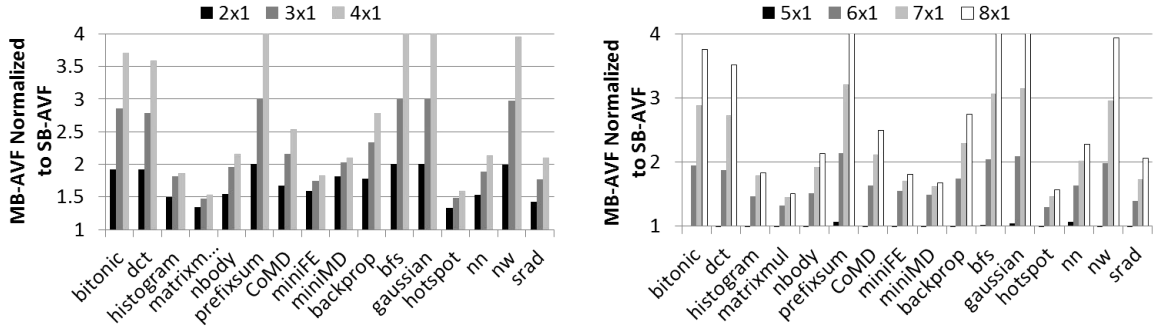
Third, we see that the MB-AVF of physical interleaving varies substantially based on the workload and style of interleaving (way vs. index). In general, interleaving across indices yields an MB-AVF closer to the theoretical minimum than interleaving across ways. This is due to workloads with strided access patterns which access different indices in succession. Thus, lines in adjacent indices are more likely to have high ACE locality than lines in the same index.

Figure 5 shows a specific example from the Mantevo MiniFE benchmark (a finite element analysis workload). Figure 5a plots the single-bit and two-bit AVF over time for the application. Both AVFs vary as the benchmark's cache usage varies over time. However, the ratio between MB-AVF and SB-AVF also changes based on application phase. During the phase where the SB-AVF is greater than 90%,



(a) 2x1 MB-AVF and SB-AVF with x2 index interleaving (b) 2x1 MB-AVF with different interleaving schemes

Figure 5: DUE SB-AVF and DUE MB-AVF in the L1 cache for the Mantevo MiniFE benchmark. The ratio between MB-AVF and SB-AVF changes during different phases of the application, as does the ratio between MB-AVF for different interleaving schemes.



(a) Parity (b) SEC-DED ECC

Figure 6: The effect of fault mode on DUE MB-AVF in the L1 cache with x4 way-physical interleaving. MB-AVF increases for larger fault modes because a larger fault is more likely to affect at least one ACE bit.

MB-AVF is 99%, or 10% higher than SB-AVF. In the phase when SB-AVF is 45%, MB-AVF is 71%, or 58% higher than the SB-AVF.

Similarly, Figure 5b plots MiniFE's 2x1 MB-AVF over time with x2 logical, way-physical, and index-physical interleaving. In one phase of the application, logical interleaving has a substantially lower MB-AVF than way-physical and index-physical interleaving. In the other phase, all interleaving styles have approximately the same MB-AVF.

Overall, for our workloads, way-physical interleaving had a 56% higher MB-AVF and index-physical interleaving had a 65% higher MB-AVF than logical interleaving in the L1 cache, because a cache with logical interleaving has higher ACE locality than a cache with physical interleaving. This is a general result that can be used to guide design: increasing the ACE locality in a structure will reduce its MB-AVF.

C. Effects of Fault Mode on Multi-bit AVF

Figure 6 shows the effect of fault mode on the MB-AVF for detected, uncorrected errors (DUE MB-AVF). Figure 6a shows MB-AVFs when using parity and x4 way-physical interleaving, and Figure 6b shows MB-AVFs when using SEC-DED ECC and x4 way-physical interleaving. We draw two main conclusions from these figures. First, the figures show that MB-AVF increases for larger fault modes because a larger fault group has a higher probability of containing an ACE bit. On average across all benchmarks, 4x1 MB-AVF is 2.74x SB-AVF when using parity protection, and this ratio varies from 1.52x to 4.0x among benchmarks. As discussed previously, the increase in MB-AVF with fault mode is much lower when using logical interleaving, but the trend of increased MB-AVF with fault mode size remains.

Second, MB-AVFs for Mx1 faults with SEC-DED ECC are similar to MB-AVFs of $M-I$x1 faults with parity, where I is the level of interleaving in the structure. This is because an 8x1 fault with SEC-DED ECC will be uncorrected in at most 4 cache lines, the same as a 4x1 fault with parity. From the figures, we observe that 8x1 MB-AVF with SEC-DED ECC is 2.7x SB-AVF, the same as 4x1 MB-AVF with parity.

VII. Extending MB-AVF to Undetected Errors

Undetected errors have the potential to cause silent data corruptions (SDCs). Even when using parity or ECC, a large enough multi-bit fault can defeat the protection and cause SDC. In absence of AVF estimates, a designer is typically forced to conservatively assume that an undetected multi-bit fault will cause SDC. This may lead to overdesign of a reliability scheme, wasting area and power. Therefore it is worth understanding SDC AVF behavior for multi-bit faults.

Unfortunately, there exists a complication to our approach for MB-AVF ACE analysis resulting from our method of using single-bit masking behavior to describe multi-bit

masking behavior. Specifically, a bit's ACEness may be changed by the presence of another fault. For example, a program may read the first two bytes in a cache line and XOR the least significant bit of both bytes. A single-bit fault in the least significant bit of either byte alone could result in SDC. A multi-bit fault covering both bits, however, will be unACE since the result of the XOR operation will be the same as in the fault-free case. We call this effect *ACE interference*. Because our model does not capture ACE interference, we cannot conclude that our model accurately measures SDC MB-AVF.

In Section VII-A, we evaluate this potential inaccuracy using a fault injection study. Our results show that ACE interference occurs extremely rarely and thus introduces negligible error in our model. Section VII-B extends our ACE analysis model to estimate SDC AVF, and Section VII-C presents our findings.

A. Evaluating Model Accuracy

To evaluate the accuracy of our SDC MB-AVF model, we inject faults into a representative SRAM structure, the GPU vector general-purpose register file (VGPR). We use a fault injection framework built into the simulator multi2sim [39]. The simulation infrastructure used in this study is different from that used for multi-bit AVF measurement, however this study is focused on examining program-level masking behavior, not microarchitectural behavior. The workloads for our study come from the AMD OpenCL sample suite [2].

To determine the prevalence of ACE interference, we first identify SDC ACE bits in each benchmark with a set of 5000 random single-bit fault injections. SDC ACE bits are determined by comparing program output to a baseline output obtained without injecting faults. We then create multi-bit fault groups using the identified SDC ACE bits and adjacent bits, and inject multi-bit faults into these fault groups. We report the number of fault groups that exhibit ACE interference. The fraction of fault groups with ACE interference estimates the probability of multiple faults interacting in such a way as to change the ACEness of surrounding bits. We intuitively expect such interaction to be low, allowing us to describe multi-bit ACE behavior using single-bit ACE behavior.

We perform multi-bit injections for 2x1, 3x1, and 4x1 fault modes and report the results in Table II. We report the number of identified SDC ACE bits for each benchmark and the number of fault groups with ACE interference for each fault mode.

Table II shows that out of 1730 total ACE bits, only 2 multi-bit fault groups (0.1%) showed ACE interference. The majority of workloads have no multi-bit fault groups that exhibit ACE interference. The interactions resulting in ACE interference are complex. In PrefixSum, for example, both the single-bit injection and multi-bit injection with ACE interference caused a change in control flow which caused program execution to diverge. For the multi-bit injection, execution converged again to the fault-free instruction stream 412 instructions later, eventually leading to the same output as the fault-free case. For the single-bit injection, the execution never converged, leading to incorrect output at the end of execution.

Benchmark	SDC ACE Bits	Multi-bit Fault Groups with ACE Interference		
		2x1	3x1	4x1
ScanLargeArrays	36	0	0	0
DCT	199	0	0	0
DwtHaar1D	12	0	0	0
FastWalshTransform	236	0	0	1
Histogram	300	0	0	0
MatrixTranspose	300	0	0	0
PrefixSum	300	0	1	0
RecursiveGaussian	47	0	0	0
MatrixMultiplication	300	0	0	0

Table II: ACE interference in multi-bit faults. SDC ACE bits were identified by single-bit fault injection. Only 0.1% of multi-bit fault groups containing SDC ACE bits exhibited ACE interference.

From this study, we conclude that using ACE analysis to estimate SDC AVF will have low error due to program-level interactions between multiple flipped bits. Therefore, multi-bit ACE analysis can serve as an accurate early estimate of SDC MB-AVF during architecture of a processor.

B. Estimating SDC MB-AVF

Determining SDC MB-AVF through ACE analysis is similar to the process for DUE MB-AVF, with a few differences. Again, two measurements must be considered when determining the effect of a protection domain overlapped with a fault group: the ACEness of the region and the action taken by the protection domain upon observing the fault. For DUE analysis, single-bit ACE analysis was required to determine the ACEness of regions and groups. For SDC analysis, program-level masking behavior (e.g., dynamic deadness [25]) must be accounted for as part of the single-bit AVF ACE analysis.

The ACEness of an overlapped region is determined using equation 5. Using this information, the reaction of the protection domain and the results of the program-level masking analysis, the region can be classified as: unACE, true DUE ACE, false DUE ACE, or SDC ACE. SDC ACE results when a region is undetected and affects program output. A region that is undetected but does not affect program output is unACE. The ACEness of the fault group is then the ACEness of the worst-case region in the group. For instance, if any region in a group is SDC ACE, the group is SDC ACE. Next in order of precedence is true DUE ACE, false DUE ACE, and unACE.

In cache structures, we classify a fault group with both SDC and DUE ACE overlapped regions as SDC ACE since we cannot guarantee that the DUE ACE region will be detected before the SDC ACE region has propagated to program output. This simplification is not fundamental to

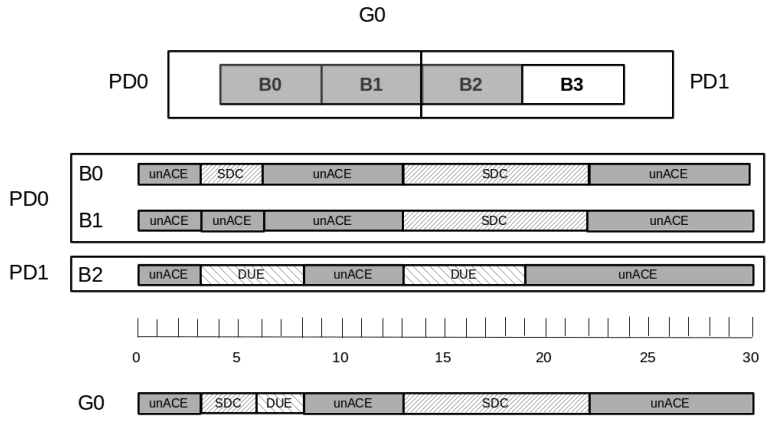
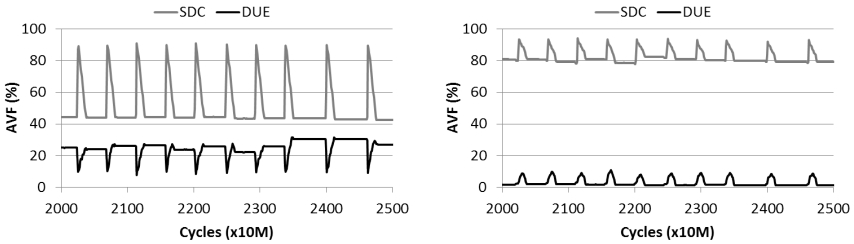


Figure 7: MB-AVF ACE Analysis of a 3x1 Fault over two parity protection domains. A fault that affects B0-B2 will be undetected in PD0, causing SDC if either B0 or B1 is ACE. Time that is SDC ACE in PD0 and DUE ACE in PD1 is classified as SDC ACE in group G0 since we cannot guarantee detection of the error in PD1 before data in PD0 propagates to program output.



(a) Parity, x2 Index-Physical Interleaving (b) Parity, x2 Way-Physical Interleaving

Figure 8: SDC MB-AVF for 3x1 faults in the L1 cache for MiniFE. Without MB-AVF analysis, designers would conservatively assume that all 3x1 faults caused SDC. In reality, MiniFE sees SDC MB-AVFs between 50% and 95%, and also DUE MB-AVFs of 5-30%. Furthermore, index-physical interleaving has a 1.8x lower SDC MB-AVF than way-physical interleaving.

our approach, and we relax this for other structures in Section VIII.

Figure 7 illustrates this process with an example 3x1 fault group. This figure depicts fault group G0 overlapping parity protection domains PD0 and PD1 over a period of 30 cycles. To calculate the MB-AVF of these bits, ACE analysis is used to calculate the ACEness of this group over this 30 cycle period. We first perform single-bit ACE analysis to determine ACE and unACE cycles for each individual bit. We then analyze each overlapped region between the fault group and any protection domains separately. Consider the overlap with PD0, which is a parity protection domain containing two bits from the fault group. If a 3x1 multi-bit fault occurs, this protection domain will not be able to detect this fault. Because of this we can classify the ACE time in bits B0 and B1 as SDC. Considering the overlap with PD1, which contains one bit from the fault group, we can determine the protection domain will detect all errors upon reading the data. We can therefore classify all ACE time as DUE. Because of the precedence described earlier, all cycles resulting in SDC in at least one overlapped region are SDC ACE in the group, and all cycles with no SDC ACE regions but at least one DUE ACE region are DUE ACE in the group.

C. SDC MB-AVF Results

Figure 8 shows DUE and SDC AVF for 3x1 faults with parity and x2 index-physical and way-physical interleaving when running MiniFE. The figure shows that SDC MB-AVF is significantly higher than DUE MB-AVF for both interleave modes, but there is a non-trivial rate of DUE for 3x1 faults. This is because a 3x1 fault group may contain one DUE ACE line and zero SDC ACE lines. Each 3x1 fault can result in SDC in one fault group (two flipped bits), and DUE in another fault group (one flipped bit). Whether the SDC and DUE fault groups are ACE or unACE determines the relative values of the SDC and DUE MB-AVFs, and depends on the access pattern of the workload. In the absence of MB-AVF measurements, designers are typically forced to assume that all 3x1 faults cause SDC. Our results show that this assumption overestimates the SDC rate by up to 2x for MiniFE. Moreover, this assumption underestimates the DUE rate because DUE MB-AVFs are 5-30% in MiniFE. Other benchmarks exhibit similar behaviors.

Figure 9 plots the MB-AVF for 5x1 through 8x1 fault modes with SEC-DED ECC and x2 way-physical interleaving for all benchmarks. There are three interesting findings in this figure. First, the figure shows that the SDC AVF increases substantially from 5x1 to 6x1 faults. With x2

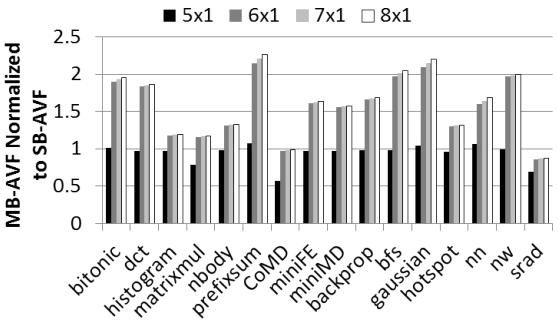


Figure 9: SDC MB-AVF for 5x1 through 8x1 faults with SEC-DED ECC and x2 interleaving. The SDC MB-AVF plateaus for larger fault modes due to high ACE locality within cache lines.

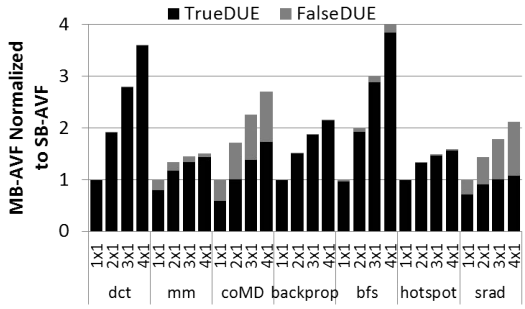


Figure 10: False DUE is a small contributor to AVF on average, but it doubles the DUE AVF in certain benchmarks. Changes in false DUE across fault modes depend on workload access pattern.

interleaving, 5x1, 6x1, 7x1, and 8x1 faults each affect exactly two cache lines. With a 5x1 fault, however, one of the two cache lines will detect the fault since it has only two erroneous bits. Therefore, some of the 5x1 MB-AVF is DUE. 6x1 faults, by contrast, are undetected in all affected cache lines, and thus all 6x1 MB-AVF is SDC.

Second, the SDC AVF plateaus or shows only a slight increase from 6x1 to 8x1 faults because bits within a cache line tend to have high ACE locality, and all three of these fault modes affect the same number of cache lines. Therefore, 8x1 faults do not cause substantially more SDC cases than 6x1 faults. This is a general result for any structure that exhibits high ACE locality within a protection domain.

D. Measuring False DUE

Figure 9 also shows that the 5x1 SDC MB-AVF is sometimes lower than the DUE SB-AVF (depicted as bars less than 1 in the figure). This is a consequence of *false DUE*. False DUE are detected, uncorrected errors that would not have resulted in incorrect program output if left undetected. This quantifies the expected increase in error rate when adding protection. Figure 10 shows the amount of true and false DUE in the L1 cache for our workloads, plotted by fault mode. Benchmarks not shown on this plot had negligible levels of false DUE.

On average, false DUE is a small contributor to overall DUE. However, certain benchmarks exhibit high levels of false DUE. In CoMD, 41% of the single-bit DUE AVF is false DUE. The contribution decreases slightly for 4x1 faults, where 36% of the 4x1 DUE MB-AVF is false DUE. In srad, on the other hand, 29% of the single-bit DUE AVF is false DUE, while 50% of the 4x1 DUE MB-AVF is false DUE. These differing trends are the result of different cache access patterns between the two workloads.

Fault Mode	1x1	2x1	3x1	4x1	5x1	6x1	7x1	8x1
Fault Rate	96.4	3.0	0.2	0.1	0.1	0.05	0.05	0.025

Table III: Fault rates used for our study. We set the total fault rate to 100 and derive multi-bit fault percentages from [17].

VIII. Case Study: Using MB-AVF for Design

In this section, we demonstrate the use of multi-bit AVF and multi-bit ACE analysis during the design process. We focus on specifying protection to meet SDC targets in the GPU vector general-purpose register file (VGPR), one of the largest on-chip SRAM structures [1]. This is typical of studies done during the architecture stage of a processor's design cycle. The goal is to minimize overall die area spent on reliability while achieving specified SER targets. For this study, we compare MB-AVF analysis to using single-bit AVF to approximate MB-AVFs for all fault modes, a typical approximation made in absence of MB-AVF analysis.

As shown in Table III, we assume a total fault rate of 100 broken down into single-bit and multi-bit faults up to 8 bits as per the study by Ibe et al. for a 22nm process technology [17]. We assume that each 32-bit register has its own ECC or parity. We compare SEC-DED ECC, which has an area overhead of 21.9%, to parity, which has an area overhead of 3.1%. We assume that registers are interleaved together to protect against multi-bit faults. We model two different styles of interleaving: *intra-thread interleaving*, where different registers from the same GPU thread are interleaved together (e.g., R0 and R1 from thread 0); and *inter-thread interleaving*, where the same register from different GPU threads are interleaved together (e.g., R0 from thread 0 and R0 from thread 1).

In Section VII-B, our modeling treated a fault group with overlapping SDC ACE and DUE ACE as SDC ACE because we could not guarantee detection of the DUE before the SDC propagated to program output. A GPU, however, reads and writes registers from multiple threads simultaneously. In our model, operations occur on 16 threads at a time. Therefore, when inter-thread interleaving is performed within groups of 16 threads, a fault group in the VGPR with overlapping SDC ACE and DUE ACE will detect the DUE before the SDC propagates to program output. We account for this when modeling the MB-AVF of inter-thread interleaving.

A. Results

Figure 11 shows SDC rates for several design points, summed over all fault modes and averaged across benchmarks, from MB-AVF analysis as well as from using SB-

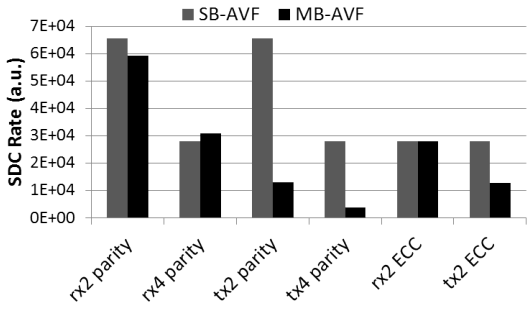


Figure 11: SDC MB-AVF of the GPU VGPR with parity or ECC and intra-thread x2 and x4 (rx2 and rx4), and inter-thread x2 and x4 (tx2 and tx4) interleaving. MB-AVF analysis yields lower SDC rates than approximating MB-AVFs with SB-AVFs, and shows that inter-thread interleaving has lower SDC than intra-thread.

AVF to approximate MB-AVFs. The figure provides two interesting insights. First, applying MB-AVF analysis can reduce SDC estimates relative to approximating MB-AVFs from SB-AVFs. This is because MB-AVF analysis correctly identifies more multi-bit faults that cause DUE, rather than SDC, and classifies a larger fraction of total AVF as DUE AVF rather than SDC AVF.

Second, the figure shows that parity with x4 inter-thread interleaving has substantially lower SDC than ECC with x2 interleaving, including 86% and 71% lower SDC than ECC with intra-thread and inter-thread interleaving, respectively.

This result is due to two effects. First, inter-thread interleaving outperforms intra-thread interleaving because most fault groups that contain SDC ACE bits also contain DUE ACE bits. Therefore, many SDCs in those fault groups are converted to DUEs by an adjacent thread's DUE when inter-thread interleaving is used. Second, ECC may not detect any fault greater than its detection capability (e.g., ECC with x2 interleaving can miscorrect 6x1 and 7x1 faults). Parity, on the other hand, guarantees detection of all faults with an odd number of bit flips (e.g., parity with x2 interleaving detects all 6x1 faults and many 7x1 faults, since both fault modes create at least one overlapped region with a 3-bit fault). Therefore, as large multi-bit faults become more common, parity may have a detection advantage over ECC.

One implication of this result is that parity may be a better choice than ECC in systems where detection is the primary concern, such as systems with higher-level rollback/recovery mechanisms. In addition, future systems may be better off decoupling detection from correction (e.g., [20]) in order to meet reliability targets.

IX. Conclusion

As spatial multi-bit faults become more common, it will be more important to have accurate analyses of their impact on a system. Estimating multi-bit AVFs allows architects to model the impact of multi-bit faults at design time and to deploy appropriate strategies to protect their designs. In this paper, we proposed a method to quantify architectural vulnerability factors for spatial multi-bit faults (MB-AVFs). We proposed an ACE analysis methodology to measure multi-bit detected uncorrected error AVFs and found that these MB-AVFs vary independently from single-bit AVFs. We also extended this methodology to estimate multi-bit silent data corruption AVFs and confirmed that the model estimates SDC AVFs to reasonable precision. Finally, we demonstrated the use of MB-AVFs to make area and reliability tradeoffs during processor design.

References

[1] AMD, "AMD graphics cores next (GCN) architecture," http://www.amd.com/us/Documents/GCN_Architecture_whitepaper.pdf.

[2] ——, "OpenCL accelerated parallel processing (app) SDK."

[3] R. F. Barrett, M. A. Heroux, P. T. Lin, C. T. Vaughan, and A. B. Williams, "Mini-applications: Vehicles for co-design," in *Proceedings of the 2011 Companion on High Performance Computing Networking, Storage and Analysis Companion*, 2011, pp. 1–2.

[4] F. Bauer, G. Georgakos, and D. Schmitt-Landsiedel, in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, L. Svensson and J. Monteiro, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. A Design Space Comparison of 6T and 8T SRAM Core-Cells, pp. 116–125.

[5] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sept. 2005.

[6] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[7] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. S. Mukherjee, and R. Rangan, "Computing architectural vulnerability factors for address-based structures," in *International Symposium on Computer Architecture (ISCA-32)*, 2005.

[8] A. Biswas, P. Racunas, J. Emer, and S. Mukherjee, "Computing accurate AVFs using ACE analysis on performance models: A rebuttal," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 21–24, 2008.

[9] A. Biswas, N. Soundararajan, S. S. Mukherjee, and S. Gurumurthi, "Quantized AVF: A means of capturing vulnerability variations over small windows of time," in *Workshop on System Effects of Logic Soft Errors (SELSE-5)*, 2009.

[10] S. Che, M. Boyer, M. Jiayuan, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IEEE International Symposium onWorkload Characterization*, Oct. 2009, pp. 44–54.

[11] C. Constantinescu, M. Butler, and C. Weller, "Error injection-based study of soft error propagation in AMD Bulldozer microprocessor module," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2012, pp. 1–6.

[12] A. Dixit, R. Heald, and A. Wood, "Trends from ten years of soft error experimentation," in *Silicon Errors in Logic - System Effects (SELSE), 2009 IEEE Workshop on*, 2009.

[13] N. Farazmand, R. Ubal, and D. Kaeli, "Statistical fault injection-based analysis of a GPU architecture," in *Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2012.

[14] N. J. George, C. R. Elks, B. W. Johnson, and J. Lach, "Bit-slice logic interleaving for spatial multi-bit soft-error tolerance," in *International Conference on Dependable Systems and Networks (DSN)*, 2010, pp. 141–150.

[15] ——, "Transient fault models and AVF estimation revisited," in *International Conference on Dependable Systems and Networks (DSN)*, 2010, pp. 477–486.

[16] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in GPGPU," in *Proceedings of the IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID)*, 2010, pp. 691–696.

[17] E. Ibe, H. Taniguchi, Y. Yahagi, K.-I. Shimbo, , and T. Toba, "Impact of scaling on neutron-induced soft error in srams from a 250 nm to a 22 nm design rule," in *Electron Devices, IEEE Transactions on*, Jul 2010, pp. 1527–1538.

[18] H. Jeon, M. Wilkening, V. Sridharan, S. Gurumurthi, and G. Loh, "Architectural vulnerability modeling and analysis of integrated graphics processors," in *Workshop on Silicon Errors in Logic - System Effects (SELSE)*, Stanford, CA, March 2012.

[19] H. Jeon and M. Annavaram, "Warped-DMR: Light-weight error detection for GPGPU," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, Dec 2012, pp. 37–47.

[20] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar, "Low-power, low-storage-overhead chipkill correct via multi-line error correction," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13)*, 2013, pp. 24:1–24:12.

[21] S. Kim and A. K. Somani, "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," in *International Conference on Dependable Systems and Networks (DSN-32)*, 2002.

[22] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN)*, 2004, pp. 145–154.

[23] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Architecture-level soft error analysis: Examining the limits of common assumptions," in *International Conference on Dependable Systems and Networks (DSN-37)*, 2007.

[24] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, "Characterization of multi-bit soft error events in advanced SRAMs," in *Digest of Electron Devices Meeting*, December 2003, pp. 21.4.1–21.4.4.

[25] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *International Symposium on Microarchitecture (MICRO-36)*, 2003.

[26] A. A. Nair, S. Eyerman, L. Eeckhout, and L. K. John, "A first-order mechanistic model for architectural vulnerability factor," in *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA)*, 2012, pp. 273–284.

[27] E. Normand, "Single event upset at ground level," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, Dec 1996.

[28] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *IEEE Transactions on Reliability*, pp. 114–122, Apr 1990.

[29] N. Seifert, B. Gill, S. Jahinuzzaman, J. Basile, V. Ambrose, S. Quan, R. Allmon, and A. Bramnik, "Soft error susceptibilities of 22nm tri-gate devices," *IEEE Transactions on Nuclear Science*, pp. 2666–2673, Dec 2012.

[30] S. Shazli, M. Abdul-Aziz, M. Tahoori, and D. Kaeli, "A field analysis of system-level effects of soft errors occurring in microprocessors used in information systems," in *IEEE International Test Conference (ITC)*, 2008, pp. 1–10.

[31] C. Slayman, "Soft error trends and mitigation techniques in memory devices," in *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS)*, Jan. 2011, pp. 1–5.

[32] V. Sridharan and D. R. Kaeli, "Eliminating microarchitectural dependency from architectural vulnerability," in *International Symposium on High Performance Computer Architecture (HPCA-15)*, 2009, pp. 117–128.

[33] ——, "Using hardware vulnerability factors to enhance AVF analysis," in *International Symposium on Computer Architecture (ISCA-37)*, 2010, pp. 461–472.

[34] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng shui of supercomputer memory: Positional effects in DRAM and SRAM faults," in *Proceedings the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13)*, 2013, pp. 22:1–22:11.

[35] J. Suh, M. Annavaram, and M. Dubois, "MACAU: A Markov model for reliability evaluations of caches under single-bit and multi-bit upsets," in *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture (HPCA)*, 2012, pp. 1–12.

[36] J. Suh, M. Manoochehri, M. Annavaram, and 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)*, 2011, pp. 85–96.

[37] L. G. Szafaryn, B. H. Meyer, and K. Skadron, "Evaluating overheads of multibit soft-error protection in the processor core," *IEEE Micro*, pp. 56–65, July-Aug 2013.

[38] J. Tan, N. Goswami, T. Li, and X. Fu, "Analyzing soft-error vulnerability on GPGPU microarchitecture," in *Proceedings of the IEEE International Symposium on Workload Characterization*, June 2011, pp. 226–235.

[39] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2Sim: A simulation framework for CPU-GPU computing," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2012.

[40] K. R. Walcott, G. Humphreys, and S. Gurumurthi, "Dynamic prediction of architectural vulnerability from microarchitectural state," in *International Symposium on Computer Architecture (ISCA-34)*, 2007, pp. 516–527.

[41] N. J. Wang, A. Mahesri, and S. J. Patel, "Examining ACE analysis reliability estimates using fault-injection," in *International Symposium on Computer Architecture (ISCA-34)*, 2007, pp. 460–469.

[42] C. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, "Techniques to reduce the soft error rate of a high-performance microprocessor," in *International Symposium on Computer Architecture (ISCA-31)*, 2004, pp. 264 – 275.

[43] Y. Zhang, S. Ghosh, J. Huang, J. W. Lee, S. A. Mahlke, and D. I. August, "Runtime asynchronous fault tolerance via speculation," in *Proceedings of the 10th International Symposium on Code Generation and Optimization (CGO)*, 2012, pp. 145–154.