Data Block Partitioning Methods to Mitigate Stuck-At Faults in Limited Endurance Memories

Jiangwei Zhang, *Student Member, IEEE*, Donald Kline, Jr., *Student Member, IEEE*, Liang Fang, Rami Melhem, *Fellow, IEEE*, and Alex K. Jones, *Senior Member, IEEE*

***Abstract*— Deep scaling in conjunction with increased process variation has resulted in increasingly faulty memories. Emerging memories, particularly phase-change and resistive memories, can experience stuck-at faults due to limited endurance. Partition and flip (PAF) schemes partition data into blocks and invert these blocks as needed to ensure data that is written matches the stuck-at cells. In this paper, we propose two novel correction schemes that substantially enhance the fault-tolerance capabilities of existing PAF techniques. First, dynamic partitioning increases the number of possible configurations with equivalent auxiliary bits. At high fixed error rates, the increase in configurations results in improved write error rates for flip-N-write and Aegis partitioning by 7%–72% and 5–53×, respectively. Our second novel partitioning method, relaxed partitioning, dramatically and effectively increases the partitioning search space by specifying minimally overlapping configurations. Through Monte Carlo simulations, data-aware dynamic partitioning tolerates 25% and 27% more faults over its lifetime than Aegis with 36 and 43 auxiliary bits per 512-bit data block, respectively, while relaxed partitioning achieves an extra 15% and 24% additional improvement while requiring two fewer overhead bits per data block.**

***Index Terms*— Dynamic partitioning, emerging memories, reliability, stuck-at faults.**

I. Introduction

DRAM and Flash memory encounter significant challenges due to technology scaling. This scaling results in increased process variation and leads to wider divergence of capacitance of memory cells. This results in more "weak" cells that have either reduced data retention time or poor charge sensing fidelity, which leads to increasing numbers of memory faults [2], [3].

Emerging memories, such as phase-change memory (PCM) and memristors (RRAM), provide great scalability, high density, and nonvolatility making them promising candidates to replace DRAM and Flash in main memory or secondary storage applications [4], [5]. Unfortunately, both PCM and RRAM suffer from limited write endurance [6]. For example, a typical PCM cell can sustain from 10^8 to 10^9 writes before it becomes stuck at either '0' or '1.'

Writes to PCM or RRAM cells typically require a RESET operation. As the device is cycled through RESET and SET operations, it becomes increasingly resistant to these operations, ultimately becoming impervious to the RESET operation. This results in being stuck-at a particular value. Process variation exacerbates the situation. While each individual cell has an optimal RESET current, each cell's optimal RESET current deviates from the group average due to process variation. Cells with optimal RESET currents that are relatively far from the average can quickly become impervious to the normalized optimal RESET current, leading to early endurance failures. We call these cells "weak cells."

Stuck-at cells can still be accessed, but writes will always produce a fault as the cell cannot be overwritten. However, when attempting a write operation to a stuck-at bit, if the written value is the same as the stuck-at value [stuck-at-right (SA-R)], the fault does not result in an error, but if the two values are opposed [stuck-at-wrong (SA-W)], the fault will result in an error. In the latter case, the bit can always be stored as its inverse, as long as it can be marked as the inverse using auxiliary information. In this way, regardless of the stuck-at state, the bit can still be stored.

Encoding and correction schemes can be used to mitigate stuck-at faults; example methods for this approach include error correction codes (ECC) [7], coset encoding [8], and PRES [9]. ECC can protect against stuck-at faults; however, ECC is typically employed to protect memory against transient or bus-related faults [10], [11], which are relatively rare compared to stuck-at faults in PCM or RRAM. Error correction pointers (ECPs) [12] were proposed for PCM, where ECC was insufficient for stuck-at faults. ECP uses a pointer to address the stuck-at bit and an extra spare data bit to replace the faulty bit within its protected data block.

Manuscript received February 2, 2018; revised May 16, 2018 and June 21, 2018; accepted July 18, 2018. Date of publication August 13, 2018; date of current version October 23, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61332003, in part by the U.S. NSF Graduate Research Fellowship under Grant 1747452, and in part by SHREC industry and agency members and the I/UCRC Program of the National Science Foundation under Grant CNS-1738783. This paper was presented at the 2017 IEEE/ACM International Conference on Computer Aided Design [1]. *(Corresponding author: Jiangwei Zhang.)*

J. Zhang is with the National University of Defense Technology, Changsha 410073, China, and also with the ECE Department, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: jiz148@pitt.edu).

D. Kline, Jr., and A. K. Jones are with the ECE Department, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: dek61@pitt.edu; akjones@pitt.edu).

L. Fang is with the National University of Defense Technology, Changsha 410073, China (e-mail: lfang@nudt.edu.cn).

R. Melhem is with the CS Department, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: melhem@cs.pitt.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2858186

Another method, to correct stuck-at faults in particular, is partition-and-flip (PAF) [13]–[15]. PAF approaches attempt to leverage the errorfree characteristic of SA-Rs and mask errors through encoding to eliminate SA-Ws. They first use an efficient method to partition SA-Rs and SA-Ws into different groups and then invert groups with SA-Ws, recording the PAF setting using auxiliary bits to allow the original data to be restored. Flip-N-write [13] (FNW) is a simple example of a PAF approach. Aegis [15] employs a more sophisticated partitioning method for better protection against stuck-at faults at the expense of additional space and runtime overheads. Aegis utilizes a method that requires partitioning a block into a prime number of groups making it inefficient to encode. This results in "wasting" possible configurations that can be encoded by the auxiliary bits but that Aegis cannot utilize. In general, PAF schemes can provide a similar tolerance of stuck-at faults with lower auxiliary storage overhead requirements to schemes, such as ECP or ECC [15].

In this paper, we propose a partitioning method that dynamically changes the number of groups and size of each group to mitigate more stuck-at faults in a data block. The dynamic partitioning approach can be applied to existing PAF schemes and provides an additional degree of freedom to improve their effectiveness. In some cases, dynamic partitioning can leverage the unused encoding states in schemes such as Aegis. Dynamic partitioning relaxes the requirements of some PAF schemes designed to create a fault correction guarantee. For example, Aegis creates perfectly nonoverlapping configurations to allow isolation of faults, which is the principle of Aegis and creates a particular fault correction guarantee. However, as long as a configuration exists such that all groups contain only SA-R or SA-W faults, the data can be encoded and stored correctly. Thus, dynamic partitioning creates more partitioning choices to increase the search space to identify such a configuration. Moreover, we provide a relaxed partitioning method to maximize the potential for a more diverse search space. In particular, this paper makes the following contributions.

1) A novel *dynamic partitioning* scheme to enhance stuck-at fault tolerance through variation of group size is proposed for the PAF methodology.
2) A *relaxed partitioning* scheme that further improves the effectiveness of dynamic partitioning is proposed that relaxes the nonoverlapping requirement for a particular group size.
3) A *combined* dynamic and relaxed partitioning approach is proposed that utilizes both nonuniform group size while relaxing the nonoverlapping requirement within the same framework.
4) A *detailed characterization* of stuck-at fault tolerance, which illustrates the significant improvements of dynamic partitioning and relaxed partitioning in the context of faulty auxiliary bit storage.

The remainder of this paper is organized as follows. In Section II, we provide a background and discuss the related work on fault-tolerance schemes applied to stuck-at faults in emerging memories. In Section III, we describe the dynamic partitioning scheme in detail [1]. In Section IV, we introduce relaxed partitioning to enhance fault-tolerance effectiveness. Section V provides the experimental methodology and results of the proposed schemes compared to the current leading approaches. In Section VI, we relate conclusions and potential future directions.

II. Background

To discuss the details of previous work and our proposed approaches, we must first define some terminology. Then, we discuss prior work in both error correction and error mitigation for limited endurance memories.

A. Preliminaries

To allow terms that could otherwise be vague or have multiple meanings to be defined precisely for the description of PAF schemes, we define the following terms.

Definition 1 (Block): A block is an n-bit unit of data that is accessed as a single operation.

Definition 2 (Configuration): A configuration describes one way that the bits of a block can be divided into equal size groups. In particular, a configuration is a set of groups such that all n-bits in the block are a member of exactly one group and all groups have the same number of bits.[1]

Definition 3 (Partition): A partition describes a set of configurations where the number of groups and group size is fixed.

Thus, a configuration describes one way to divide the bits of a block into groups. A partition describes several ways to divide the bits of the block into groups of the same size. For example, if a block consists of 32 bits, then there are multiple configurations that divide the 32 bits into four groups of eight bits each. A partition would consist of several of these configurations. Another partition would consist of configurations that divide the 32 bits into five groups of seven bits each (where one group will have three virtual bits[1]).

B. Error Mitigation for Limited Endurance Memories

Error correction codes (ECC) [7] are general approaches that are used to protect memories from transient faults but can also be applied to correct stuck-at faults. Among these, single-error correction double-error detection (SECDED) ECC (64, 72) based on Hamming codes is the most popular form of ECC. When applied for stuck-at faults, it can recover one SA-W and tolerate any number of SA-R within the original 64 data bits and the eight additional parity bits. When the memory fault rate is $<10^{-6}$, SECDED ECC is sufficient for fault recovery [16] as the probability of more than one stuck-at fault appearing in a data block is below 10^{-12}. As memories with limited endurance start to experience early failures of weak cells due to process variation, the fault rate can begin to exceed 10^{-6}. This is exacerbated by spatial correlation such that faults can be clustered within data blocks, quickly exceeding ECC's capability. Moreover, while data bits have a relatively low probability of being written due to data locality, ECC parity bits are written much more frequently, encouraging faster wear, and further limiting fault-tolerance in this scenario.

[1]If n is not divisible by the number of groups, additional virtual (unused) bits may be added to make the groups even.

ECP [12], [17] uses auxiliary bits to record the location of one or more faulty bits within the row and includes the same number of spare bits to store the correct data. For a 512-bit data block, to protect f faulty bits, ECP requires $10f + 1$ *fault-free* bits to ensure protection.

As previously discussed, Aegis [15] and FNW [13] are PAF approaches to improve fault tolerance in the presence of stuck-at faults. Additional PAF approaches include RDIS [18] and SAFER [14]. RDIS transforms a 1-D data block into a 2-D matrix in an attempt to distribute SA-Rs and SA-Ws into different groups. Each row or each column of this matrix requires a small z-bit counter to conduct recursive inversion. Given the large number of counters, this can result in a significant overhead. For example, to guarantee 3-bit correction, a 512-bit data block is logically organized into a 32×16 matrix and requires 2-bit counters. The resulting overhead is $(32 + 16) \times 2 = 96$ bits or an 18.7% overhead. RDIS can potentially correct more faults than its guarantee if the additional faults happen to fall into the right groups.

SAFER assumes a particular number of available configurations with each group containing a pointer to the fault within the group, similar to ECP. Whenever a new fault occurs, the groups are repartitioned using the `xor` of the pointers to the fault locations in the data blocks. The goal is to partition the faults such that the number of stuck-at cells per group is $\leq$1. If the number of faults exceeds the number of groups, correction may be possible, but it is not guaranteed.

Aegis uses a prime interleaving principal to create unique configurations. It interprets a 1-D data block as a 2-D matrix. Inspired by the principle that any two points in a line on a Cartesian plane determine the slope of the line, Aegis uses different slopes to generate different configurations. If the matrix has prime numbers of rows and columns and the number of rows is greater than the number of columns, Aegis ensures that all possible combinations of two bits, which are in the same group of one configuration, would not be in the same group of the other configurations. The slope k represents an integral offset by a prime value (the length A of the X dimension of the matrix) and a prime modulus value (the length B of the Y dimension of the matrix) to form each group. This ensures that for the first element of each group (its starting position) results in unique group members for each different slope value. In Fig. 1, as an example, we illustrate how to partition a 32-bit data block into 7×5 matrices according to Aegis for two slope values of $k = 0, 1$. The different symbols represent the members of each group for the different slopes, which except for the initial element, are different for the different slopes. If there are f stuck-at faults in a data block, Aegis guarantees that the faults will be partitioned into different groups through this partitioning uniqueness when there are at least $(f(f-1)/2) + 1$ possible slopes or group configurations [15]. Compared to SAFER, Aegis creates a better distribution of stuck-at faults with equivalent capacity overhead.

To provide context of the overhead for these schemes, to correct three faults, SAFER, Aegis, ECP, and RDIS require 14, 25, 31, and 96 bits, respectively. However, to correct six faults, Aegis, SAFER, and ECP require 27, 55, and 61 bits, respectively. Thus, SAFER is effective for correcting relatively few faults, and Aegis is effective for correcting larger numbers of faults. More details on the capabilities and overheads of these schemes are shown in Section V.

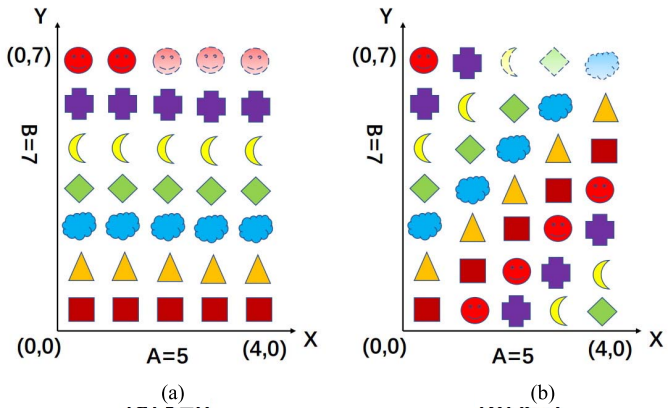

Fig. 1. Example of Aegis partitioning [15] of a 32-bit block into 7×5 matrices with different slopes. Each bit is included in a configuration as represented by a symbol. The symbols with dotted outlines correspond to unused bits (i.e., 33–35th bits). There are seven configurations in the partition, corresponding to seven slopes (i.e., $0 \leq k \leq 6$). (a) $k = 0$. (b) $k = 1$.

III. Dynamic Partitioning Scheme

The goal of dynamic partitioning is to distribute stuck-at faults to different groups by dynamically changing group sizes and orientations within an existing PAF scheme. This can improve the fault tolerance without increasing space overhead. Thus, we define the following.

Definition 4 (Dynamic Partition): A dynamic partition describes a set of partitions where each partition has a different number of groups and group size.

A dynamic partition can be represented by splitting the auxiliary bits into two segments: the first segment, S1, specifies how many unique partitions are used and the second segment, S2, contains the auxiliary bits needed to specify the makeup of the configurations within the partition and subsequently flip each of the groups within that partition. In Sections III-A and III-B, we demonstrate dynamic partitioning by applying the concept to existing PAF strategies of block-based partitioning (i.e., FNW) and prime interleaved partitioning (i.e., Aegis), respectively.

A. Dynamic Partitioning for FNW

FNW is traditionally implemented as a static block-based partition. For example, a 32-bit data block with eight auxiliary bits groups blocks of four adjacent bits. Each group has a corresponding auxiliary bit, which indicates whether the entire group is flipped or not to attempt to avoid writing any SA-W in the block. FNW fails when there is at least one SA-R and SA-W bit in the same group. Dynamic partition-based FNW (FNW_{DY}) is a simple and effective method to explain the power of our dynamic partitioning scheme.

Fig. 2 shows how FNW_{DY} would protect this 32-bit data block assuming an allocation of 10 auxiliary bits. The auxiliary bits are partitioned, such that the leading two bits (S1) are used to identify the number of groups used in the partition and the

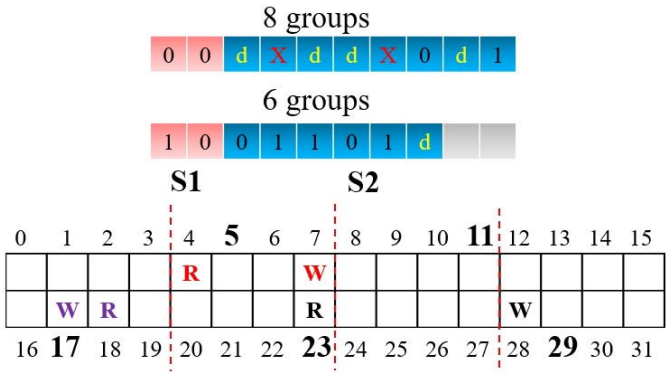

Fig. 2. Illustration of dynamic FNW protecting a 32-bit data block with 10 auxiliary bits. The smaller FNW partition size outperforms the larger one.

remaining eight bits (S2) control the inversion. The S1 bits count down from the maximum number of partitions, in this case eight. Thus, for the example, the data can be partitioned into eight, seven, six, and five groups. S1="00" represents an eight-group partition (8–0) in the data block with the partition of the data bits shown in the red dotted lines. Unfortunately, groups containing bits 4–7 and 16–19 contain both SA-R and SA-W, which prevents the success of this encoding (marked as X). S1="10" represents a six-group partition where the lower two flip bits are ignored. In this case, the partition divides the SA-R and SA-W bits into separate groups, allowing encoding to proceed. Here, the blocks end on the **bolded** numbers.

In general, for a large block, using a small number of bits to indicate the partition (adjusting the group size) can be more beneficial than using those bits to have a fixed partition with more, smaller groups. More discussion of dynamic FNW can be found here [1]. There are clear limitations to the FNW policy even with dynamic partitions due to its block-based nature, particularly as the number of faults increases. Thus, we apply dynamic partitioning to another PAF scheme with more partitioning flexibility, namely Aegis, in Section III-B.

B. Dynamic Partitioning for Aegis

The number of configurations available to Aegis is determined by the height of the 2-D rectangle used to organize the bits in the data block. The partitioning of Aegis has two restrictions on this rectangle height: it must be a prime number and this prime number must be greater than or equal to the width of the rectangle. As a result, any two bits in the same group of one configuration will not be in the same group in another configuration. Thus, this partitioning guarantees the correction of a certain number of faults by partitioning each fault into a different partition [15].

However, when the number of faults exceeds this guarantee, it may still be possible to correct these faults if they are partitioned to group SA-R's and SA-W's, but with Aegis, there is a limited search space to accomplish this. The dynamic partition strategy applied to Aegis does not limit itself to prime heights. Instead, it uses multiple combinations of prime and nonprime heights for a larger number of partition configurations. Thus, our strategy can leverage the existing advantages of static Aegis while allowing for many more partitioning options in a more flexible manner than partitioning schemes such as SAFER and RDIS.

Fig. 3 shows how a 32-bit data block is partitioned into different configurations according to *static* (original) Aegis and *dynamic* Aegis (Aegis_{DY}). For this block size, Aegis uses 10 auxiliary bits for fault correction, including seven flag (inversion) bits and three slope bits to indicate slopes from 0 to $B-1$. For each slope, the data block is partitioned into a 7×5 matrix. The width (A) and the height (B) of the various matrices are marked, while the available slopes (k) are also shown. The numbers for the bits in the matrices represent their group identifiers (IDs) for the slope shown in red and recorded in the slope bits ("000"). The bits labeled 'X' are bits which are not in the 32-bit data block.

In the case shown, Aegis_{DY} uses two bits in S1 and the other eight bits in S2. S2 is then partitioned into two subsegments: the flag bits and slope bits. The two bits in S1 represent the size of the matrix available, such as "10" for the 6×6 matrix. The number of slopes for each partition size is limited by the bits left over after removing the flag bits needed for the number of groups. For example, the 6×6 matrix has two bits available as slope bits (four slopes), when, if an additional auxiliary bit had been allocated, six different slopes could have been applied. In contrast, for a 5×7 matrix, the auxiliary bits permit three slope bits. Thus, the number of groups possible (five) in this matrix orientation limits the usable slopes from eight that could be encoded to five. Because the group partitioning encoded in S1 ranges from eight down to five, the number of available configurations is the sum of all the different configurations possible in each matrix size, which is $1 + 2 + 4 + 5$, or a total of 12. Traditional Aegis only has seven configurations.

Fig. 4 shows the encoding logic for a 32-bit data block to determine the encoding vector. The initial height bits (S1) and slope bits are applied to decoders in the ROM. For each stuck-at fault location (address), the ROM will reveal the group ID of the stuck-at fault. The primary goal is to locate a configuration, where stuck-at faults are all isolated into different groups with a secondary goal of isolating SA-R and SA-W faults into separate groups. If there is no collision of stuck-at faults within any group, to minimize write overheads, a new ID is only initiated when a new fault appears and creates a fault collision. If there is a collision of stuck-at faults within a group ID, that configuration does not satisfy the primary goal and the slope counter is incremented. If the slope surpasses the maximum available value at the data in S1 (see Fig. 3), the height counter is incremented and the slope is reset. If, during the search, an ID is found that satisfies the secondary goal of isolating SA-R and SA-W faults, we note it as a fall back option should the search for the primary goal fail.

In this manner, we search the possible data patterns in S1 and their correspondingly available slopes until we find a configuration where no collision occurs, which tolerates all the stuck-at faults in the block. If no such configuration is found, the fall back configuration that isolates SA-Ws and SA-Rs is used. The worst case encoding includes the traversal of all possible configurations, but in practice for lower error rates, this process occurs infrequently, and requires a very few

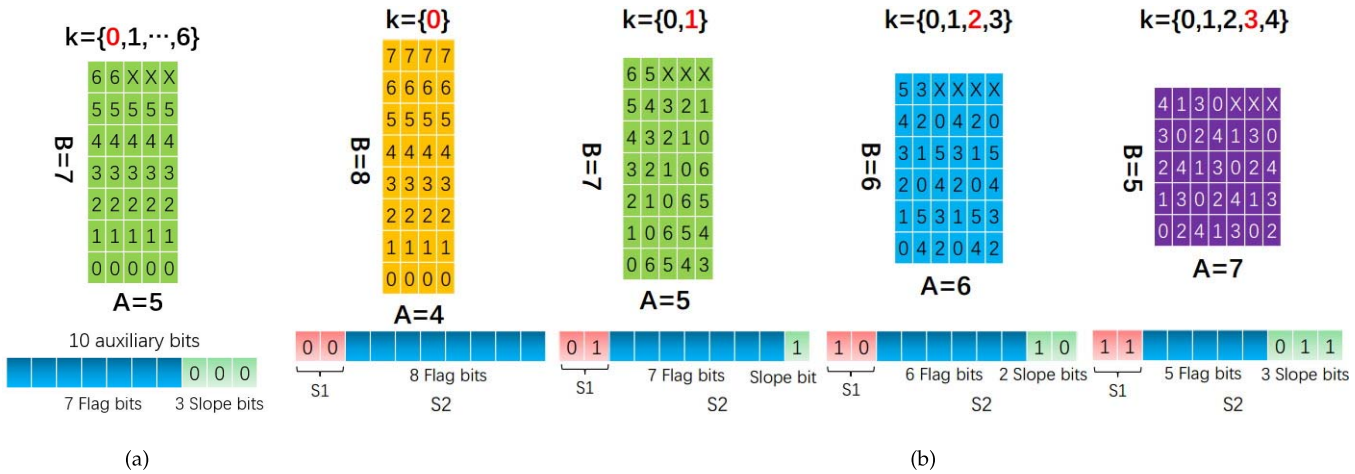

Fig. 3. (a) Static partitioning and (b) dynamic partitioning for a 32-bit block within Aegis.

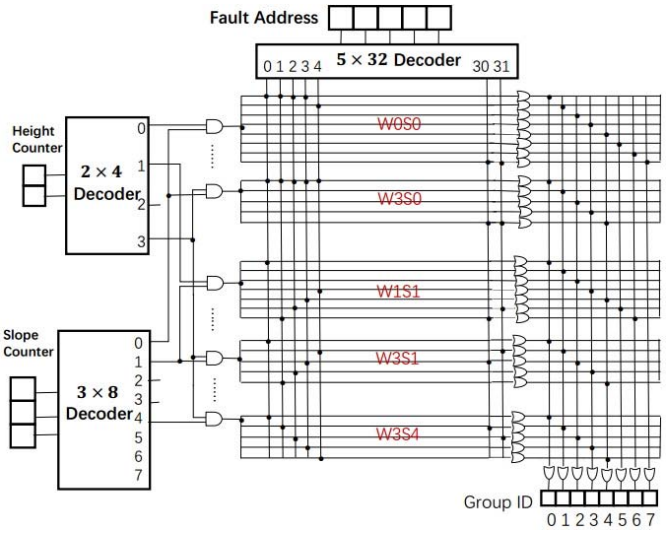

Fig. 4. Aegis$_{DY}$ logic to look up a group ID of a stuck-at fault for a specific height (S1) and slope, shown for a 32-bit block. In this example, a 49×32 and 49×7 ROM are used as lookup tables.

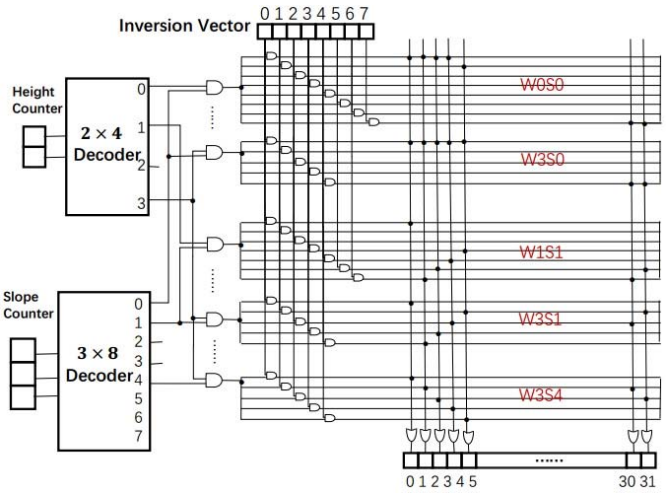

Fig. 5. Aegis$_{DY}$ logic to look up an inversion mask corresponding to the height, slope, and group inversion. A 49×32 ROM is used as a lookup table.

search iterations. As a performance versus storage tradeoff, an auxiliary bit can indicate a primary search failure and to enter a *gracefully degraded mode*, such that all future searches only isolate SA-R and SA-W faults.

In Fig. 5, we display the decoding logic to know which bits among a 32-bit data block were written in their inverted forms. Note that decoding is more efficient than encoding, because only a single lookup is required. In each implementation, we use a ROM-based lookup table to record the relationship between the input and the output information. The ROM capacity overhead is discussed in Section V.

IV. Relaxed Partitioning

The novelty of Aegis versus prior PAF schemes is that it can provide a stronger guarantee of fault tolerance. Aegis uses a unique partitioning such that each two bit pair in the same group of one configuration must be in different groups in any other configuration. This property allows Aegis to guarantee correction of a number of faults based on the number of groups in the partition. To ensure this unique partitioning, Aegis requires a prime number of groups in its partition, which is also the number of unique configurations that are recorded as slopes within its auxiliary bits.

However, the number of configurations that can be recorded by the slope bits in Aegis is always larger than the number of Aegis' configurations, meaning that Aegis always wastes several additional possible configurations enumerated by the slope bits. For example, consider a 512-bit data block, if the number of sets in a configuration is 23, Aegis uses five slope bits to record 23 configurations. However, these five bits can encode up to 32 configurations, so nine "extra" configurations could be recorded by these slope bits ($2^5 - 23 = 9$). If nine useful relaxed (e.g., nearly nonoverlapping) configurations can be determined, they can be encoded to fully leverage the flexibility of these slope bits. Furthermore, if the relaxed configurations are also efficient (nearly as efficient as the original Aegis configurations) at distributing faults into different groups, the fault tolerance could be dramatically improved by adding additional slope bits.

In this section, we propose a systematic method to generate relaxed configurations that guarantee that at most two bits in the same group of a relaxed configuration will be in the same group of an original Aegis configuration or another relaxed

configuration. Each relaxed configuration is also partitioned into a prime number of sets as Aegis. Thus, the relaxed configurations are compatible with Aegis configurations in hardware implementation by merely adding them to the ROMs.

A. Relaxed Partitioning Design

We propose to *relax* the property in Aegis that each partition configuration must have unique bit pairs. When we allow configurations that permit one redundant bit pair per group, the search space may be expanded with many useful configurations. To accomplish this, we utilize the same integer multiple of prime interleaving concept as Aegis, however, for each element in the group the interleaving increases by a constant value which we call the *slope gap*. The *slope gap* transforms the equation determining the vertical position for a group in each column in the rectangle representation of the data from linear to quadratic.

To define the "slopes" of a quadratic curve for partitioning, we use two variables: initial slope k and slope gap Δk. Conceptually, k and Δk are coefficients to linear and quadratic terms of the horizontal position used to describe the vertical position. In essence, Δk is the "acceleration" of the slope increase. To define the vertical positions b_i as a function of horizontal positions a_i for the ith element of a group with a starting point in column a_0 in the matrix, the corresponding "accumulated" slope is $(i-1)\Delta k$. To calculate the vertical position b_i of the ith column, we include the initial starting vertical position and compute the remainder over B to remain in the bounds of the matrix. Thus, the vertical offset of the column a_i can be computed as $(\Sigma_{j=0}^{i-1}(k + a_j\Delta k)) \bmod B$ which leads to

$$b_i = \left(a_i k + \frac{(a_i - 1)a_i\Delta k}{2}\right) \bmod B. \tag{1}$$

Equation (1) can be easily extended by a constant b offset y, where y refers to the yth group and $y \in [0, B)$. For each point, (a, b) in the matrix can be described by

$$b = \left(y + ak + \frac{(a-1)a\Delta k}{2}\right) \bmod B. \tag{2}$$

Fig. 6 shows an example of these relaxed partitioning configurations when the initial slope k is 0 or 1 and the slope gap Δk is 1. For illustration purposes, we arrange the n-bit data block into a matrix. The size of the matrix is $A \times B$, where B is the number of groups and A is the number of bits in each group. B is a prime number and $B \geq A$. The matrix has the same requirements as the matrix for static Aegis: given B, A is the smallest integer that meets the requirement that $A \times B \geq n$. As $B \geq A$ and $A \times B \geq n$, then it follows that $B \geq \sqrt{n}$. Each bit at position x in the data block is mapped to position (a,b) in the matrix, where $x = aA + b$. In Fig. 6(a), $k = 0$ zeros the first term of (1). Thus, for the zeroth group where $a_0 = 0$, according to the second term of (1), $b_0 = 0$. Continuing the example $a_1 = 1, b_1 = 0$, $a_2 = 2, b_2 = 1$, $a_3 = 3, b_3 = 3$, and $a_4 = 4, b_4 = 6$. Fig. 6(b) follows the example where both terms of (1) contribute to the vertical location.

The quadratic curve governed by (2) has the property that for any value Δk, no pair of bits will be in the same group

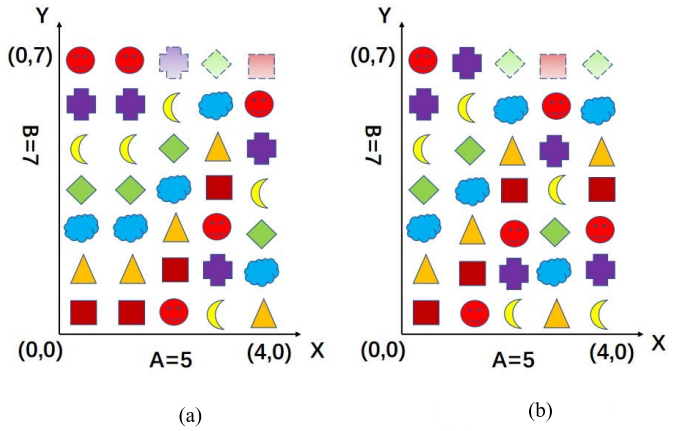

Fig. 6. Example of relaxed partitioning for a 32-bit block using 7×5 matrices with different initial slopes $k = 0, 1$ at a given slope gap $\Delta k = 1$. Similar to Fig. 1, each bit is represented by a symbol that represents to which group it belongs and symbols with dotted outlines do not correspond to an actual bit. (a) $K = 0$, $\Delta k = 1$, (b) $K = 1$, $\Delta k = 1$.

for any slope k. Moreover, when combined with traditional Aegis (relaxed partitioning, $\Delta k = 0$), the aggregate of both configurations guarantees that only one pair of bits from each partition overlaps with any other configuration. We demonstrate this property in the following theorems and proofs.

First, we demonstrate that the group offset y must be the same for all elements of the same group.

Theorem 1: For points (a, b) in an $A \times B$ matrix governed by (2), given an initial slope $k \in [0, B)$ and slope gap $\Delta k \in [0, B)$, there is a unique $y \in [0, B)$.

Proof: For a point to satisfy (2), it must also satisfy

$$y = mB + b - ak - \frac{(a-1)a\Delta k}{2} \tag{3}$$

which is the reorganization of (2) solved for y. This equation introduces the integer m from the mB term, derived from the mod$B$ term of (2). First, since a, b, k, and Δk are integers, then $c = b - ak - ((a-1)a\Delta k/2)$ is also an integer. Thus, we can rewrite (3) as $y = mB + c$.

Let us assume that for any point (a, b), there are two values, m_1 and m_2, where $m_1 \neq m_2$, with the corresponding values y_1 and y_2, respectively, where $y_1 \neq y_2$ that satisfy (3). In this case,

$$|y_1 - y_2| = |m_1 B + c - (m_2 B + c)| = |m_1 - m_2|B.$$

Since $|m_1 - m_2| \geq 1$, then $|y_1 - y_2| \geq B$. To guarantee that $y_1, y_2 \in [0, B)$, then $|y_1 - y_2| \in (0, B)$ must also hold. However, the assumption that $y_1 \neq y_2$ violates the requirement that $y_1, y_2 \in [0, B)$ meaning $y_1 = y_2$. □

According to Theorem 1, for each group's anchor point of the curve $(0, b)$, $b = y$, which is on the y-axis. For each given Δk, $\Delta k \in [0, B)$, quadratic partitioning generates a set of B configurations. In fact, the set of configurations generated by Aegis can be a special case of this expression, where $\Delta k = 0$. Next, we demonstrate that for quadratic partitioning with any particular Δk value, we generate nonoverlapping partitions.

Theorem 2: Given an $A \times B$ matrix, for each given $\Delta k \in [0, B)$, where $0 < A \leq B$ and B is prime, any two points in the same group in a configuration must not be in the same group again in another configuration.

Proof: Consider two points (a_α, b_α) and (a_β, b_β) that are in the same group of a configuration (defined by $k = k_0$), where b_α, b_β can be represented as a function of a_α, a_β using (2), where $0 \leq k_0, y < B$, and $0 \leq a_\alpha < a_\beta < B$. Let us assume that these two points are also in the same group in another configuration (defined by $k' \neq k_0$), which can be similarly represented using (2), where $0 \leq k', y' < B$.

From this assumption, we can equate the right-hand side of (2) for point a_α in both configurations yielding

$$(y + a_\alpha k) \bmod B = (y' + a_\alpha k') \bmod B \tag{4}$$

noting that $((a_\alpha - 1)a_\alpha \Delta k/2)$ is the same for both sides and has been canceled. Thus, we can reorganize (4) as follows:

$$y' - y = a_\alpha(k - k') + m_\alpha B \tag{5}$$

where as before m_α and m_β are integers. Similarly, we can generate the same equation for a_β which can be equated through $y' - y$ resulting in (6) and simplified to (7)

$$a_\alpha(k - k') + m_\alpha B = a_\beta(k - k') + m_\beta B \tag{6}$$

$$(a_\alpha - a_\beta)(k - k') = (m_\beta - m_\alpha)B. \tag{7}$$

As $a_\alpha, a_\beta \in [0, A)$, $k, k' \in [0, B)$, $a_\alpha \neq a_\beta$, and $k \neq k'$, then $|a_\alpha - a_\beta| \in (0, A)$ and $|k - k'| \in (0, B)$. Recalling that B is prime and neither $a_\alpha - a_\beta$ nor $k - k'$ can be a multiple of B, then (7) cannot hold, thus proving the theorem. □

For each $\Delta k \in [0, B)$, relaxed partitioning generates a set of B configurations, so this scheme can generate B sets of configurations. The sum of all these configurations is B^2.

Theorem 3: Among B^2 configurations generated by relaxed partitioning, for an $A \times B$ matrix, where $0 < A \leq B$ and B is prime, no more than two points in the same group in one configuration may be in the same group in another configuration.

Proof: Any three points (a_α, b_α), (a_β, b_β), and (a_γ, b_γ) in the same group of a configuration (defined by $k = k_0$ and $\Delta k = \Delta k_0$), where b_α, b_β, and b_γ can be represented as a function of a_α, a_β, and a_γ, respectively, using (2), where $0 \leq k_0, \Delta k_0, y < B$, and $0 \leq a_\alpha < a_\beta < a_\gamma < B$. Let us assume that these three points are also in the same group in another configuration (defined by k' and $\Delta k'$ such that $\Delta k' \neq \Delta k_0$), which can be similarly represented using (2), where $0 \leq k', \Delta k', y' < B$.

Similar to Eq. (5) but reintroducing the $((a_\alpha - 1)a_\alpha \Delta k/2)$ term which is now distinct between configurations we create the following expression:

$$y' - y = a_\alpha(k - k') + m_\alpha B + \frac{(\Delta k - \Delta k')(a_\alpha - 1)a_\alpha}{2}. \tag{8}$$

Similar to (7), we can generate the same equation for a_β which can be equated through $y' - y$ after simplification resulting in

$$(a_\alpha - a_\beta)\left[(k - k') + \frac{(\Delta k - \Delta k')(a_\alpha + a_\beta - 1)}{2}\right] = (m_\beta - m_\alpha)B. \tag{9}$$

If $a_\alpha, a_\beta \in [0, A)$, then $|a_\alpha - a_\beta| \in (0, A)$, and if $A \leq B$, then $|a_\alpha - a_\beta|$ cannot be a multiple of B. Because B is prime, then the remaining term in brackets must be a multiple of B for (9) to hold. Therefore, we define $h_{\alpha\beta}$ to be two times this integer multiple of B, which is also an integer multiple of B such that

$$2(k' - k) = (\Delta k - \Delta k')(a_\alpha + a_\beta - 1) - h_{\alpha\beta}B. \tag{10}$$

Equation (10) can be generated for points (a_α, b_α) and (a_γ, b_γ), and by setting them equivalent, we get the following:

$$(a_\beta - a_\gamma)(\Delta k - \Delta k') = (h_{\alpha\gamma} - h_{\alpha\beta})B. \tag{11}$$

We can obtain the same contradiction from (11) as from (6). As $a_\beta, a_\gamma \in [0, A)$, $\Delta k, \Delta k' \in [0, B)$, $a_\beta \neq a_\gamma$, and $\Delta k \neq \Delta k'$, then $|a_\beta - a_\gamma| \in (0, A)$ and $|\Delta k - \Delta k'| \in (0, B)$. Recalling that B is prime and neither $a_\beta - a_\gamma$ nor $\Delta k - \Delta k'$ can be a multiple of B, then (11) cannot hold, thus proving the theorem. □

The encoding and decoding logic for the relaxed partitioning scheme follows the same strategy as Aegis [15], but it requires larger ROMs in order to accommodate the additional configurations from both linear and quadratic curves. We describe the effectiveness of dynamic and relaxed partitioning and the experimental evaluation in Section V.

V. Evaluation

To validate our proposed dynamic partitioning and relaxed partitioning strategies, we first conduct a Monte Carlo simulation for high error rates. We compare relaxed partitioning Aegis_{RE}, dynamic versions FNW_{DY}, Aegis_{DY}, and $\text{Aegis}_{\text{DY,RE}}$ (dynamic partitioning combined with relaxed partitioning) against static counterparts of FNW and Aegis as well as baselines of ECP and ECC as well as other PAF strategies in the literature [14], [18]. Both data-oblivious partitioning and data-aware partitioning are studied. Data-oblivious partitioning attempts to segregate all stuck-at cells into different groups, while data-aware partitioning considers grouping together SA-R and SA-W cells based on the data to be stored. We then evaluate FNW_{DY}, Aegis_{DY}, Aegis_{RE}, and $\text{Aegis}_{\text{DY,RE}}$ in benchmark experiments where the auxiliary bits are also potentially faulty. These experiments consider the stuck-at fault rates of 10^{-3} and 10^{-4}. We conduct a similar set of experiments using probabilistic methods to further stress each memory location to include a sensitivity study with stuck-at faults rates from 10^{-3} to 10^{-6}.

A. Experimental Methodology

We developed a PIN-based simulator [19] and implemented ECC, ECP, FNW, FNW_{DY}, Aegis, Aegis_{DY}, Aegis_{RE}, and $\text{Aegis}_{\text{DY,RE}}$ to evaluate their tolerance to stuck-at faults. The PIN simulator evaluates the main memory writes by encoding or partitioning the data block and the auxiliary bits and recording a fault if the value of any fault bit being written is opposite to its stuck-at value. To model the stuck-at faults, a fault map, including fault bits stuck at '0' or '1,' is developed by using the Bayesian distribution to mimic the impact of process variation and includes spatial correlation of faults [20], [21]. For this paper, we followed the model described in [20] to generate maps of weak cells for a 4-GB PCM.

1) Model Implementation Details: In our evaluation, stuck-at faults can be tolerated in a data block for each of the data-aware schemes as follows.

1) For a Hamming code-based error correction (ECC-1_{64}), one SA-W and any number of SA-Rs can be tolerated in each data block or its parity bits.
2) For ECP$_f$, f SA-Ws can be tolerated in the data block, assuming that no SA-Ws compromise the auxiliary bits.
3) For FNW, there is no group that has both SA-Ws and SA-Rs in the group's data and its corresponding flag encoding) bit.
4) For FNW$_{\mathrm{DY}}$, there is a configuration that no group of the configuration has both SA-Ws and SA-Rs in the group data and its corresponding flag bit. The auxiliary bits (in S1) to record the group partitioning may not have any SA-Ws.
5) For Aegis and Aegis$_{\mathrm{RE}}$, there is a configuration that no group of the configuration has both SA-Ws and SA-Rs in the group data and its corresponding flag bit. The slope bits may not have any SA-Ws.
6) For Aegis$_{\mathrm{DY}}$ and Aegis$_{\mathrm{DY,RE}}$, there is a configuration that no group has both SA-Ws and SA-Rs in the group data and its corresponding flag bit. The auxiliary bits in S1 and the slope bits in S2 may not have any SA-Ws.

Each of the above-mentioned conditions can also be applied for data-oblivious by combining SA-R and SA-W into total faults and replacing any mentions of SA-W with "faults." The data block size examined in the study was 512 bits. For FNW, FNW$_{\mathrm{DY}}$, Aegis$_{\mathrm{DY}}$, Aegis$_{\mathrm{RE}}$, and Aegis$_{\mathrm{DY,RE}}$, we used 10, 15, 21, 28, 36 (33 for Aegis$_{\mathrm{RE}}$), and 43 (41 for Aegis$_{\mathrm{RE}}$) overhead bits per data block to tolerate stuck-at faults. These overhead ratios are 1.96%, 2.93%, 4.10%, 5.47%, 7.03% (6.45%), and 8.4% (8.01%). Aegis and Aegis$_{\mathrm{RE}}$ require a minimum of 23-bit encoding overhead for a 512-bit block to guarantee that each possible slope would have valid partitioning, as discussed in Section III-B. For fewer encoding bits, this correction cannot be guaranteed as the group size is not sufficiently large to guarantee independent slopes, however, it is still effective. For comparison, we relax this requirement and also allow Aegis to use 10, 15, and 21 auxiliary bits per data block, generated in the same manner [15].

For comparison, we provide the results for ECC-1_{64}, which requires 64 bits per data block, with an additional eight parity bits per word for 1-bit error correction and 2-bit error detection, corresponding to an overhead ratio of 12.5%.[2] ECP$_f$ requires $f \times 10 + 1$ bits per data block. We compare our proposed schemes with ECP$_f$, such that ECP requires the minimum auxiliary bits that exceed the auxiliary bits of our scheme (e.g., a 15-bit encoding would compare to ECP$_2$ that requires 21 bits).

First, we conduct Monte Carlo simulations of dynamic partitioning and relaxed partitioning to observe their differences in lifetime. To study the effect of aging, we assume that the lifetime of each cell in PCM follows a normal endurance failure distribution with a mean value of 10^8 writes with a 25% coefficient of variance. These simulations assume a 4-kB operating system page size with a data block size of 512 bits. Assuming an effective wear leveling method [22], [23], writes with random data are assumed to be uniformly distributed over the whole memory and differential write is adopted to reduce the frequency of cells written for each operation. Under these conditions, the simulations continuously issue page writes to a memory protected by various PAF schemes including dynamic and relaxed partitioning until there is an unrecoverable fault to determine their average lifetime improvement.

TABLE I
NUMBER OF CONFIGURATION OPTIONS

Aegis	Aegis$_{\mathrm{RE}}$	Aegis$_{\mathrm{DY}}$	Aegis$_{\mathrm{DY,RE}}$
$\min(2^s, B)$	$\min(2^s, B^2)$	$\sum_{i=0}^{D-1} \min(2^{s_i}, B_i)$	$\sum_{i=0}^{D-1} \min(2^{s_i}, B_i^2)$

In our subsequent experiments, we used our custom simulator to perform two kinds of evaluations on constant fault maps. First, we evaluate the memory accesses for the PARSEC benchmark suite [24] using our Pintool for different fault maps. The entire benchmark suite is executed, and an error rate is determined by accesses with uncorrectable errors compared with total accesses. This evaluation provides a snapshot of the correction capability of the PAF schemes for individual benchmarks across the lifetime of the memory. Second, we calculate what we call a "true random error rate," which we define as the error rate if a perfect distribution of all possible values was applied to each location of a fault map through an exhaustive search. This provides snapshots of the true correctness of the memory for all possible data patterns at different points throughout its lifetime. For each fault rate, we use the average error rates for five fault maps to evaluate and compare the effectiveness of the different fault recovery schemes.

2) Design Implementation: To guide the comparison of dynamic encoding and relaxed partitioning for Aegis, we implemented the encoding and decoding lookup tables (ROMs) for Aegis, Aegis$_{\mathrm{DY}}$, Aegis$_{\mathrm{RE}}$, and Aegis$_{\mathrm{DY,RE}}$ based on the number of configurations required, as shown in Table I. The hardware was implemented for a 45-nm CMOS by running the ROM lookup table designs in Synopsis Design Compiler targeting a 45-nm FreePDK [25].

The numbers of configuration options for Aegis and Aegis$_{\mathrm{RE}}$ are both a function of the number of slope bits s and the number of groups B. The numbers of configuration options for Aegis$_{\mathrm{DY}}$ and Aegis$_{\mathrm{DY,RE}}$ are a summation of options within all partitions, where D and B_i are the number of partitions and the number of groups for partition i, respectively.

Aegis$_{\mathrm{DY}}$, Aegis$_{\mathrm{RE}}$, and Aegis$_{\mathrm{DY,RE}}$ contain more configuration options for identical bit overhead and thus require additional lookup table area and delay. We provide a detailed study of the area and latency implications of the different dynamic and relaxed versions of Aegis in Appendix A of the Supplementary Material. The encoding latency reported is for evaluating one data partitioning option, while encoding for

[2]We also considered ECC-1_{256} to achieve a similar overhead (auxiliary bits) compared with Aegis and FNW. However, ECC performed so poorly it was more appropriate to compare with the more common (64, 72) ECC at higher overhead.

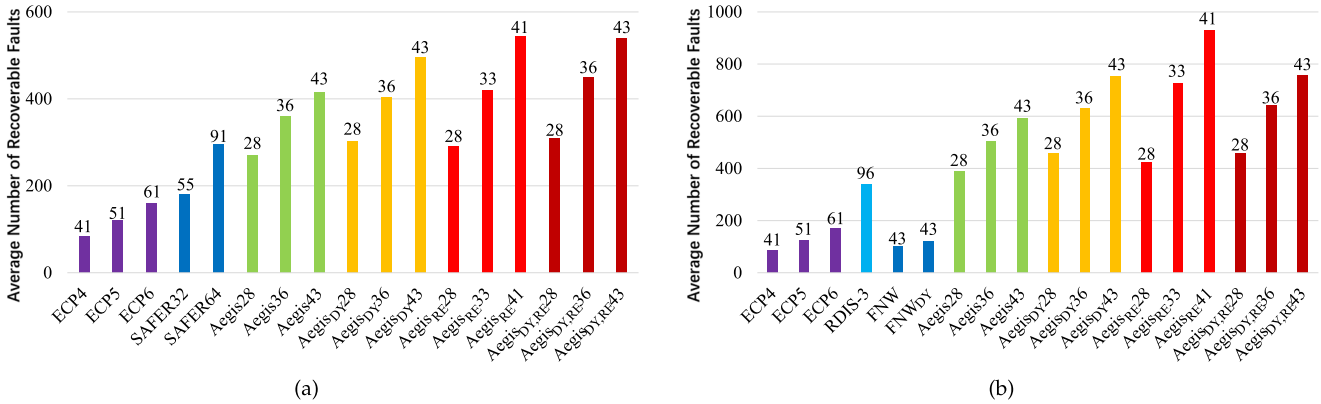

Fig. 7. Faults corrected before failure in a 4-kB page for various fault mitigation schemes. The number of auxiliary bits per block is shown above its bar. (a) Data oblivious. (b) Data aware.

Aegis, Aegis$_{DY}$, Aegis$_{RE}$, and Aegis$_{DY,RE}$ multiple encoding options may need to be explored to find a successful encoding which partitions SA-Rs and SA-Ws into separate groups. Moreover, this process requires evaluation with each stuck-at fault until an encoding is found which works for all stuck at faults in the data block. However, encoding is not on the critical path, and for lower error rates, it is rare to require many encoding attempts.

Table I in Appendix A of the supplementary material also enumerates the area and latency comparisons for decoding. For both 28 and 36 auxiliary bits, Aegis$_{DY}$ has a significant increase in latency over Aegis, while 15 and 21 auxiliary bits in Aegis$_{DY}$ have approximately the same delay as 28 and 36 bits for Aegis, respectively. At 28 auxiliary bits and below, Aegis$_{RE}$ tends to increase the ROM area more than Aegis$_{DY}$ as it typically adds fewer configurations by keeping the prime interleaving interval higher, similar to Aegis. It fills in additional configurations allowed by the auxiliary bits with a small number of mostly nonoverlapping partitions. Above 28 auxiliary bits, we can adjust the number of groups in a partitioning in order to adjust the tradeoff between a number of options and area/latency. In a later evaluation, we will demonstrate that even for these iso-performance comparisons, Aegis$_{DY}$ can achieve improved reliability compared with Aegis, while Aegis$_{RE}$ has a similar fault-correcting capability as Aegis at 28 auxiliary bits and below. As might be expected, Aegis$_{DY,RE}$ ROMs are considerably larger and require additional latency compared with Aegis$_{DY}$ or Aegis$_{RE}$, individually. However, Aegis$_{DY,RE}$ tends to provide $\geq 5\times$ the number of configuration options of Aegis$_{RE}$ for relatively small (typically ≤ 1 ns) increases in decoding and encoding latency.

B. Monte Carlo Simulations

Fig. 7 shows the average numbers of recovered faults in a 4-kB page by Aegis, Aegis$_{DY}$, and Aegis$_{RE}$ partitioning compared to ECP, SAFER, and RDIS-3, for data-oblivious [Fig. 7(a)] and data-aware [Fig. 7(b)] partitioning. For comparison purposes, we also show Aegis$_{DY,RE}$ which combines the concept of dynamic and relaxed partitioning into one approach. To ensure that the conclusions drawn from these results were not skewed by examining the mean of these particular fault maps studied in the experiment, we conducted a stabilization analysis of these trends. In particular, we examined these trends shown in the figure for Aegis, Aegis$_{DY}$, Aegis$_{RE}$, and Aegis$_{DY,RE}$ for different fault maps with different features such as a more even distribution of faults (easier to mitigate) and those with more clustered faults (harder to mitigate). What we found was that while the overall magnitude changed in terms of faults mitigated, the relative trends remained the same for each map and reflected the average trends reported. The detailed experiment is shown in Appendix B in the Supplementary Material.

From Fig. 7, we observe that the Aegis-based schemes achieve much higher effectiveness than the other schemes, including ECP, SAFER, and RDIS, with similar or even (often dramatically) lower capacity overheads. Both FNW and FNW$_{DY}$ are less effective than ECP with a similar overhead. Among Aegis, Aegis$_{DY}$, and Aegis$_{RE}$, Aegis$_{RE}$ outperforms the other schemes when there are ($\geq$33) auxiliary bits, followed by Aegis$_{DY}$ and Aegis in that order. For example, data-oblivious Aegis$_{DY}$ tolerated 12% and 19% more faults than its static counterpart with 36 and 43 auxiliary bits, respectively, while Aegis$_{RE}$ recovered an extra 4% and 10% faults over Aegis$_{DY}$ with two fewer auxiliary bits. Similarly, in Fig. 7(b), data-aware Aegis$_{DY}$ tolerated 25% and 27% more faults than static Aegis with 36 and 43 auxiliary bits, respectively, while Aegis$_{RE}$ recovered an extra 15% and 24% faults over Aegis$_{DY}$ with two less overhead bits. With 28 auxiliary bits, Aegis$_{DY}$ tolerates 4% more faults than Aegis$_{RE}$ and 8% more faults than Aegis. Aegis$_{RE}$ does not perform well in this instance, because the encoding bits do not provide an opportunity to record all the useful relaxed configurations without requiring additional overhead.

In Fig. 8, a similar trend in lifetime improvement can be observed although the gaps between different fault-tolerance schemes are much smaller. Data-aware Aegis$_{RE}$41 partitioning tolerated 10.8$\times$ more faults than ECP$_4$, while the lifetime improvement is only 79% higher than ECP$_4$. The reason is that, when approaching the end of the lifetime, cell failure rate increases exponentially, quickly exhausting the additional fault-tolerance capability. Nonetheless, data-aware Aegis$_{RE}$ still achieves a higher lifetime improvement with $\geq$33 auxiliary bits than the other data-aware schemes with a similar

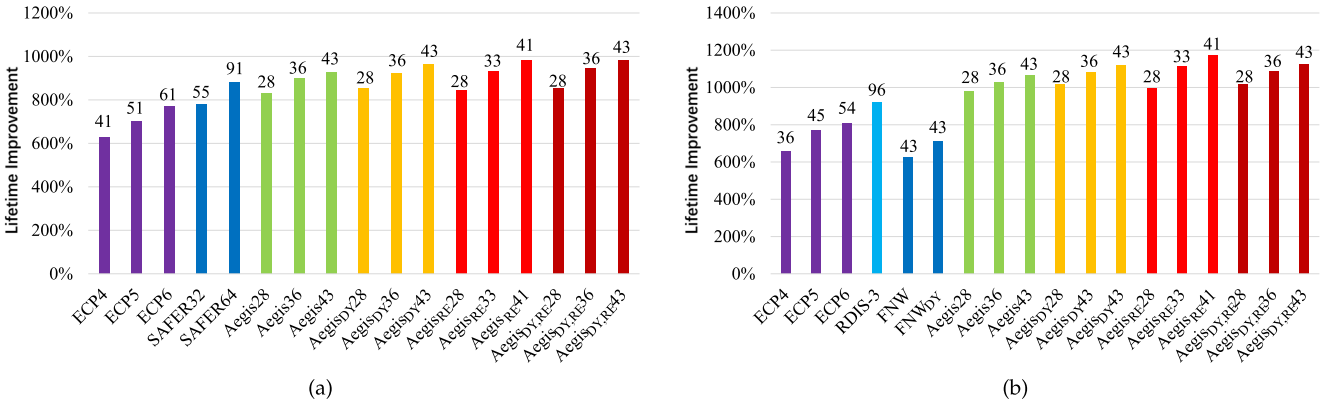

Fig. 8. Lifetime improvement (in terms of accesses) of a 4-kB page for various fault mitigation schemes compared with an unprotected page. The number of auxiliary bits per block is shown above its bar. (a) Data oblivious. (b) Data aware.

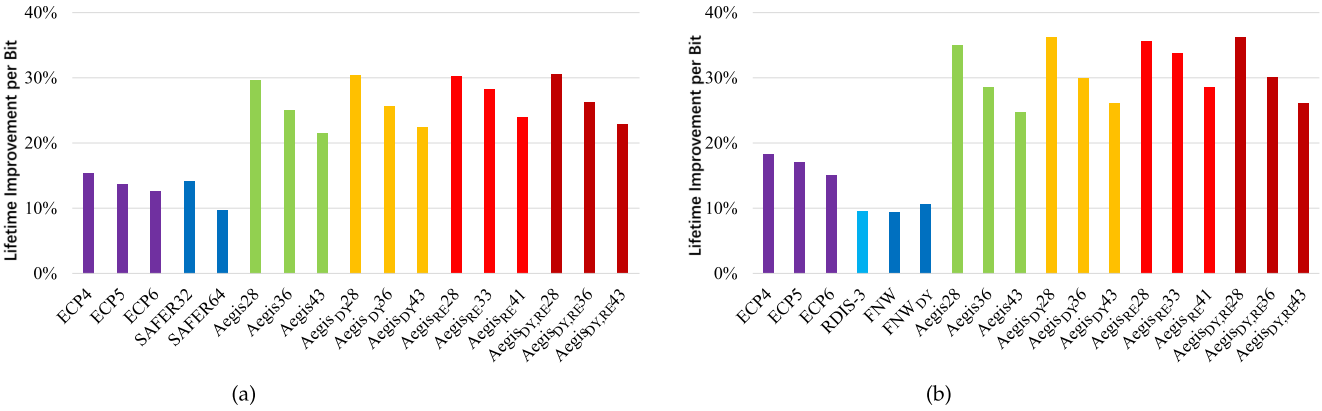

Fig. 9. Lifetime improvement per bit of various fault mitigation schemes. (a) Data oblivious. (b) Data aware.

overhead. For example, in Fig. 8(a), Aegis_{RE} improved the lifetime by 9.3× compared to an unprotected page, compared to only 9.0× and 9.2× improvements of Aegis and Aegis_{DY}, respectively, while using three fewer auxiliary bits. A similar trend is observed for data-aware partitioning [Fig. 8(b)].

The fault-tolerance per bit decreases as the number of auxiliary bits increases, as shown in Fig. 9. Data-aware Aegis_{RE} [Fig. 9(b)] provides a 34% lifetime improvement per bit when 33 auxiliary bits are used compared to 30% and 28% improvements for Aegis_{DY} and Aegis, respectively, when 36 auxiliary bits are used. A similar trend is observed for data-oblivious schemes [Fig. 9(a)].

To further compare the effectiveness of the fault-tolerance schemes, in Fig. 10, we show the probability of failure for the recovery schemes with different numbers of faults within a data block. For data-oblivious [Fig. 10(a)] before the number of faults reaches the guaranteed tolerance threshold, the probability of failure remains zero. ECP completely fails when its threshold is exceeded, while the other schemes gradually increase their probability of failure as the number of faults increases. With 28 auxiliary bits, the probabilities of failure of Aegis, Aegis_{DY}, and Aegis_{RE} almost overlap, and with 36 (or 33) auxiliary bits, Aegis_{DY} and Aegis_{RE} partitioning show an advantage over Aegis, and furthermore, with 43 (or 41) auxiliary bits, Aegis_{RE} exhibits a clear advantage. For data-aware trends [Fig. 10(b)], the advantage of Aegis_{RE} becomes more significant. For $\text{Aegis}_{\text{DY,RE}}$, the number of partitions in the data-oblivious case results in it surpassing the correction capability of Aegis_{RE}. However, in the data-aware case, Aegis_{RE} has more effective partitioning than $\text{Aegis}_{\text{DY,RE}}$ due to its guarantee of no more than two points in the same group in different configurations.

C. Benchmark Evaluation

In this section, we evaluate the effectiveness of the error mitigation strategies described in Section V-A1 for the PARSEC benchmark suite. We obtain the error rates for the initial stuck-at-fault rates of 10^{-3} and 10^{-4} shown in Figs. 11 and 12, respectively. In each figure, the fault mitigation schemes are compared to a baseline of ECC-1_{64} shown with a green line.

For the 10^{-3} initial stuck-at-fault rate (Fig. 11), all the schemes outperform ECC-1_{64} even though it requires the largest capacity overhead. While FNW does not dramatically improve over ECC-1_{64}, FNW_{DY} does provide improvements over *static* FNW for 10 and 15 auxiliary bit cases with larger improvement margin for 28 auxiliary bits, amounting to a 64% and 90% improvement, respectively. Recalling that (apart from 21 auxiliary bits), ECP_f uses slightly more auxiliary bits than the other schemes, ECP_f outperforms both ECC-1_{64} and FNW_{DY} but it is far inferior to Aegis, which achieves more than a 20× improvement over the next leading candidate. However, our dynamic partitioning, Aegis_{DY}, far outstrips Aegis. With 10 and 15 auxiliary bits, Aegis_{DY} achieves 5× and

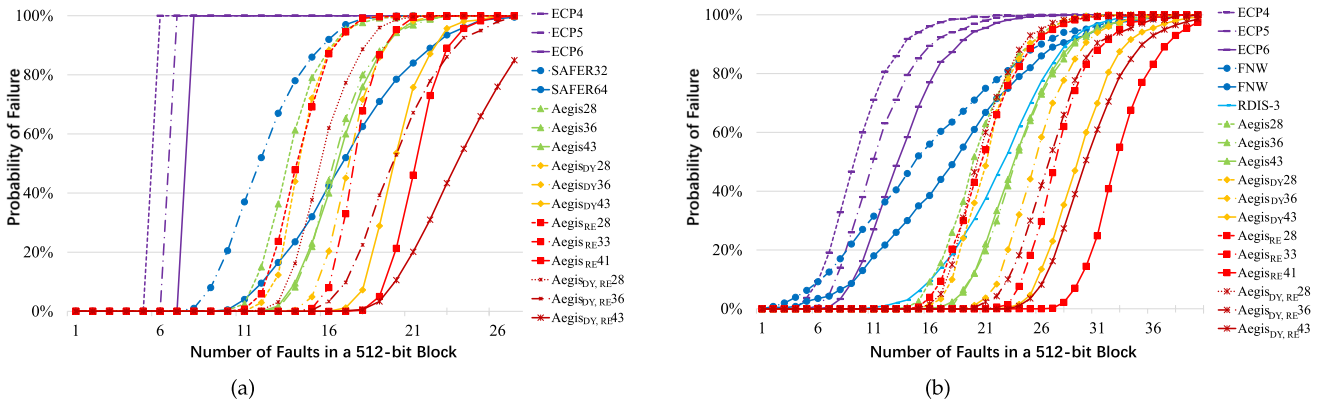

Fig. 10. Failure probability of a 512-bit data block with various numbers of faults under different fault-tolerance schemes. (a) Data oblivious. (b) Data aware.

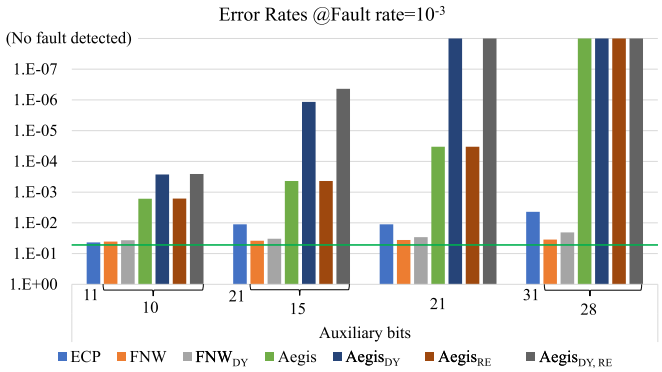

Fig. 11. Comparisons of different recovery schemes on the PARSEC benchmarks for a 10^{-3} fault rate.

Fig. 12. Comparisons of different recovery schemes on the PARSEC benchmarks for a 10^{-4} fault rate.

372× lower error rates than its static counterpart, respectively. When there are 21 auxiliary bits, Aegis_{DY} completely recovers from all the stuck-at faults in the fault maps, while static Aegis still has a higher than 10^{-5} error rate. They both achieve perfect protection when there are 28 auxiliary bits.

Interestingly, Aegis_{RE} performs similar to Aegis in these experiments, while $\text{Aegis}_{\text{DY,RE}}$ achieves notable improvements including a 1.7× improvement over Aegis_{DY} for 15 auxiliary bits. The advantage of Aegis_{DY} over Aegis_{RE} in these experiments appears to come from using a smaller initial prime partitioning that allowed for many more configurations to be explored. When the number of configurations is small, adding more configurations has a big impact for tolerance. When Aegis has large numbers of potential configurations, the dynamic configurations are less impactful. This is why for 15 auxiliary bits, $\text{Aegis}_{\text{DY,RE}}$ outperforms Aegis_{DY} as the former adds many quadratic configurations to supplement the dynamic configurations.

For an initial stuck-at-fault rate of 10^{-4} (Fig. 12), we see a similar trend as the 10^{-3} case, except that ECC-1_{64} is more effective than FNW, and FNW_{DY} prior to 28 auxiliary bits. Unsurprisingly, Aegis, Aegis_{DY}, $\text{Aegis}_{\text{DY,RE}}$, and Aegis_{RE} achieve perfect protection with fewer auxiliary bits, while only *static* Aegis and Aegis_{RE} with 10 auxiliary bits see any faults, but they still achieve an uncorrectable error rate of 10^{-5}. While for the same number of auxiliary bits, clearly Aegis_{DY} and $\text{Aegis}_{\text{DY,RE}}$ provide better protection than *static* Aegis and Aegis_{RE}, a method to distinguish lower uncorrectable error rates is necessary. We make this comparison with a probabilistic study in Section V-D.

D. Probabilistic Evaluation

In this evaluation, we simulate all possible data patterns in every row of the generated memory model and discover the "true error rates" of a perfectly even data distribution to distinguish lower error rates than exhibited through benchmark evaluation. Using true error rates, first, we compare the effectiveness of the error correction schemes with ISO auxiliary bits. Second, we compare the effectiveness of the *dynamic* schemes with their counterparts for an ISO performance comparison. The ISO performance comparison also provides an advantage as it requires a lower auxiliary bit storage overhead for Aegis_{DY} and $\text{Aegis}_{\text{DY,RE}}$.

1) ISO Auxiliary Bits Comparison: Figs. 13 and 14 show the comparisons of the various recovery schemes at the initial fault rates of 10^{-3} and 10^{-4}, respectively. FNW_{DY} protected 23% and 22% more data patterns than *static* FNW with 21 auxiliary bits at the two fault rates, while the improvements are 68% and 72% with 28 auxiliary bits, demonstrating the value of the dynamic partitioning strategy. For Aegis, dynamic partitioning is even more striking, with Aegis_{DY} obtaining 53× and 92× lower error rates at the two fault rates for 15 auxiliary bits. At the fault rate of 10^{-3}, Aegis_{DY} also

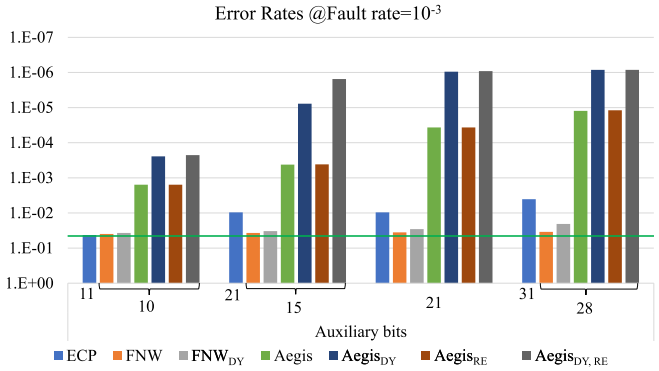

Fig. 13. Comparisons of different recovery schemes at a fault rate of 10^{-3} using ISO auxiliary bits.

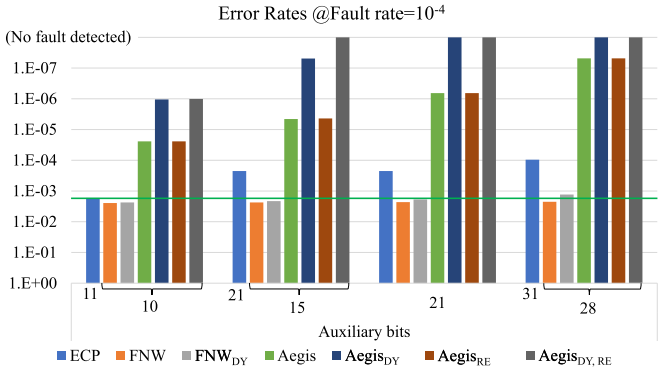

Fig. 14. Comparisons of different recovery schemes at a fault rate of 10^{-4} using ISO auxiliary bits.

TABLE II

COMPARISONS OF FNW_{DY} AND AEGIS_{DY} WITH THEIR STATIC COUNTERPARTS FOR 28 AUXILIARY BITS. "NONE" MEANS NO FAULT DETECTED FOR FIVE 256-MB FAULT MAPS

	FR	*Both*	*Dynamic*	*Static*	*Neither*
FNW	10^{-3}	0.96	$1.48{\cdot}10^{-2}$	$8.70{\cdot}10^{-4}$	$1.97{\cdot}10^{-2}$
	10^{-4}	0.998	$1.00{\cdot}10^{-3}$	$6.00{\cdot}10^{-5}$	$1.24{\cdot}10^{-3}$
Aegis	10^{-3}	0.999987	$1.17{\cdot}10^{-5}$	$7.16{\cdot}10^{-7}$	$6.76{\cdot}10^{-7}$
	10^{-4}	1-$4.83{\cdot}10^{-8}$	$4.83{\cdot}10^{-8}$	none	none

achieves 38× and 14× lower error rates over static Aegis with 21 and 28 auxiliary bits, respectively, while at the fault rate of 10^{-4}, Aegis_{DY} achieves perfect protection, but static Aegis still obtains an unrecoverable fault rate of 6.5×10^{-7} and 4.8×10^{-8}.

To illustrate the advantage of dynamic partitioning, we compare the error rates of the static and dynamic schemes in Table II. In Table II, FR is the initial fault rate, *both* refers to faults corrected by both the static and dynamic schemes, *dynamic* refers to faults only corrected by the dynamic scheme, *static* refers to faults corrected only by the static scheme, and *neither* refers to faults uncorrectable by either scheme. If the *dynamic* column is much larger than the *static* column, it illustrates the superiority of the dynamic scheme. For FNW, the numbers for FNW_{DY} are 21× larger than *static* FNW on average. For Aegis, Aegis_{DY} is 64× better than *static* Aegis at stuck-at fault 10^{-3}, while Aegis_{DY} achieves perfect protection, but Aegis still fails to protect a few data patterns at stuck-at fault 10^{-4}.

At the two fault rates for 15 auxiliary bits, $\text{Aegis}_{\text{DY,RE}}$ significantly outperforms Aegis_{DY}, while for the 10, 21, and 28 auxiliary bits, the gap is negligible. At the fault rate of 10^{-3}, by adding relaxed configurations, $\text{Aegis}_{\text{DY,RE}}$ achieves 4× lower error rate over the original Aegis_{DY}, while at the fault rate of 10^{-4}, $\text{Aegis}_{\text{DY,RE}}$ achieves perfect protection, but Aegis_{DY} still has an unrecoverable error rate of 4.9×10^{-8}.

With 10 auxiliary bits, $\text{Aegis}_{\text{DY,RE}}$ has only 3 configurations[3] more than that of Aegis_{DY} which has 12 configurations, while with 15 auxiliary bits, $\text{Aegis}_{\text{DY,RE}}$ has 83 configurations more than that of Aegis_{DY} which has 112 configurations. With 21 or more auxiliary bits, added relaxed configurations have reduced the impact, because they do not augment the protection of the auxiliary bits.

[3]These added relaxed configurations have a smaller number of groups, so that their effectiveness is comparably low.

TABLE III

IMPROVEMENT RATIO IN FAULT RATE OF AEGIS_{DY} OVER AEGIS FOR DIFFERENT AUXILIARY BITS AT AN INITIAL STUCK-AT RATE OF 10^{-3}. FOR EXAMPLE, 4.75 INDICATES THAT THE FAULT RATE OF AEGIS DIVIDED BY THE FAULT RATE OF AEGIS_{DY} IS 4.75 AT THE RESPECTIVE AUXILIARY BITS

	Aegis *Aux*			
Aegis_{DY} *Aux*	*10*	*15*	*21*	*28*
10	6.41	***1.71***	***0.15***	***0.05***
15	—	54.07	***4.75***	***1.60***
21	—	—	38.85	13.07
28	—	—	—	14.69

* Configurations within 10% decoding latency are in bold

2) ISO Performance Comparison: Due to the marked superiority of the dynamic strategy, Aegis_{DY} can achieve improved effectiveness over its static counterpart while maintaining performance (latency) with fewer auxiliary bits. In Table III, we show the improvement of Aegis_{DY} over Aegis at the initial fault rate of 10^{-3} with equivalent or reduced auxiliary bits. We mark configurations within 10% of decoding latency overhead (see Table I in Appendix A of the supplementary material) in bold and italics. For example, the error rate of Aegis_{DY} with 15 auxiliary bits is 1.60× lower as the rate of Aegis with 28 auxiliary bits at equivalent decoding latency. $\text{Aegis}_{\text{DY,RE}}$ and FNW_{DY} have a similar advantage over Aegis_{RE} and FNW, respectively.

E. Sensitivity Study for Lower Fault Rates

We further expand the range of the fault rate of the memory model to the lower initial fault rates of 10^{-5} and 10^{-6} in Figs. 15 and 16, respectively. At both fault rates, FNW fails to improve over ECC-1_{64}, only reaching equivalence to it for FNW_{DY} with 28 auxiliary bits, and performs worse than ECP_f as expected. Aegis and Aegis_{DY} continue to be dramatically more effective than the other schemes with at least two and three orders of magnitude improvement in uncorrectable error rates than ECP_f, respectively. At the fault rate 10^{-5}, Aegis_{DY} significantly outperforms static Aegis using 10 auxiliary bits

Fig. 15. Comparisons of different recovery schemes at a fault rate of 10^{-5} using ISO auxiliary bits.

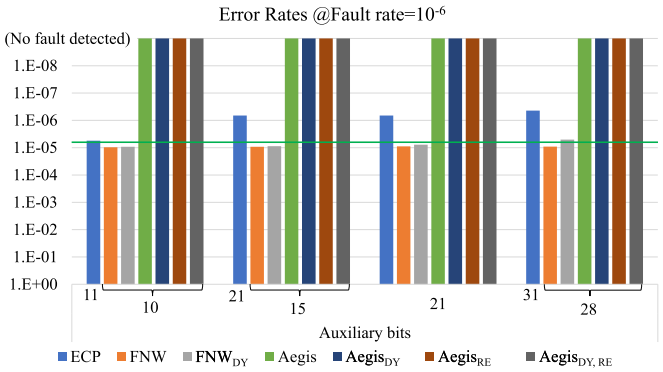

Fig. 16. Comparisons of different recovery schemes at a fault rate of 10^{-6} using ISO auxiliary bits.

with a 15× improvement. When there are at least 15 auxiliary bits, Aegis_{DY} achieves perfect protection, while Aegis requires 28 auxiliary bits to achieve the same goal. At the fault rate of 10^{-6}, Aegis and Aegis_{DY} protect all possible accesses, while the other schemes fail to achieve this goal. For both incident fault rates, Aegis_{RE} and $\text{Aegis}_{\text{DY,RE}}$ achieve similar error rates to static Aegis and Aegis_{DY}, respectively, resulting in similar comparisons between static and dynamic versions.

F. Discussion

Aegis_{DY} and Aegis_{RE} do not increase the number of guaranteed faults that can be corrected over Aegis, but they do increase the practical fault tolerance. Aegis_{DY} provides a significant tangible benefit over Aegis due to its use of many more possible partitions compared with Aegis. However, Aegis_{RE} provides more potentially valuable configurations within a particular partition, which has different benefits than Aegis_{DY}. The results from Fig. 7 show that while Aegis_{RE} does not increase the guarantee over Aegis, it pushes the probability of failure off significantly, particularly for higher numbers of auxiliary bits. This is echoed in Fig. 10. This impact can also be seen in Figs. 11, 13, and 14, where for 15 auxiliary bits, using relaxed configurations helps Aegis_{DY} due to the higher uniqueness of the added configurations. Because Aegis_{RE} contains all of the original Aegis configurations in addition to the relaxed configurations, the original configurations contain the same uniform diffusion of faults present in Aegis while benefiting from the new unique partitioning options. Thus, there is benefit from adding more partitions with different sized groups (Aegis_{DY}) on top of the fault correction guarantee, as well as benefit from adding *useful* configurations within a partition (Aegis_{RE}), and these improvements can work together.

Given the overheads from the size of the encoding/decoding ROMs and latencies shown in Table I in Appendix A of the supplementary material, the choice of encoding scheme from classic Aegis to $\text{Aegis}_{\text{DY,RE}}$ also provides an interesting tradeoff for the system designer. In the previously discussed practical range of memory overhead (e.g., 15 auxiliary bits), $\text{Aegis}_{\text{DY,RE}}$ dramatically improves error rates but increases the ROM size and latency. This creates a choice between a smaller memory overhead (fewer auxiliary bits) with a larger overhead in the memory controller (larger slower ROM) compared with a higher memory overhead (larger numbers of auxiliary bits). Thus, the former scenario employing $\text{Aegis}_{\text{DY,RE}}$ in order to achieve an improvement in fault tolerance and system lifetime becomes an attractive choice for the full system design.

VI. Conclusion

Endurance limitations are a significant challenge for mass commercialization of several emerging nonvolatile memories, including PCM and RRAM. This is especially problematic when more cells become potentially faulty due to technology scaling and resulting process variation. We presented our proposed dynamic partitioning approach to mitigate stuck-at faults in these emerging memories. Dynamic partitioning recovers more stuck-at faults in a data block by increasing the number of possible partitions, thus improving over static PAF schemes.

Furthermore, we also presented our relaxed partitioning scheme that provides more cost-effective configurations with a limited partitioning overlap. Our results, including Monte Carlo simulations, benchmark analyses, and probabilistic evaluations, show that dynamic partitioning significantly improves the effectiveness of FNW and Aegis over their static counterparts by generating more efficient configurations. The results also illustrate that relaxed partitioning can further improve the memory lifetime over dynamic Aegis with equal or reduced space overhead.

As there are some relatively infrequent cases where the static scheme can correct faults uncorrectable in our dynamic approach, it is possible to incorporate a static scheme into our dynamic scheme by using one extra bit as a record in each data block. Our evaluation shows that the extra tolerance provided by the original static scheme is negligible compared to the uncorrectable error rate. However, this could be explored further in the future work.

References

[1] J. Zhang, D. Kline, Jr., L. Fang, R. Melhem, and A. K. Jones, "Dynamic partitioning to mitigate stuck-at faults in emerging memories," in *Proc. ICCAD*, Nov. 2017, pp. 651–658.
[2] K. Kim, "Technology for sub-50nm DRAM and NAND flash manufacturing," in *IEDM Tech. Dig.*, Dec. 2005, pp. 323–326.
[3] S. Li, P. Wang, N. Xiao, G. Sun, and F. Liu, "SPMS: Strand based persistent memory system," in *Proc. DATE*, Mar. 2017, pp. 622–625.
[4] O. Zilberberg, S. Weiss, and S. Toledo, "Phase-change memory: An architectural perspective," *ACM Comput. Surv.*, vol. 45, p. 29, Jun. 2013.

[5] J. Zhang *et al.*, "A generalized model of TiO_x-based memristive devices and its application for image processing," *Chin. Phys. B*, vol. 26, pp. 70–81, Aug. 2017.
[6] C. J. Xue, G. Sun, Y. Zhang, J. J. Yang, Y. Chen, and H. Li, "Emerging non-volatile memories: Opportunities and challenges," in *Proc. CODES+ISSS*, Oct. 2011, pp. 325–334.
[7] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[8] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Coset coding to extend the lifetime of memory," in *Proc. HPCA*, Feb. 2013, pp. 222–233.
[9] S. M. Seyedzadeh, R. Maddah, D. Kline, A. K. Jones, and R. Melhem, "Improving bit flip reduction for biased and random data," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3345–3356, Nov. 2016.
[10] Z. Wu and S. Chen, "nMOS transistor location adjustment for n-hit single-event transient mitigation in 65-nm CMOS bulk technology," *IEEE Trans. Nucl. Sci.*, vol. 65, no. 1, pp. 418–425, Jan. 2018.
[11] Z. Wu, S. Chen, J. Yu, J. Chen, P. Huang, and R. Song, "Recoil-ion-induced single event upsets in nanometer CMOS SRAM under low-energy proton radiation," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 654–664, Jan. 2017.
[12] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," *ACM SIGARCH Comput. Archit.*, vol. 38, no. 3, pp. 141–152, Jun. 2010.
[13] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. MICRO*, Dec. 2009, pp. 347–357.
[14] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "Safer: Stuck-at-fault error recovery for memories," in *Proc. MICRO*, Dec. 2010, pp. 115–124.
[15] J. Fan, S. Jiang, J. Shu, Y. Zhang, and W. Zhen, "Aegis: Partitioning data block for efficient recovery of stuck-at-faults in phase change memory," in *Proc. MICRO*, Dec. 2013, pp. 433–444.
[16] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "ArchShield: Architectural framework for assisting DRAM scaling by tolerating high error rates," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 72–83, 2013.
[17] J. Zhang, D. Kline, L. Fang, R. Melhem, and A. K. Jones, "Yoda: Judge me by my size, do you?" in *Proc. ICCD*, Nov. 2017, pp. 395–398.
[18] R. Melhem, R. Maddah, and S. Cho, "RDIS: A recursively defined invertible set scheme to tolerate multiple stuck-at faults in resistive memory," in *Proc. DSN*, Jun. 2012, pp. 1–12.
[19] C.-K. Luk *et al.*, "Pin: Building customized program analysis tools with dynamic instrumentation," *ACM SIGPLAN Notices*, vol. 40, no. 6, pp. 190–200, 2005.
[20] Z. Al-Ars, "DRAM fault analysis and test generation," Doctoral thesis, Delft Univ. Technol., Delft, The Netherlands, 2005.
[21] T. Yuan, S. Z. Ramadan, and S. J. Bae, "Yield prediction for integrated circuits manufacturing through hierarchical Bayesian modeling of spatial defects," *IEEE Trans. Rel.*, vol. 60, no. 4, pp. 729–741, Dec. 2011.
[22] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. MICRO*, vol. 14, Dec. 2009, pp. 14–23.
[23] J. Zhang, D. Kline, Jr., L. Fang, R. Melhem, and A. K. Jones, "RETROFIT: Fault-aware wear leveling," *IEEE Comput. Archit. Lett.*, vol. 17, no. 2, pp. 167–170, Jul./Dec. 2018.
[24] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. PACT*, Oct. 2008, pp. 72–81.
[25] J. E. Stine *et al.*, "FreePDK: An open-source variation-aware design kit," in *Proc. MSE*, Jun. 2007, pp. 173–174.

Jiangwei Zhang (S'17) received the M.E. degree in electronic science and technology from the National University of Defense Technology, Changsha, China, in 2014, where he is currently working toward the Ph.D. degree.

From 2016 to 2018, he was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA. His current research interests include memory reliability, emerging memories, and image processing.

Donald Kline, Jr. (S'13) received the B.S. degree in computer engineering and the M.S. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2015 and 2017, respectively, where he is currently working toward the Ph.D. degree in electrical and computer engineering under the guidance of Dr. A. Jones and Dr. R. Melhem.

He is currently an NSF Graduate Research Fellow with the University of Pittsburgh. His current research interests include computer architecture, emerging memories, reliability, compilers, and machine learning.

Liang Fang received the Ph.D. degree in computer science from the National University of Defense Technology, Changsha, China, in 1995.

From 2005 to 2006, he was a Visiting Professor with the Max-Plunck Institute of Microstructure Physics, Halle, Germany. He is currently a Professor with the School of Computer, National University of Defense Technology. His research interests include microelectronics devices, high-performance microprocessors, single-electron transistors, graphene transistor, carbon nanotube computers, resistive switching memory, and quantum computing.

Rami Melhem (F'00) received the B.E. degree in electrical engineering from Cairo University, Giza, Egypt, in 1976, and the M.A. degree in mathematics and the M.S. degree in computer science and the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 1981 and 1983, respectively.

He is currently a Professor and the Past Chair (2000–2009) of the Computer Science Department at the University of Pittsburgh. Formerly, he was an Assistant Professor with Purdue University, West Lafayette, IN, USA. His current research interests include power management, parallel computer architectures, real-time and fault-tolerant systems, optical networks, and high performance.

Dr. Melhem is a member of the ACM. He served and is serving on program committees of numerous conferences and Editorial Boards of several journals, including the IEEE TRANSACTIONS ON COMPUTERS and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.

Alex K. Jones (SM'08) received the B.S. degree in physics from the College of William and Mary, Williamsburg, VA, USA, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 2000 and 2002, respectively.

He is a Sustainability Faculty Fellow, and Professor of electrical and computer engineering and computer science (by courtesy) with the University of Pittsburgh, Pittsburgh, PA, USA. His current research interests include compilation techniques for configurable systems and architectures, sustainable computing, and fault-tolerant computing. He has authored over 150 publications in these areas. His research is funded by the U.S. National Science Foundation, DARPA, and industry.