Exploiting Asymmetry in eDRAM Errors for Redundancy-Free Error-Tolerant Design

Shanshan Liu (Member, IEEE), Pedro Reviriego (Senior Member, IEEE), Jing Guo, Jie Han (Senior Member, IEEE) and Fabrizio Lombardi (Fellow, IEEE)

Abstract—For some applications, errors have a different impact on data and memory systems depending on whether they change a zero to a one or the other way around; for an unsigned integer, a one to zero (or zero to one) error reduces (or increases) the value. For some memories, errors are also asymmetric; for example, in a DRAM, retention failures discharge the storage cell. The tolerance of such asymmetric errors would result in a robust and efficient system design. Error Control Codes (ECCs) are one common technique for memory protection against these errors by introducing some redundancy in memory cells. In this paper, the asymmetry in the errors in Embedded DRAMs (eDRAMs) is exploited for error-tolerant designs without using any ECC or parity, which are redundancy-free in terms of memory cells. A model for the impact of retention errors and refresh time of eDRAMs on the False Positive rate or False Negative rate of some eDRAM applications is proposed and analyzed. Bloom Filters (BFs) and read-only or write-through caches implemented in eDRAMs are considered as the first case studies for this model. For BFs, their tolerance to some zero to one errors (but not one to zero errors) is combined with the asymmetry of retention errors in eDRAMs to show that no ECC or parity is needed to protect the filter; moreover, the eDRAM refresh time can significantly be increased, thus reducing its power consumption. For caches, this paper shows that asymmetry in errors can be exploited also by using a redundancy-free error-tolerant scheme, which only introduces false negatives, but no false positives, therefore causing no data corruption. The proposed redundancy-free implementations have been compared with existing schemes for BFs and caches to show the benefits in terms of different figures of merit such as memory size, area, decoder/encoder complexity and delay. Finally, in the last case study, we show that the asymmetry of retention errors can be used to develop additional error correction capabilities in Modular Redundancy Schemes.

Index Terms— Asymmetric errors, memory design, eDRAMs, Bloom filters, caches, error tolerance

1 Introduction

Memories play a significant role in integrated circuit design. Volatile memories such as Static Random-Access Memories (SRAMs) and Dynamic Random-Access Memories (DRAMs) [1] are widely used in computing and networking applications. SRAMs offer high-speed storage, but they need at least six transistors, so integration density is modest. They are usually used as first level caches in CPUs. While DRAM cells are very small (requiring one transistor and a capacitive element), so they have a high density that makes them attractive to be employed for larger caches or main memory [2],[3]. In particular, embedded DRAMs (eDRAMs) are widely used to implement on chip memories [4]. Few novel DRAM cache structures can potentially have the same (or better) speed than SRAM caches [5]. However, eDRAMs need frequent refresh operations to retain the stored value; otherwise, the value can be changed due to leakage current and crosstalk in the circuit, causing the so-called retention errors. Such retention errors occur mostly on the charged cells, so they are highly asymmetric (if a "1" ("0") is stored in the cell then it will be flipped to "0" ("1")). The retention times are theoretically in the order of milliseconds, but in practice, they are in the order of a few tens of microseconds due to a few cells discharging faster [6]. Therefore, eDRAMs incur in a significant power consumption to perform periodic refresh operations. Memories are also prone to other soft errors, such as radiation induced Single Event Upsets (SEUs); SEUs can modify the contents of a word, causing data corruption and affecting system integrity. Radiation induced soft errors in SRAMs are usually symmetric (i.e., flipping a stored "1" to "0" or "0" to "1") [7], but in eDRAMs this depends on the part of the circuit that is been affected by the SEUs [5], [7]-[10]. If SEUs occur on the memory cell (the transistor and capacitor pair), soft errors are asymmetric, because the free carriers generated at or near the drain of the n-channel access transistor due to radiation are collected across the drain/substrate junction, thereby discharging the capacitor. Therefore, they exhibit the same behavior for state changing as retention errors. However, if SEUs occur on other parts of the eDRAM circuit (for example sense amplifiers), then the errors can be symmetric.

Error Control Codes (also referred to as Error Correction Codes, ECCs) have been extensively used in memory applications to deal with different classes of errors such as those introduced above [11]-[13]. To protect a memory, an ECC first needs few parity bits (Figure 1); these parity bits

- *Manuscript received 26 June, revised XX October, 2019. (Corresponding author: Shanshan Liu)*

- *S. Liu and F. Lombardi are with Northeastern University, Dept. of ECE, Boston, MA 02115, USA (email: ssliu@coe.neu.edu, lombardi@ece.neu.edu)*
- *P. Reviriego is with Universidad Carlos III de Madrid, Av. Universidad 30, Leganés, Madrid, Spain (email: revirieg@it.uc3m.es)*
- *J. Guo is with School of Instrument and Electronics, North University of China, Taiyuan, 030051, China (email: guojing19861229@163.com).*
- *J. Han is with University of Alberta, Dept. ECE, Edmonton, Alberta T6G 2V4, Canada (email: jhan8@ualberta.ca)*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2019.2960491, IEEE Transactions on Emerging Topics in Computing

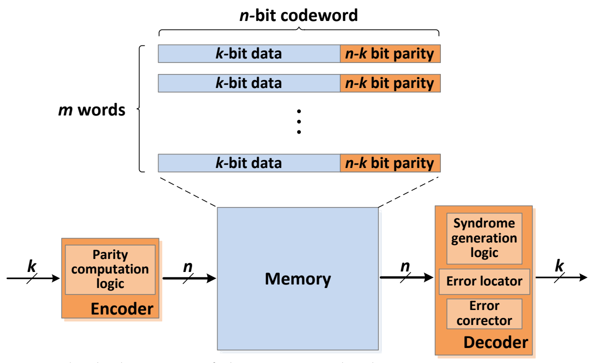

Figure 1 Block diagram of the proposed scheme

are computed based on memory data and performed by the encoder prior to the memory write operation. Then, the parity and the data bits are stored together in each memory word during the write operation. After a read operation, a decoding procedure is performed; the parity bits are first recomputed based on the stored data and checked with the stored parity bits to generate the syndrome. As per the decoding algorithm, the syndrome can be used to locate the error and then correct it (if correctable). The correct data is then provided as output by the decoder. In addition to improving the reliability of memories, ECCs used in eDRAM can also improve the refresh rate by correcting retention errors, thus reducing power consumption. For example, the use of SEC-DED codes (that have capabilities of single-bit error correction and double-bit error detection), or stronger 5EC-6ED BCH codes (that can correct 5-bit errors and detect 6-bit errors) have been considered in eDRAMs [13],[14]. However, the drawback of ECCs is that the introduced redundant cells and the circuitry for the encoder/decoder increase the memory size and affects the read/write latency. Especially for strong ECC functions, the overhead introduced could be rather large and not acceptable for some designs.

In some applications, errors have a different impact if they change a “1” to “0” or a “0” to “1”. For example, when an unsigned integer is stored in a register, the “1” to “0” errors reduce the value while the “0” to “1” errors would increase it. If this value is related to the remaining time to perform a refresh operation in eDRAMs, a decrement of the value would cause a higher refresh frequency so incurring in a larger power consumption. However, an increase in the value would delay the refresh and thus may lead to data corruption. In a Bloom Filter (BF) (as commonly used in networking applications), a small percentage of false positives (matching occurs in a pair of mismatched data) is inherent to the filter nature; no false negative (mismatching occurs in a pair of matched data) can occur in an error free filter. Therefore, errors from “1” to “0” cannot be tolerated; however, errors from “0” to “1” would only increase the false positive rate. As long as the “0” to “1” errors are limited in number, their impact on the reliability of BFs is negligible. This is also the case in counting BFs or count-min sketch applications [15]-[17]. Another example can be read-only caches, in which false positives can cause error propagation, but false negatives are not an issue, because they cannot lead to data corruption.

As error-tolerant techniques for memories are usually costly, it is interesting to exploit the asymmetry of errors in specific memories to implement efficient protection. In this paper, we propose redundancy-free (in terms of memory cells) schemes that can deal with asymmetric errors in memories for applications that tolerate some false positives or negatives. Bloom Filters and caches are used as case studies in this paper to illustrate the proposed strategy. By combining the asymmetry of errors in memories and error-tolerance, an effective protection is designed by introducing only additional logic, but no ECC or parity. This significantly reduces the hardware overhead compared to the use of existing protection techniques. Finally, the application of the proposed scheme to modular redundant systems is also discussed.

2 DRAM Background

2.1 DRAM Architecture

As illustrated in Figure 2, a DRAM module is arranged in a hierarchical manner. Each DRAM module consists of a set of memory banks (e.g., eight) that include independent memory arrays; each two dimensional array consists of cells, row and column decoders, sense amplifiers, and I/O gating. Each memory cell includes a capacitor and an access transistor that connects the associated capacitor to a bitline.

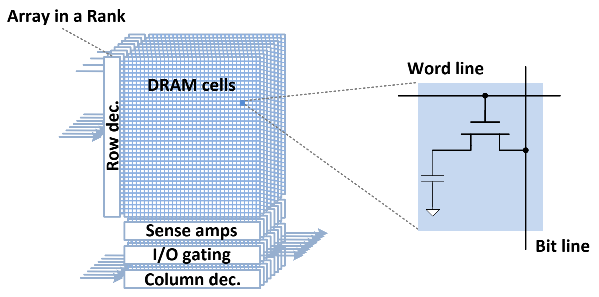

Figure 2 DRAM architecture

In an eDRAM cell, data is stored as an electrical charge in a capacitor, i.e., a binary bit “1” or “0” is represented by the amount of stored charge. Depending on circuit design, a true-cell and anti-cell refer to those eDRAM cells that present “1” with the charged and discharged states (i.e., the stored values are inverted) respectively [18]. eDRAM chips can be implemented by only using true-cells, or anti-cells, or both. In this paper, true-cell implementations are assumed; for anti-cell eDRAMs, an inverted coding can be simply removed or included in the proposed schemes.

When an eDRAM is in stand-by mode, the charged capacitor in a cell discharges over time due to leakage currents in the circuitry and eventually it may lead to an error. So eDRAMs require refresh operations to ensure data retention. The read operation in a DRAM is achieved by checking the flow direction of the current path (the in-flow is considered as a logic “0”, while the out-flow is considered as logic “1”). This may also damage the original state of the cell, so writing the data back into the cell is needed after a read operation.

2.2 Asymmetry of Retention Errors

Considering that a logic "1" is represented by a charged cell (true-cell), leakage currents in the cell may mostly cause "1" to "0" errors but few "0" to "1" errors, therefore eDRAM retention errors are highly asymmetric [14],[19]. When eDRAM cells follow the tail (main) distribution, the unidirectional gate-induced drain leakage current (junction leakage current) constitutes the dominant leakage mechanism for the weakest eDRAM cells; it discharges the cells and can cause asymmetric errors.

In addition to cell leakage, crosstalk may also cause a retention failure. There are four major sources of crosstalk in eDRAMs, bitline to bitline coupling, wordline to wordline coupling, wordline to bitline coupling, and bitline to cell coupling [18]. When the charge stored in an eDRAM cell is weak due to cell leakage, crosstalk can flip the degraded cell charge on the bitline during the sensing process, causing symmetric retention errors. However, when the charge in the cell is strong (for short refresh intervals), or significantly degraded (for long refresh intervals), crosstalk has little impact on the sensing process. Experimental results [18] have shown the highly asymmetric property of retention errors. Even if some "0" to "1" errors (caused by sub-threshold leakage and crosstalk) are observed, their number is very small and thus, negligible in most cases. The asymmetry of retention errors is also the underlying principle in many published works [19]-[21].

In a traditional design, the refresh time of the eDRAM is driven by a small percentage of cells that discharge fast. This leads in practice to refresh times of a few tens of microseconds and thus a significant power consumption is incurred when performing refresh. This is in stark contrast with theory in which the refresh times are expected to be in the order of milliseconds for most of the cells [6].

3 Retention Error Model of eDRAMs

As discussed in a previous section, a storage capacitor loses charge over time due to leakage currents, so refresh operations are needed in DRAMs to ensure data retention. The retention time of a DRAM cell is defined as the time for which the cell can retain its correct state, and it depends on the leakage currents. Therefore, the refresh period of a DRAM should be shorter than the cell retention time to avoid data corruption. Embedded DRAMs mostly use fast logic transistors with a higher leakage current than conventional DRAMs; therefore, they have a significantly shorter refresh time (about 1000 times shorter than DRAMs).

Retention errors depend on the cells that discharge faster, so they are related to the distribution of the retention time among eDRAM cells. Previous works have analyzed the relationship between the retention time of a cell and the probability of a fault-free eDRAM or single bit failure [13],[22]. Figure 3 shows the probability distribution of single bit failures with refresh times of an eDRAM [13]. Figure 3 shows that the probability of a single bit failure increases with a longer refresh time of the eDRAM. The plot of Figure 3 can be fitted as approximation model by equation (1):

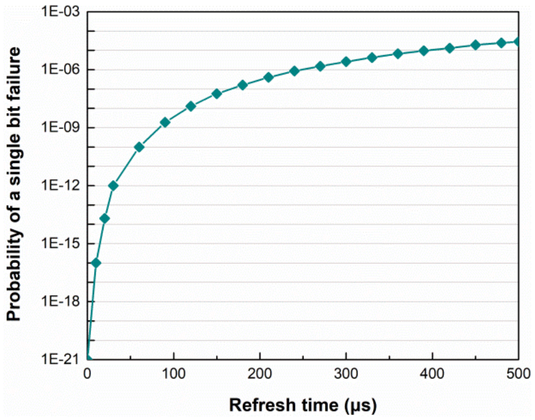

Figure 3 eDRAM retention time distribution [13]

$$p_{ret} \cong \begin{cases} 1.27\times10^{-17}\cdot t^{4.59} & t \le 4798\ \mu s \\ 1 & t > 4798\ \mu s \end{cases} \quad (1)$$

where p_{ret} is the probability of a single bit failure caused by a retention error (where $p_{\text{ret}} = p_{\text{ret\_c}} + p_{\text{ret\_d}}$), t is the refresh time of the eDRAM. An eDRAM with a refresh time in the order of a few tens of microseconds has an extremely low probability of failure.

Consider $p_{\text{ret\_c}}$ as the probability of a bit "1" changing to "0" in the charged cell, and $p_{\text{ret\_d}}$ as the probability of a bit "0" changing to "1" in the discharged cell. Therefore, when taking into account retention errors, the probability of a bit being "0" (a discharged cell) on a memory word p_{d} will change to p_{d}'; this is given by:

$$p_d' = p_d\cdot\left(1-p_{ret\_d}\right)+p_c\cdot p_{ret\_c} \quad (2)$$

where p_{c} is the probability of a charged cell on the word, i.e., $p_{\text{c}} = 1 - p_{\text{d}}$. The case of p_{c}' that is changed from p_{c} is similar. As per equation (2), it is then possible to find the probability of a given memory word changing to another word and causing a false positive or a false negative for some applications. Therefore, based on equations (1) and (2), the impact of the refresh time of an eDRAM on its false positive rate ($\triangle FPR$) or false negative rate ($\triangle FNR$), can be represented as a function given by:

$$\Delta FPR\ or\ FNR(t) = f\left(p_d'\right) = f\left(p_d, t\right) \quad (3)$$

This function will be developed for specific implementations for a few case studies in the next sections.

4 Case study 1: Bloom Filters

4.1 Overview

Bloom Filters (BFs) are widely used in applications such as computing and networking systems that need to check if a given element belongs to a set [23],[24]. There are many algorithms that can achieve this function, but they are usually based on saving all elements of the set and comparing them with the given element. This would result in very high storage as well as incurring in a large searching latency as the number of elements in the set increases. Whereas BFs proposed in 1970 can filter the elements efficiently [25] and thus are commonly implemented in electronic circuits to perform approximate membership checking, BFs have also been extended to support the removal of elements by adding counters as proposed in [15],[26].

A BF is a data structure that consists of an array of "0" and "1"; it is stored in a high-speed memory (e.g., eDRAM). The content of a BF array is accessed using some hash functions; each element is mapped to some points in the BF array based on the hash functions, and those points are set to "1". When searching for a given element, we can simply check whether its associated points in the array are "1". If any point is "0", the element does not belong to the set. If all points are "1", the element is likely to exist in the set.

Figure 4 shows a diagram of a BF, and its operations are defined as follows:

- ***Insertion***: Each element is mapped to q bits that are set to "1" in the BF array **S (**originally all zeros on all m bits) that correspond to the positions of q hash functions. For example, when inserting element x, find its positions based on the hash functions $h_1(x)$, $h_2(x)$, … , $h_q(x)$ first, and then set them to "1".
- ***Query***: when checking if a given element (e.g. y) belongs to the BF array, the bits on the positions of $h_1(y)$, $h_2(y)$, … , $h_q(y)$ are read. Only when all of the bits are "1", the element y is considered to be a member of the BF array. Otherwise, in the case of searching an element z, if there is at least one "0" on its positions associated to $h_1(z)$, $h_2(z)$, … , $h_q(z)$, then it is not a member of this set.

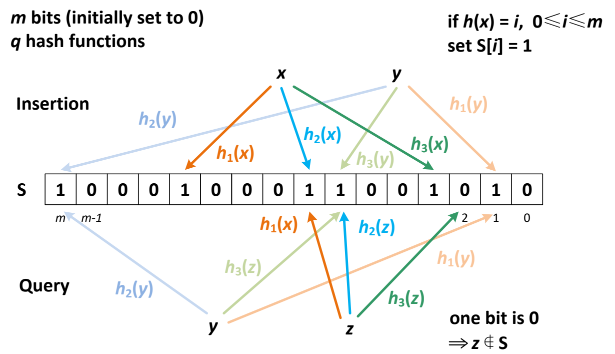

Figure 4 Illustration of a Bloom filter

These operations ensure that all inserted elements are found in the query process; false negatives therefore do not occur in an error-free BF. However, false positives can happen as a given element that is not in the BF may have "1" on the positions of its hash functions. For example, in Figure 4, assume that element y has not been inserted in the BF array (i.e., it is not a member of this set). When testing for y, the positions that correspond to $h_1(y)$, $h_2(y)$, … , $h_q(y)$ are checked. However, the value on all of these positions can be "1" due to other elements; in this case, such given element would be considered to belong to the set (while it is not), hence a false positive occurs. BFs are usually designed to have an acceptable false positive rate, for example 1% or lower, so that they have a small impact on performance.

4.2 Proposed Error-tolerant eDRAM-based BFs

As discussed before, BFs are used in many networking applications to speed up packet processing [27]. In some of these applications, the on-chip BFs are used to avoid accesses to external memory; therefore, BFs can be implemented by using eDRAMs [28],[29]. However, in an eDRAM-based BF, retention errors changing a stored "1" to "0" would cause false negatives. Thus, some protection techniques like ECCs are needed to protect the BF against such errors. Ad-hoc protection techniques have also been proposed for BFs, but they still require at least the use of a parity check per memory word [30]. Note that "1" to "0" errors are deleterious because they can lead to false negatives, while "0" to "1" errors can only degrade the false positive rate. This is the motivation to propose a redundancy-free error-tolerant scheme to avoid false negatives in eDRAM-based BFs.

Figure 5 shows the block diagram of the proposed scheme. When storing the BF array into the eDRAM, we can first simply code "1" as "0" and "0" as "1", and then write the data into the memory. This ensures that if a retention error changes a charged to a discharged cell, this is equivalent to an error from "0" to "1" in the BF array. In this case, the proposed scheme may only introduce false positives, so false negatives are avoided. Even when an additional number of "1" are introduced in the filter, their impact on the increase of the false positive rate can be negligible. Moreover, the refresh rate is improved and the refresh power consumption is significantly reduced without any other protection technique.

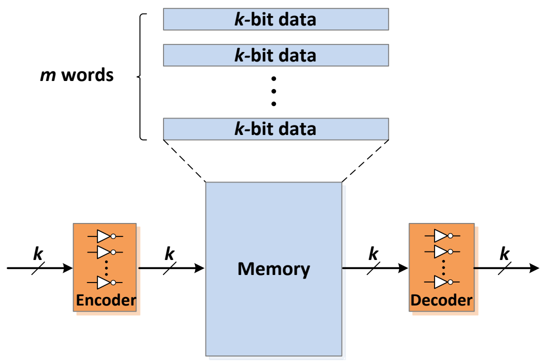

Figure 5 Block diagram of the proposed scheme

Assume there are q hash functions used for mapping elements to the positions in a BF array. The false positive rate (FPR) in the case that the eDRAM is error-free, can be approximated as $p_1{}^q$, where p_1 is the probability of a bit being "1" on the BF array output from the decoder. When considering possible retention errors, p_1 increases to $p_1{}'$ as modeled in equation (2) (p_1 ($p_1{}'$) on the decoded data in the filter is equal to p_d ($p_\text{d}{}'$) on the memory as we perform an inversion coding). Therefore, we can establish the relevant increase of FPR ($\triangle FPR$) presented previously in equation (3) of Section 3 as:

$$\Delta FPR \cong \frac{\left(p_1\left(1-p_{ret\_d}\right)+\left(1-p_1\right)p_{ret\_c}\right)^q - p_1{}^q}{p_1{}^q} \tag{4}$$

From Figure 3 and equation (1), the probability of a single bit retention failure is extremely low (<<1) in an eDRAM with a refresh period in the order of a few tens of microseconds in practice. For example p_ret is 7.7E-11 when the refresh time is 30μs. Therefore, equation (4) can be approximated by developing the first part of the numerator and is given by:

$$\Delta FPR \cong \frac{q(1-p_1)\,p_{ret\_c}\cdot\left(1-(q-1)\,p_{ret\_d}\right)}{p_1} - q\cdot p_{ret\_d} \qquad (5)$$

When the proposed scheme is utilized, the refresh time of the eDRAM can be decreased and p_{ret} will also increase to p_{ret}'. In this case, the impact of the proposed scheme on the FPR can be estimated as the updated relevant increase of FPR ($\triangle FPR'$) given by:

$$\begin{aligned}\Delta FPR' &\cong \frac{\left(p_1\left(1-p_{ret\_d}'\right)+(1-p_1)\,p_{ret\_c}'\right)^q-\left(p_1\left(1-p_{ret\_d}\right)+(1-p_1)\,p_{ret\_c}\right)^q}{\left(p_1\left(1-p_{ret\_d}\right)+(1-p_1)\,p_{ret\_c}\right)^q}\\ &\cong \frac{q(1-p_1)\left(p_{ret\_c}'\left(1-(q-1)\,p_{ret\_d}'\right)-p_{ret\_c}\left(1-(q-1)\,p_{ret\_d}\right)\right)}{p_1}+q\left(p_{ret\_d}-p_{ret\_d}'\right)\end{aligned} \qquad (6)$$

Assume that variations in the false positive rate smaller than 1% from the error free case are negligible. According to equations (1) and (6), the maximum refresh time when introducing a negligible effect on FPR is given by:

$$1.27\times10^{-17}\cdot t^{4.59} \le \frac{0.01+q\cdot p_{ret\_d}'}{1-p_{ret\_d}'\cdot(q-1)}\cdot\frac{p_1}{q(1-p_1)}+p_{ret\_d}' \qquad (7)$$

As discussed previously, retention errors in eDRAMs are highly asymmetric; only a negligible portion is from "0" to "1" errors, for example they accounts for only 0.005% of all retention errors as per the experiment results of [18]. Therefore, an approximate model can be established from equation (7); this is given by:

$$1.27\times10^{-17}\cdot t^{4.59} \le \frac{0.01\cdot p_1}{q(1-p_1)} \qquad (8)$$

Figure 6 shows the maximum refresh time for those cases (such that p_1 varies from 0.1 to 0.8 and q from 2 to 6) as found in practice. For example, when p_1=0.8, q=2, the refresh time can increase from a few tens of microseconds (e.g. 30μs) up to 2046μs by using the proposed scheme, with an increase of only 1% in the FPR. Even in the worst case of p_1=0.1, q=6, the refresh time increases up to 737μs.

Therefore, the proposed scheme that is redundancy-free can significantly improve the refresh time by introducing a negligible effect on the false positive rate in BFs. Note that when the refresh time is increased, the refresh power consumption is substantially reduced, because the refresh frequency decreases. In the next subsection, the saving in terms of refresh power is evaluated.

The proposed scheme can protect eDRAM-based BFs without any ECC or parity, and significantly increase the refresh time, thus reducing power consumption. Moreover, it can tolerate all possible retention errors (including single and multiple bit errors) on each word (i.e., $k\cdot(1-p_1)\cdot p_{\text{ret\_c}}$ bit errors, where k is the length of a memory word). Note that the proposed error-tolerant scheme can also be applied to other similar structures like counting BFs, or count-min sketches.

The proposed scheme is highly accurate when the "0" to "1" retention errors have a negligible contribution. However, when these errors cannot be neglected, the proposed scheme may generate false negatives as not appearing in error free BFs. In this case, the proposed scheme can be enhanced by utilizing any pattern with q-1 ones from the hash functions as element matching to avoid false negatives, but increasing the false positive rate (as proposed in [31]). Therefore by combining both techniques, the proposed scheme is also applicable to scenarios in which the errors are asymmetric, but the number of "0" to "1" errors is not negligible.

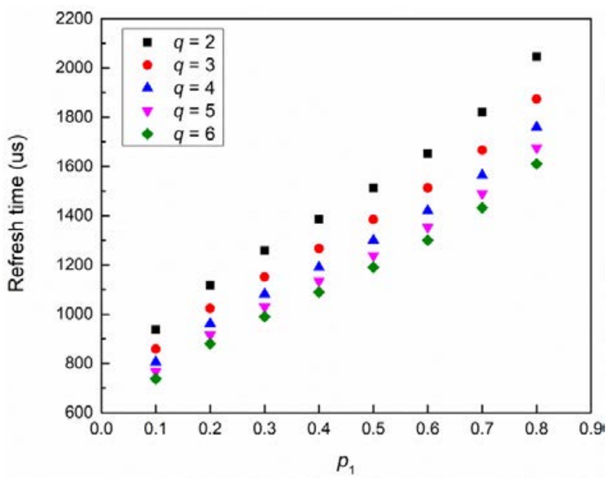

Figure 6 Estimated refresh time for different p_1 and q

4.3 Evaluation

Initially, the refresh power consumption reduction pro vided by the proposed scheme is evaluated. Then, the overheads for the area, delay and power consumption introduced by the proposed scheme are also evaluated. Existing schemes using SEC-DED codes and 5EC-6ED BCH codes are also compared to show the advantage of the proposed scheme.

Refresh power consumption: Consider that when we increase the refresh time t to t', the saving in the refresh power ($Power_{\text{saving}}$) is given by:

$$Power_{saving} = \frac{\frac{1}{t}\cdot Power_0-\frac{1}{t'}\cdot Power_0}{\frac{1}{t}\cdot Power_0} = 1-\frac{t}{t'} \qquad (9)$$

where $Power_0$ is the power consumption of each refresh operation.

Consider equation (9); the saving in refresh power is related to an increase in refresh time. Therefore, in the example considered previously when the refresh time is increased from 30μs to 2046μs, the refresh power is reduced by 98.5% with a negligible impact (+1%) on the false positive rate. This is better than using ECCs, such as the SEC-DED codes and 5EC-6ED BCH codes [13]. For example, when using SEC-DED (5EC-6ED BCH) codes, the refresh time can be improved from 30μs to 150μs (440μs), and the refresh power can be reduced by 80.0% (93.2%).

Table 1 Features and memory overhead introduced for different schemes

Scheme	Data size	# parity bits/word	Increase in memory size	Worst FPR[*1]	ECC Function level
SEC-DED	8-bit	4	50.0%	1×10^{-6}	1-bit error correction and 2-bit error detection
	16-bit	6	37.5%		
	32-bit	7	21.8%		
	64-bit	8	12.5%		
5EC-6ED BCH	8-bit	20	250.0%	1×10^{-6}	5-bit error correction and 6-bit error detection
	16-bit	27	168.8%		
	32-bit	27	84.4%		
	64-bit	35	54.7%		
Proposed	**8-bit**	**0**	**0**	$\mathbf{1\times10^{-6}+1\times10^{-8}}$	**All possible error correction[*2]**
	16-bit	**0**	**0**		
	32-bit	**0**	**0**		
	64-bit	**0**	**0**		

[*1] The worst FPR is estimated when p_1= 0.1, q=6.

[*2] The proposed scheme can deal with all possible errors (including single and multiple bit errors) by introducing a negligible impact on FPR so its ECC function level can be considered as all possible error correction.

Overheads: The proposed scheme is redundancy-free so has no impact on memory size; the SEC-DED codes and 5EC-6ED BCH codes need few parity bits stored in each memory word. Table 1 shows the increase in memory introduced by different schemes, the worst FPR and their ECC function level when protecting the eDRAM-based BFs with 8, 16, 32, 64-bit words. The proposed scheme significantly saves memory size at the cost of introducing a slight impact on the FPR.

To compare the protection circuitry (encoder and decoder) needed by the different schemes, designs were implemented in HDL and synthesized by the Synopsys Design Compiler mapping to a TSMC 65nm library. The tool was set with maximum effort in terms of area and delay. Table 2 shows the comparison results for the encoders and decoders (the area and power results include the corresponding figures for the encoders and decoders; also for the delay, decoders are included because the operational speed depends on the decoding latency); Figures 7 plots the PDP (Power Delay Product) results. The proposed scheme achieves significant improvements over existing schemes in all cases.

Table 2 Comparison for protection circuitry

Data size	Scheme	Area (μm^2)	Delay (ns)	Power (mW)
8-bit	SEC-DED	234.4	0.61	0.28
	5EC-6ED BCH	2052.4	1.12	2.08
	Proposed	19.2	**0.10**	0.02
16-bit	SEC-DED	467.6	0.65	0.6
	5EC-6ED BCH	55468	1.74	33.52
	Proposed	38.4	**0.10**	0.02
32-bit	SEC-DED	921.6	0.73	1.44
	5EC-6ED BCH	270936.8	1.88	80.75
	Proposed	76.8	**0.10**	0.06
64-bit	SEC-DED	1871.2	0.90	2.72
	5EC-6ED BCH	1102215.2	2.05	165.35
	Proposed	153.6	**0.10**	0.12

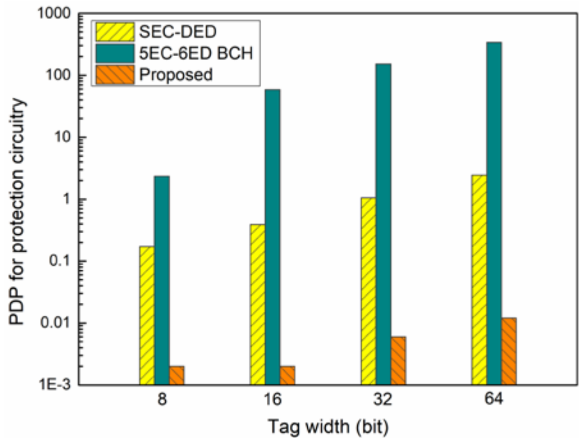

Figure 7 PDP (ns·mW) for the protection circuitry of different schemes versus data size

5 Case Study 2: Caches

5.1 Overview

In a processor, there are typically several levels of caches. They are smaller in capacity, but faster than the main memory, and store frequently used data to increase the throughput of a processor. Generally, the first level cache can be implemented in SRAMs due to the fast speed; higher level/larger caches can be implemented based on embedded DRAMs (eDRAMs) to allow for a larger capacity [3],[4]. Each entry in a cache has at least a Tag that identifies the address to which the entry corresponds in the next level cache or the main memory and the stored Data (also denoted as Value). To access the cache, we first read the position that corresponds to the incoming address. Then for each way, the Tag is compared with the incoming Tag. If there is a match (also denoted as a "hit"), Data that corresponds to the Tag is read out next. Otherwise, it is a "miss" (mismatch); Data will then be fetched from the next cache level or the main memory. Most caches, for example the Instruction Caches (ICs) or Translation Lookaside Buffers (TLBs) also have a valid bit in each entry to identify if the data is valid or invalid. So the incoming Tag is only compared with the Tags that are valid.

A soft error, for example a retention error in an eDRAM-based cache can have different effects depending whether

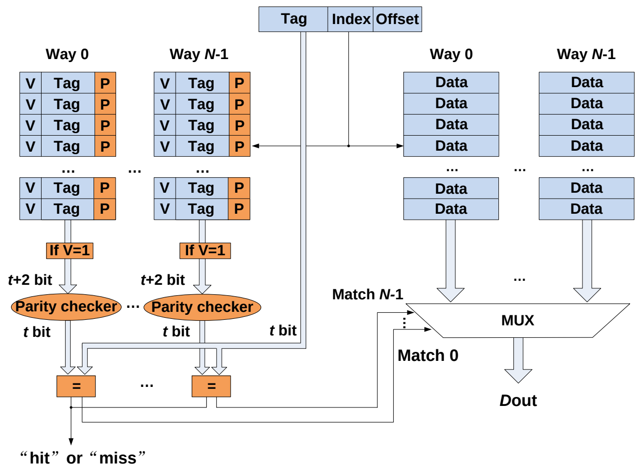

Figure 8 An *N*-way Cache protected with a single parity bit

it affects the Tag or Data [27]. If the Tag is corrupted, there are two scenarios, false positive and false negative. A false positive occurs when an incoming Tag matches a stored Tag that has been modified by an error. In this case, an incorrect Data is returned and this may lead to a system error. False negatives occur when an incoming Tag should have matched a stored Tag; however, it is not matched because it has been changed by the error. In this case, there is no data corruption as long as the Data stored in the entry (whose Tag was affected by the error), has not been modified. If the Data stored in cache has been changed and not updated in main memory, data corruption will occur. Therefore, false negatives are only an issue for write-back caches, but not for read-only or write-through caches (e.g., ICs or TLBs) because they would only cause a "miss" that can degrade performance but it cannot affect the correctness of the results.

ECCs are widely used to protect caches against soft errors [32],[33]. For example, a single parity bit can be added for the Tag in each entry to detect single bit errors. Figure 8 illustrates this scheme for an *N*-way cache. The access to the cache first reads the position that corresponds to the incoming address based on Index and Offset; then for a valid Tag in each way, a parity checker is used to detect if there is an error in the Tag. If there is an error, then the Tag is not considered further in the process. After checking that Tags are correct, they are compared with the incoming Tag; if there is no match, a "miss" occurs, and the system retrieves the data from the next level cache or the main memory. If there is a match, Data that corresponds to the matching Tag is read out. As per Figure 8, this error-tolerant scheme requires the following elements:

- A parity bit P for each Tag and the valid bit.
- A logic to check if the Tag is valid or not (can be simply implemented by an *and* logic).
- A parity checker circuitry with t+2 inputs for each way.
- A t bit comparison logic for each way, where t is the width of the Tag bits.

This scheme can also be used to detect single errors on Data bits by using the parity to cover both Tag and Data bits in each entry. For applications in which multiple errors are a concern, stronger function ECCs or interleaved parity bits can be used, at the cost of a larger overhead introduced by the increased number of parity bits per line and the parity checking procedure, also at a higher hardware complexity [34],[35].

5.2 Proposed Error-tolerant eDRAM-based Caches

In this subsection, a redundancy-free error-tolerant scheme for eDRAM-based read-only or write-through caches (e.g., ICs or TLBs) is proposed. This scheme can detect single bit retention errors and improves the refresh rate to reduce the refresh power consumption.

The proposed scheme protects caches based on propagating the error on the Tag in each entry to the valid bit and consists of three parts as follows.

Part 1: *Overload the valid bit.* In a cache entry, when an eDRAM cell in which the capacitor discharges fastest causes a retention error on a Tag bit, the first step of the proposed scheme propagates the error to the valid bit of this entry, as in the approach in [36] that propagated an error to a sensitive bit. This can be achieved by overloading the valid bit with the *xor* result of the original value of the valid bit and all stored Tag bits. Consider a cache entry that stores a valid bit V, a Tag of width t having bits T_1, T_2, ... T_t , and a Data of width d having bits D_1, D_2, ..., D_d. In Step 1, we overload the valid bit by using:

$$V' = V \ xor \ T_1 \ xor \ T_2 \ xor \ \ldots xor \ T_t \tag{10}$$

where V' is the recomputed valid bit and is overloaded in the entry. This procedure is the same as the encoder for a single parity bit in the single parity bit scheme. Instead of introducing an additional cell to store the parity bit (P in Figure 8), we only overload the recomputed valid bit V' with no additional cell in the proposed scheme.

Part 2: *Recover the valid bit.* When the selected entry is read from the cache, the valid bit V is recovered by performing the *xor* of V' with the bits stored in Tag, i.e., equation (11) below.

$$V_{rec} = V' \ xor \ T_1 \ xor \ T_2 \ xor \ \ldots xor \ T_t \tag{11}$$

where V_{rec} is the recovered valid bit.

In this case, any error on the Tag changes also the recovered valid bit and thus, any valid entry is changed into an invalid entry (the recovered V is "0", while the value stored was "1"). Therefore, entries that have an error on the Tag are effectively removed. This procedure can be implemented similarly to the parity checker in Figure 8 but needing one less input (therefore the parity bit is saved).

An issue with this approach is that an error on Tag bits can turn an invalid entry into a valid entry and therefore, this may result in a potential error. However, as retention errors in an eDRAM are asymmetric and only cause bit flips from "1" to "0", this can be avoided by resetting the Tag value T to all zeros when we invalidate an entry. Therefore, an invalid entry will have an all zeros value and cannot be affected by retention errors.

Part 3: *Perform a direct comparison.* Consider the overloaded valid bit in valid and invalid entries first. If the entry was valid, then from Part 1 and equation (10) the overloaded valid bit V' should be equal to 1 *xor* T. If the entry was invalid, V' should be "0" according to equation (10) as all Tag bits stored in an invalid entry are set to "0" in Part

2. Therefore, after Part 2, we would only have three cases: 1) an error-free Tag with V_{rec}=1; 2) an error-free Tag with V_{rec}=0 (retention errors considered in this case are asymmetric in an eDRAM and cannot change "0" to "1"; thus an invalid entry will always be all zeros); 3) an erroneous Tag with V_{rec}=0 (a single error that affects one Tag bit has been propagated to V_{rec} and invalidates the entry) .

As when searching for an incoming Tag V_{in}, only valid entries must be considered. The direct comparison is performed between T_{in} and T read from the entries when V_{rec}=1 to find the incoming Tag. If there is a match, a "hit" occurs; otherwise it is a case of "miss". If the read Tag is invalid, a "miss" case will also occur.

Figure 9 shows an example of the proposed scheme. In the error-free case of Figure 9 a), there is always a "hit" if the incoming Tag matches the read Tag, then Data that corresponds to the matched Tag is used. For the single bit error case in b), the read entry is invalidated so there is always a "miss" and no error is propagated to next operation.

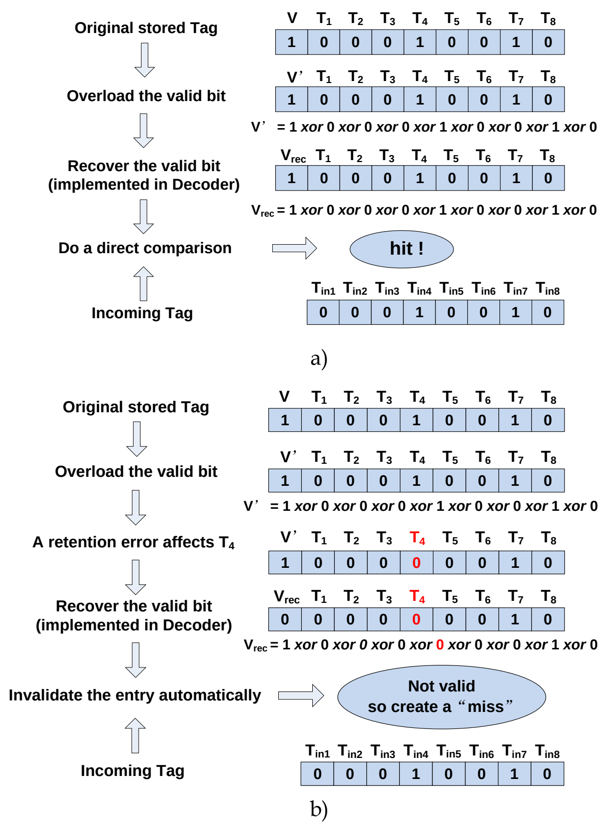
Original stored Tag
V T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 1 0 0 1 0
Overload the valid bit
V' T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 1 0 0 1 0
V' = 1 xor 0 xor 0 xor 0 xor 1 xor 0 xor 0 xor 1 xor 0
Recover the valid bit (implemented in Decoder)
Vrec T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 1 0 0 1 0
Vrec = 1 xor 0 xor 0 xor 0 xor 1 xor 0 xor 0 xor 1 xor 0
Do a direct comparison
hit !
Incoming Tag
Tin1 Tin2 Tin3 Tin4 Tin5 Tin6 Tin7 Tin8
0 0 0 1 0 0 1 0
a)
Original stored Tag
V T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 1 0 0 1 0
Overload the valid bit
V' T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 1 0 0 1 0
V' = 1 xor 0 xor 0 xor 0 xor 1 xor 0 xor 0 xor 1 xor 0
A retention error affects T4
V' T1 T2 T3 T4 T5 T6 T7 T8
1 0 0 0 0 0 0 1 0
Recover the valid bit (implemented in Decoder)
Vrec T1 T2 T3 T4 T5 T6 T7 T8
0 0 0 0 0 0 0 1 0
Vrec = 1 xor 0 xor 0 xor 0 xor 0 xor 0 xor 0 xor 1 xor 0
Invalidate the entry automatically
Not valid so create a "miss"
Incoming Tag
Tin1 Tin2 Tin3 Tin4 Tin5 Tin6 Tin7 Tin8
0 0 0 1 0 0 1 0
b)

Figure 9 Example used to illustrate the proposed scheme: a) error-free case; b) the case of a single error affecting bit T4.

As illustrated in Figure 10, the following hardware is required or the proposed scheme in an N-way cache.

- A decoder with t+1 inputs for each way to recover the valid bit for the read Tag.
- A t bit comparison logic for each way, where t is the width of the Tag bits.
- A logic block to check whether Tag is valid or not (can be simply implemented by an *and* logic).

Compared to the hardware required by the existing scheme using a single parity bit discussed in Section 5.1,

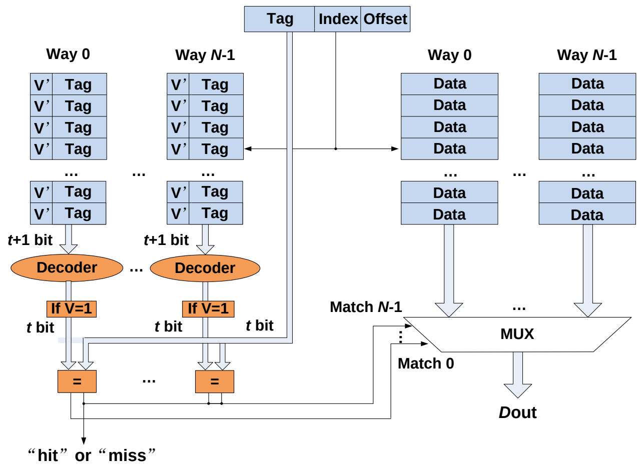

Figure 10 An N-way Cache protected with the proposed scheme

the advantage of the proposed scheme is that it is redundancy-free, i.e., there is no need to add any parity bit to detect errors, so it has no impact on memory overhead. Moreover, the proposed scheme needs a simpler circuitry for the decoder of each entry than for the parity checker used in the single parity bit scheme due to the single input in the circuit. The overhead introduced by the proposed scheme will be evaluated in subsection 5.3 to show its benefits.

As any single error on the Tag bits is detected, the proposed scheme does not introduce any false positive but it may cause false negatives for a match between the incoming Tag and the read Tag when it has been modified by the error. However, as discussed previously, this is not a problem for read-only or write-through caches, because there is no data corruption. The increase of the false negative rate ($\triangle FNR$) caused by the proposed scheme is equal to the probability of a retention error, i.e., $\triangle FPR$=p_{ret}; and it is rather low (refer to Figure 3 or equation (1)). To avoid more retention errors, the refresh operation is necessary for the eDRAM. However, the refresh rate can be significantly improved in the proposed scheme, because it can deal with single errors, so the refresh power consumption can be reduced. Note that single errors on Data bits can also be detected if the overloaded valid bit covers both Tag and Data bits.

Additionally, the proposed scheme can detect multiple errors that affect any odd number of bits, because these errors will always be propagated to the valid bit and invalid the entry. Under errors that affect an even number of bits, the proposed scheme will fail, because the entry with a valid bit "1" will still keep a valid probability, causing a fault positive due to the errors. In this case, the proposed scheme can be extended by combining it with an extra parity bit. For example, combined with a single parity bit that covers all odd bits, detection of any two adjacent bit errors

Table 3 Comparison for different caches

Cache configuration*			**Scheme**	**Area**		**Delay**		**Power**	
	N	*t*		**μm²**	%	**ns**	%	**mW**	%
DRAM cache in [37]	4	24	SPB	874.8	100	0.66	100	0.55	100
			Proposed	**862.8**	98.63	**0.64**	96.97	**0.46**	83.64
L2 TLB in ARM Cortex-A76 [38]	5	32	SPB	1526.4	100	0.68	100	1.07	100
			Proposed	**1391.6**	91.17	**0.65**	95.59	**0.82**	76.64
L2 Cache in ARM Cortex-A76 [38]	8	36	SPB	2706.4	100	0.74	100	1.82	100
			Proposed	**2679.9**	99.02	**0.71**	95.95	**1.71**	93.96
eTag DRAM cache in [39]	16	40	SPB	6000.0	100	0.78	100	3.94	100
			Proposed	**5988.8**	99.81	**0.75**	96.15	**3.88**	98.48
L3 cache in ARM CCN-508 [40]	16	44	SPB	6560.8	100	0.83	100	4.70	100
			Proposed	**6541.6**	99.71	**0.79**	95.18	**4.54**	96.60

* *N* is the associative (way); *t* is the Tag width (bit).

is accomplished because one of the two erroneous bits (i.e., the one in the odd position) is detected by the parity checker, and so causing a "miss". Even though this extended scheme is not redundancy-free, compared with a conventional interleaved parity bit scheme [41] (in which two parity bits are needed to detect double adjacent bit errors), one parity bit per entry is saved, thus reducing the memory overhead, and keeping a similar latency (the overloading/recovering of the valid bit and encoding/decoding of the parity bit operate in parallel).

5.3 Evaluation

In this subsection, the proposed scheme for single asymmetric error detection in eDRAM-based caches is evaluated. The scheme has been implemented and the overhead has been estimated and compared with the existing single parity bit (SPB) scheme in terms of memory size and protection circuitry.

Memory size: The proposed scheme is redundancy-free so the memory size remains unaltered; the existing SPB scheme needs an additional cell in each entry to store the parity bit. Therefore, let the width of each Tag in an eDRAM-based cache with an *N*-way and *E*-entry configuration be *t*. The saving ratio of the proposed scheme and the SPB scheme in terms of total number of Tag memory cells is given by:

$$\textit{Saving ratio} = 1 - \frac{E \cdot N(t+1)}{E \cdot N(t+2)} = \frac{1}{t+2} \quad (12)$$

For an example of a cache with 24-bit Tags [37], the proposed scheme reduces 3.8% the memory cells compared to the SPB scheme. As memories in most cases account for a significant portion of the area of processors, this advantage in terms of memory size makes the proposed scheme very attractive for protecting caches in many applications.

Protection Circuitry: Compared to the SPB scheme, the proposed scheme requires similar hardware in the protection circuitry and has the advantage in terms of complexity of the decoder of saving one input in the circuit. Both schemes have been studied for different caches, including the DRAM cache with 24-bit Tags for the 4-way configuration in [37], the L2 TLB with 32-bit Tags for the 5-way configurations and L2 cache with 36-bit Tags for the 8-way configurations in the ARM Cortex-A76 processor [38], the eTag DRAM cache with 40-bit Tags for the 16-way configurations in [39], and the L3 cache with 44-bit Tags for the

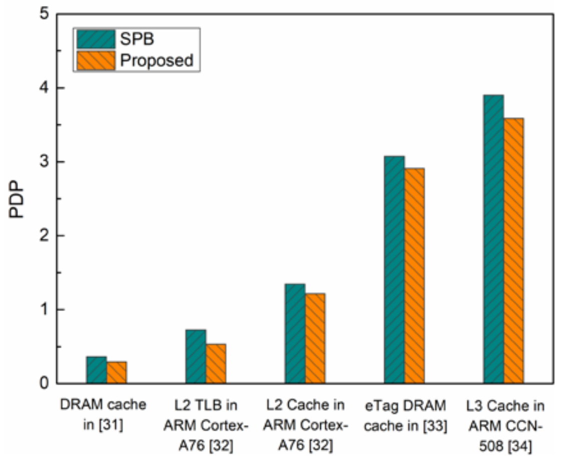

Figure 11 PDP (ns·mW) for the SPB scheme and proposed scheme for caches with different tag width.

16-way configuration in the ARM CCN-508 cache coherent network [40]. Designs have been implemented in HDL and mapped to a 65nm library from TSMC using Design Compiler configured for area and delay optimization to obtain the best results for these metrics.

Table 3 shows the synthesis results for different schemes; Figure 11 plots the PDP results. The proposed scheme shows improvements in all cases. For example, when protecting the Tag RAM of the L2 cache in the ARM cortex-A76 processor, the proposed scheme saves 1.4% in area and 18.9% in PDP. Recall that as the proposed scheme and the SPB scheme detect single retention errors, then they can also be used to reduce the refresh power consumption of the eDRAM.

6 Case Study 3: Modular Redundancy

In many applications, a computer system must continue to operate in the presence of a failure [42]. In these cases, a common solution is to use Modular Redundancy, i.e. to replicate components multiple times to ensure that the failure of at least a component does not compromise the operation of the entire system [43]. For example, in Triple Modular Redundancy (TMR) the component is triplicated and voting is performed among the outputs of the three copies to ensure that a correct result is provided in the presence of a single component failure. Similarly, when using five copies, failures in two of them can be tolerated. Dual Modular Redundancy (DMR) uses only two copies, so a single failure can only be detected. If Quadruple Modular Redundancy (QMR) is used, a single failure can be corrected and failures in two components can be detected.

For memories, Modular Redundancy targets component level failures that affect the entire memory or large parts of it (for example, failures in the access logic or control logic); it can additionally be used to correct retention errors. In this context, an interesting observation is that when errors are highly asymmetric, most errors, can be corrected by using DMR. So, when a bit has a different value in each of the two memories, the logical OR of both bits is provided as output (Figure 12 a)). This ensures that "1" to "0" errors (that represent the vast majority) in a single module are corrected. The cases in which retention errors cannot be corrected include: i) "1" to "0" errors occurring in both modules; ii) "0" to "1" errors occurring in any one or both modules. Therefore, for a given bit, the probability of a retention error that cannot be corrected when using this approach is given by:

$$p_{ret\_err\_DMR} = p_{ret\_c}{}^{2} + C_2^1 \cdot p_{ret\_d} \cdot \left(1 - p_{ret\_d}\right) + p_{ret\_d}{}^{2} \quad (13)$$

The same scheme can be extended to Quadruple Modular Redundancy. In this case, correction is accomplished when the four values of a bit are two ones and two zeros indicating that two errors are present. Again, as the "1" to "0" errors are dominant, we can output a "1". In this case, the correction logic is slightly more complex. The four values are input to a majority gate (which is shown in Figure 12 b)); the threshold of the majority gate is two, i.e., a "1" is provided as output as long as there are at least two inputs of "1". The cases in which errors cannot be corrected and an erroneous data is provided as output include: i) "1" to "0" errors occur in three or all four modules; ii) "0" to "1" errors occur in two or more modules. When using the proposed scheme in QMR, the probability of having an incorrect retention error for a bit is given by:

$$\begin{aligned} p_{ret\_err\_QMR} = {} & C_4^3 \cdot p_{ret\_c}{}^{3} \cdot \left(1 - p_{ret\_c}\right) + p_{ret\_c}{}^{4} + C_4^2 \cdot p_{ret\_d}{}^{2} \cdot \left(1 - p_{ret\_d}\right)^2 \\ & + C_4^3 \cdot p_{ret\_d}{}^{3} \cdot \left(1 - p_{ret\_d}\right) + p_{ret\_d}{}^{4} \end{aligned} \quad (14)$$

As example, the error correction capability (in terms of the probability of an uncorrectable retention error) for the proposed protection schemes are shown in Figure 13 for different retention error probabilities (i.e., at different *t*) in Figure 3 by assuming "0" to "1" errors account for a negligible fractional part of all retention errors (i.e., $p_{ret\_c} \approx p_{ret}$ and $p_{ret\_d} \approx 0$). Figure 13 shows that the proposed schemes can correct almost all retention errors during this range of refresh time.

7 Discussion

In the first case study that focuses on eDRAM-based BFs, a redundancy-free error-tolerant scheme has been proposed to deal with all possible retention errors (including single and multiple bit errors) in the eDRAM. This is achieved by an inversion coding for the BF data, so simply encoding "1" in the BF data as "0" and "0" as "1" in the

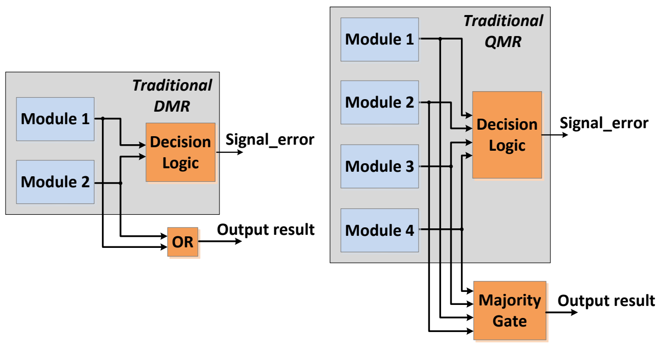

Figure 12 Illustration for the proposed scheme for: a) DMR; b) QMR.

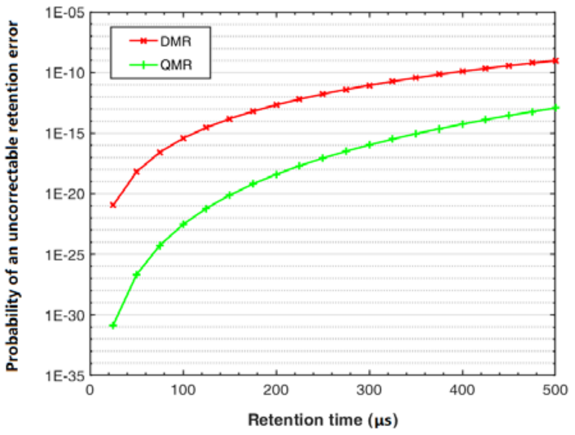

Figure 13 The probability of a retention error that cannot be corrected by the proposed schemes in DMR and QMR.

write operation for the memory. Then asymmetric retention errors (only changing charged cells to discharged ones) would only change errors from "0" to "1" but not from "1" to "0" in the decoded BF array; this increases the false positive rate (FPR) of the BF. However, BFs have an intrinsic FPR due to the nature of the data structure. We have modeled the impact of the proposed scheme on the FPR with the refresh time of an eDRAM. It has been shown that when the increase is negligible (limited to 1%), the refresh rate can be significantly improved (for example from 30μs to 2046μs, the refresh power consumption can be reduced by 98.5%) and better than for existing error-tolerant schemes using SEC-DED and 5EC-6ED BCH codes. In terms of additional overhead introduced by the different coding schemes, the proposed scheme also has a significant advantage both in memory size (saving of 50.0% memory compared to SEC-DED codes and 250.0% for the 5EC-6ED BCH codes for a memory with 8-bit words), and protection circuitry (savings of 86.5% in area, 69.7% in delay, 83.3% in power consumption for the encoder, and 94.1% in area, 83.6% in delay, 95.5% in power consumption for the decoder to the SEC-DED codes; 94.2% in area, 74.4% in delay, 95.6% in power consumption for the encoder, and 99.5%% in area, 91.1% in delay, 99.5% in power consumption for the decoder to the SEC-DED codes in a 8-bit words memory). The proposed scheme is also applicable to other similar data structures like counting BFs, or count-min sketches.

In the second case study, we have presented a redundancy-free error-tolerant scheme to detect single errors in eDRAM-based caches that have a valid bit in each entry, such as Instruction caches (ICs) or Translation Lookaside Buffers (TLBs). By setting all bits to "0" in the invalid entries when writing them into the memory, no false positives or false negatives occur as asymmetric retention errors of the eDRAM cannot change the stored "0" to "1". By overloading the valid bit to the *xor* result of the valid bit with all Tag bits in the valid entries, any retention error is propagated to the recovered valid bit, hence invalidating the entry. Therefore, the proposed scheme can only cause false negatives in the cache, but no data corruption. Compared to the existing single parity bit scheme for single errors in caches, the proposed scheme saves one cell storing the parity bit in each word, thus reducing the memory size. In terms of protection circuitry, the proposed also incurs in a lower overhead (for example, it reduces 1.4% of the area and 18.9% of the PDP for 24-bit Tags).

In the third case study, we have proposed redundancy-free schemes for the traditional DMR and QMR. In addition to retaining single module error detection (single module error correction and double module error detection) of DMR (QMR), the asymmetry of retention errors has been exploited to provide additional error correction capability (>99.99%) for these errors in all modules by adding an OR logic (majority gate), but still without any additional memory redundancy.

Moreover, the proposed schemes are also applicable in the presence of radiation induced soft errors in eDRAM cells, because these errors are also asymmetric (a change from "1" to "0" only) as introduced previously. If radiation affects other parts of the eDRAM circuits (such as the sense amplifiers) and causing symmetric errors, the proposed schemes must be combined with other protection techniques for such parts to provide a full error tolerant capability.

8 Conclusion

This paper has exploited the asymmetry of retention errors in eDRAMs for redundancy-free error-tolerant design. Applications based on eDRAMs that can tolerate false positives or false negatives have been considered. By combining asymmetry and false positive/negative error behaviors, a comprehensive model has been proposed; eDRAM-based Bloom Filters (BFs) and eDRAM-based caches have been analyzed as case studies. It has been shown that these schemes can efficiently deal with retention errors by introducing a negligible impact on the false positive rate or false negative rate; however, they do not need any ECC or parity and can be implemented by simpler decoder and encoder circuits than existing coding approaches. Moreover, the schemes can significantly improve the refresh rate of the eDRAMs, thus reducing power consumption. The asymmetry of retention errors has also been used for additional error correction capability in Modular Redundancy schemes without introducing memory redundancy.

References

[1] J.M. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits-A Design Perspective", 2nd ed, 2003.

[2] C.W. Slayman, "Cache and Memory Error Detection, Correction, and Reduction Techniques for Terrestrial Servers and Workstations", IEEE Trans. on Device and Materials Reliability, vol. 5, no. 3, pp. 397-404, 2005.

[3] B. Jacob, S. Ng, D. Wang, "Memory Systems: Cache, DRAM, Disk", Morgan Kaufmann, 2010.

[4] B. Sinharoy, et al. "IBM POWER8 Processor Core Microarchitecture", IBM Journal of Research and Development, vol. 59, no. 1, pp. 2,1-21, 2015.

[5] W.K. Luk and R. Nair, "DRAM Cache with on-Demand Reload", U.S. Patent 7,805,658, issued September 28, 2010.

[6] A. Agrawal, A. Ansari, and J. Torrellas, "Mosaic: Exploiting the Spatial Locality of Process Variation to Reduce Refresh Energy in on-Chip eDRAM Modules", IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), pp. 84-95, 2014.

[7] R.C. Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies", IEEE Trans. Device mater. Reliab. vol.5, no.3, pp.301-316, 2015.

[8] B. Narasimham, W.K. Luk, "A Multi-Bit Detection Scheme for DRAM using Partial Sums with Parallel Counters", IEEE International Reliability Physics Symposium, pp.202-205, 2008.

[9] S. Buchner, A. Campbell, R. Reed, and et.al. "Angular Dependence of Multiple-bit Upsets Induced by Protons in a 16 Mbit DRAM", IEEE Transactions on Nuclear Science, vol. 51, no. 6, pp.3270-3277, 2004.

[10] A. Makihara, H. Shindou, N. Nemoto, and et.al. "Analysis of Single-Ion Multiple-Bit Upset in High-Density DRAMs", IEEE Transactions on Nuclear Science, vol. 47, no. 6, pp. 2400-2404, 2000.

[11] S. Lin and D. J. Costello, "Error Control Coding", 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[12] K. Namba, F. Lombardi, "Non-Binary Orthogonal Latin Square Codes for a Multilevel Phase Charge Memory (PCM)", IEEE Trans. on Computers, vol.64, no. 7, pp.2092-2097, 2015.

[13] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu, "Reducing Cache Power with Low-Cost, Multi-bit Error Correcting Codes", ACM SIGARCH Computer Architecture News, vol. 38, no. 3, pp. 83-93, 2010.

[14] P. Emma, W. Reohr, and M. Meterelliyoz, "Rethinking Refresh: Increasing Availability and Reducing Power in DRAM for Cache Applications", IEEE Micro, pp. 47-56, 2008.

[15] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: a Scalable Wide Area Web Cache Sharing Protocol", IEEE/ACM Trans. on Networking, vol.8, no.3, pp.281-293, 2000.

[16] S. Pontarelli, P. Reviriego and M. Mitzenmacher, "EMOMA: Exact Match in One Memory Access," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 11, pp. 2120-2133, 2018

[17] G. Cormode and S. Muthukrishnan, "An Improved Data Stream Summary: The Count-Min Sketch and its Applications", Journal of Algorithms, vol.55, no.1, pp.58–75, 2005.

[18] K. Kraft, D.M. Mathew, C. Sudarshan, M. Jung, C. Weis, N. When, F. Longnos, "Efficient Coding Scheme for DDR4 Memory Subsystems", ACM, in Proceedings of the International Symposium on Memory Systems, pp.148-157, 2018.

[19] S. Wang, M.N. Bojnordi, X. Guo, E. Lpek, "Content Aware Refresh: Exploiting the Asymmetry of DRAM Retention Errors to Reduce the Refresh Frequency of Less Vulnerable Data", IEEE Trans. Computers, vol.68, no.3, pp.362-374, pp.362-374, 2019.

[20] B. Narasimham, W.K. Luk, "A Multi-Bit Detection Scheme for DRAM using Partial Sums with Parallel Counters", IEEE International Reliability Physics Symposium, pp.202-205, 2008.

[21] K. Kraft, C. Sudarshan, D.M. Mathew, et al., "Improving the Error Behavior of DRAM by Exploiting its Z-Channel Property", IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1492-1495, 2018.

[22] W. Kong, P.C. Parries, G. Wang, S.S. Iyer, "Analysis of Retention Time Distribution of Embedded DRAM-A new Method to Characterize Acrosschip Threshold Voltage Variation", in Proceedings of IEEE International Test Conference (ITC 2008), pp. 1-7, 2008.

[23] A. Broder and M. Mitzenmacher, "Network Applications of Bloom filters: A Survey", Internet Math., vol. 1, no. 4, pp. 485-509, 2003.

[24] Y.Kanizo, D.Hayand, I.Keslassy,"Maximizing the Throughput of Hash Tables in Network Devices with Combined SRAM/DRAM Memory", IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 3, pp. 796-809, 2015.

[25] B. Bloom, "Space/Time Tradeoffs in Hash Coding with Allowable Errors", Comm. ACM, vol. 13, no. 7, pp. 422-426, 1970.

[26] E. Safi, A. Moshovos, A. Veneris, "L-CBF: A Low-Power, Fast Counting Bloom Filter Architecture", IEEE Trans. on Very Large Scale Integ. (VLSI) Systems, vol.16, no.6, pp.628-638, 2008.

[27] S. Mukherjee, "Architecture Design for Soft Errors", Morgan Kaufmann, 2008.

[28] M. Jimeno, K. J. Christensen, A. Roginsky. "Two-tier Bloom filter to achieve faster membership testing", Electronics Letters vol. 44, no. 7, pp.503-504, 2008

[29] N. Mcvicar, C.C. Lin, S. Hauck. "K-mer counting using Bloom filters with an FPGA-attached HMC." In 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 203-210., 2017.

[30] P. Reviriego, S. Salvatore, J.A. Maestro, M. Ottavi. "A Method to Protect Bloom Filters from Soft Errors", IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), pp. 80-84, 2015.

[31] A. Sánchez-Macián, P. Reviriego, J.A. Maestro, S. Liu, "Single Event Transient Tolerant Bloom Filter Implementations", IEEE Trans. on Computers, vol. 66, no. 10, pp. 1831-1836, 2017.

[32] J. Kim, S. Kim, and Y. Lee. "SimTag: exploiting tag bits similarity to improve the reliability of the data caches", Proceedings of the Conference on Design, Automation and Test in Europe. European Design and Automation Association, 2010.

[33] S. Wang, J. Hu, and S. G. Ziavras. "Replicating tag entries for reliability enhancement in cache tag arrays", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 4, pp. 643-654, 2012.

[34] S. Liu, P. Reviriego and L. Xiao, "Evaluating Direct Compare for Double Error Correction Codes", IEEE Transactions on Device and Materials Reliability, vol.7, no.4, pp. 802-804, 2017.

[35] A. Gendler, A. Bramnik, A. Szapiro and Y. Sazeides, "Don't Correct the Tags in a Cache, Just Check Their Hamming Distance From the Lookup Tag", IEEE International Symposium on High Performance Computer Architecture (HPCA), pp.571-582, 2018.

[36] J. Martínez, J.A. Maestro, P. Reviriego, "A Scheme to Improve the Intrinsic Error Detection of the Instruction Set Architecture", IEEE Computer Architecture Letters, vol. 16, no 2, pp. 103-106, 2017.

[37] C.C. Huang, V. Nagarajan, "ATCache: Reducing DRAM Cache Latency via a Small SRAM Tag Cache", ACM, Proceedings of the 23rd international conference on Parallel architectures and compilation, pp. 51-60, 2014.

[38] "Arm Cortex-A76 Core Technical Reference Manual", Revision: r3p0, https://developer.arm.com, 2016.

[39] K.H. Yang, H.J. Tsai, C.Y. Li, et.al. "eTag: Tag-Comparison in Memory to Achieve Direct Data Access based on eDRAM to Improve Energy Efficiency of DRAM Cache", IEEE Trans. on Circuits and Systems I: Regular Papers, vol.64, no.4, pp.858-868, 2017.

[40] "Arm CoreLink CCN-508 Cache Coherent Network Technical Reference Manual", Revision: r0p1, https://developer.arm.com, 2014

[41] S. Baeg, S. Wen, and R. Wong, "SRAM Interleaving Distance Selection with a Soft Error Failure Model", IEEE Trans. Nucl. Sci., vol. 56, no. 4, pp. 2111–2118, 2009.

[42] N. Kanekawa, E.H. Ibe, T. Suge and Y. Uematsu, "Dependability in Electronic Systems", Springer Science & Business Media, New York, 2011.

[43] E. Fujiwara, "Code Design for Dependable Systems", 1st ed, Hoboken, NJ, USA: Wiley, 2006.