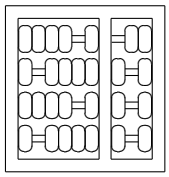

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Improving the Modeling and Analysis of Error Correction Techniques for Phase-Change Memory

Caio Hoffman *Luiz E. da S. Ramos*
Rodolfo J. de Azevedo *Guido C. S. de Araújo*

Technical Report - IC-13-34 - Relatório Técnico

December - 2013 - Dezembro

Improving the Modeling and Analysis of Error Correction Techniques for Phase-Change Memory

Caio Hoffman* Luiz E. da S. Ramos* Rodolfo J. de Azevedo*
Guido C. S. de Araújo*

2013-12-13

Abstract

Due to projections of high scalability, Phase-Change Memory (PCM) is seen as a new main memory for computer systems. In fact, PCM may even replace DRAM, whose scaling limitations require new lithography technologies that are still unknown. On the down-side, PCM has low endurance when compared with DRAM, i.e., on average, a PCM cell can only withstand 10^8 bit-flips (modification of stored bit values) before the cell fails. To address PCM's low endurance, Error Correction Techniques (ECTs) have been proposed, which aim at increasing PCM's lifetime. However, previous lifetime analyses of ECTs have not considered the difference between the bit-flip frequencies of data bits and code bits (used to correct errors in data bits). That observation is crucial for the correct analysis of the ECTs since high bit-flip frequencies lead to faster wear-out.

In this work, we improve the wear-out analysis of PCM by modeling and analyzing the bit-flip probabilities of five ECTs, namely, ECP, DRM, SECDED, SAFER, and FREE-p. We then use our analysis to evaluate the impact of error correction on the wear-out of PCM. To do our analyses, we mathematically modeled and simulated the ECTs using both a theoretical bit-flip probability (50%) and an empirical bit-flip rate (15%), obtained from the execution of SPEC CPU2006. Our results show clear endurance degradation in techniques that use error-correcting codes, which contradicts some previous results in the literature. Finally, we analyzed the power consumption of the ECTs, and found that ECP and SAFER exhibit the best trade-off between endurance and write energy.

1 Introduction

Phase-Change Memory (PCM) is a new memory technology considered a possible replacement for DRAM. PCM is byte-addressable, non-volatile, has multi-level cell (MLC) capability, and has demonstrated scalability beyond the 5 nm manufacturing process [9]. A concern about PCM that hinders its use as main memory is its low endurance. Currently a PCM cell withstands around 10^8 bit-flips (modification of stored bit values) [4] before

*Institute of Computing, University of Campinas. This work were sponsored by the grants CNPq (133161/2011-0) and FAPESP (2011/05266-3, 2013/05257-0).

failing. In fact, considering the variability in the manufacturing process, PCM cells may withstand even less than 10^8 bit-flips, and a single cell failure may invalidate an entire PCM chip. Besides, PCM's endurance is far from that of DRAM (considered unlimited) [7].

To extend the average lifetime of PCM chips, Error Correction Techniques (ECTs) have been used to ensure that a few cell failures will not cause a chip failure. Recently, some ECTs for PCM were proposed in the literature: DRM [2], ECP [7], SAFER [8], and FREE-p [10]. To correct cell failures, ECP and SAFER leverage the features of resistive memories, whereas DRM and FREE-p use Error-Correcting Codes (ECCs).

Unfortunately, the works that propose those ECTs have simplifying assumptions that overlook important characteristics of memory writes. First, they assume a **Bit-Flip Probability** (BFP) of 50%, i.e., on a write to a PCM memory block, every bit flips with a 50% probability. Second, they assume that the bit-flip behavior in data bits and code bits is identical. As we show in this paper, those assumptions can mislead the evaluation of PCM's lifetime while not enabling an accurate analysis of energy consumption. Moreover, Schechter *et al.* argue that ECCs speed up cell wear-out because any modification to data bits requires a rewrite of code bits (those used to correct errors in the data bits) [7]. However, the authors do not confirm their hypothesis with experimental results or theoretical analyses.

In this work, we carefully evaluate those hypotheses by mathematically modeling the bit-flip behavior in DRM, SECDED, ECP, SAFER, and FREE-p. Specifically, our models consider that data and code bits have different bit-flip frequencies. This approach not only enables a more accurate evaluation of PCM's endurance for each ECT, but also it enables computing PCM's write energy at a finer granularity. This kind of analysis and comparison had not been done in the original ECT research. In our results, we evaluate PCM's wear-out and compare our results to those in the literature. We also evaluate energy consumption, and their trade-off, for each ECT. Finally, we show results that support Schechter's hypothesis.

This work is organized as follows. Section 2 presents more details on PCM and describes the ECTs that we study. Section 3 describes the simulation model and the BFP model for each technique. In Section 4, we present and discuss our results. Finally, in Section 5 we conclude our work.

2 Background and Related Work

2.1 Phase-Change Memory and Bit-Failure

PCM cells are composed of a chalcogenide material sandwiched between two electrodes. One of the electrodes is connected to the chalcogenide through a material called heater, which modifies the state of a PCM cell during writes. A write is a thermal-mechanical stress process of switching the chalcogenide's resistive state between high (logical zero) and low (logical one). The write stress causes either delamination or diffusion on the components of PCM cells [1], resulting the cells become stuck at one of the states. If the memory system is unable to handle cell failures, the whole memory fails.

2.2 Error Correction Techniques

To prevent memory failures, memory subsystems typically utilize ECTs. For a memory with a limited lifetime, ECTs not only provide a mechanism to recover the information stored in a failed cell, but they can also extend memory lifetime. For PCM, some ECTs were proposed in recent years: DRM (*Dynamically Replicated Memory*), ECP (*Error-Correcting Pointers*), SAFER (*Stuck-at-Fault Error Recovery*) and FREE-p (*Fine-Grained Remapping With ECC and Embedded-Pointer*).

In this subsection, we describe the main ECTs found in the literature. We also describe SECDED (*Single Error Correction Double Error Detection*), traditionally used in DRAM, and tested in PCM [2, 7, 10].

2.2.1 Storage Overhead

Throughout this paper we use the term code bits as synonymous of meta data bits. Meta data is encoded information of the data, which provides means of detecting or recovering a missed information in the data. We define storage overhead (SO) as the length of the meta data, described as a percentile of the data's length.

In this work, a memory block is one 512-bit data block plus a SO of at most 64 code bits (which represents 12.5% of the data block's length, value widely adopted in the literature).

2.2.2 DRM

DRM reuses deallocated pages to extend memory lifetime. A page is deallocated when its first error is detected. The page remains deallocated until another page (without error overlap with the first page) happens to be deallocated. In that case, the two of them are paired up and form a single new logical page that becomes available to the memory system again. Pages with errors are paired and unpaired dynamically, since their compatibility may change as new errors occur and error overlapping changes. The parity scheme uses one code bit for every byte of data, which results in 64 code bits per memory block (12.5% of SO).

2.2.3 SECDED

A (n, k) Hamming code is an n-bit codeword with k data bits and $n - k$ code bits. The SECDED is a (127, 120) Hamming code that can detect and correct up to one error. Although 120 data bits are covered, only 64 of them are used in SECDED. Therefore, SECDED is a non-standard (71,64) Hamming code plus one parity bit. The extra parity bit detects a second error in one of the 71 bits of the codeword.

The resulting overhead is 8 code bits for every 64 data bits (12.5%). For our data block of 512 bits, there are eight SECDED blocks. Hence, SECDED can correct up to 8 errors, as long as they do not occur in the same SECDED block.

2.2.4 ECP

Providing e spare cells to replace failed cells in a memory block, ECP is able to correct up to e errors. When a cell failure is detected in a data block of size N bits, a pointer of $\log_2 N$ bits will store the position of the failed cell in the block. Every read or write requested to that block will need a pointer decoding, in order to redirect a write to the spare cell or to provide a reading of the correct cell value stored in the spare cell.

A spare cell along with a pointer is called an ECP entry. The total overhead of keeping all ECP entries is $e + e \cdot \log_2 N + 1$. The 1 refers to a *full bit* that indicates whether all ECP entries are being used. The notation ECP_e means that a memory block has e ECP entries. With $N = 512$ bits, ECP_6 has 61 bits of SO (close to 12.5%).

2.2.5 SAFER

Taking advantage of the failed cells themselves to correctly store the data, SAFER links a failed cell to a group of data bits in a memory block, so that there is only one failed cell per group. When a write to the block is about to modify that group, if the state of the failed cell logically matches the value of the bit to be stored, then the update to the group is committed. Otherwise, the bits to be written the group are inverted, before being stored. A spare bit is used to identify whether the bits stored in a group are inverted or not.

The most complex task in SAFER is forming the groups. The partitioning of a memory block depends on the position of the failures in that block. For that reason, the number of errors that SAFER can correct varies. For example, the configuration SAFER_{32} (with a SO close to 12.5% for 512-bit data blocks) can correct up to 32 failed cells in the best scenario, but depending on the position of the failed cells, at most only 7 errors can be corrected.

2.2.6 FREE-p

Using a (572,512) BCH code to help correcting hard and soft errors, FREE-p can correct and detect up to six errors. The (572,512) BCH code is adapted from the (1023,963) BCH. Moreover, in addition to the 572 bits of codeword, one parity bit is added, making the ECC used in FREE-p a 6EC-7ED. The power of FREE-p lies on its mechanism for recycling failed blocks (those with more than 4 hard errors), which are used to store pointers. When those blocks receive a write request, FREE-p uses the pointer stored in them to redirect the writes to a page without errors. Hence, a page with failed blocks remains functional until all its blocks have failed.

3 Simulation Model

3.1 Our Hypotheses

There are five hypotheses that guide our model. **First**, PCM cells may not have the same lifetime. This comes from the known imperfection of any process fabrication. **Second**, writes only modify the bits that differ from the bits already stored. This consideration refers to the mechanism known differential writes [3] or read-before-write [10]. **Third**, each

cell stores a single bit. **Fourth**, the memory is perfectly wear-leveled. That is, the wear-out is spread over all memory blocks, without hotspots. This hypothesis is consistent with works in the literature [6]. **Fifth**, there is an intra-row wear leveling. Essentially, this is the same idea of the fourth hypothesis, but within every memory block. This hypothesis avoids favoring data bits or code bits in wearing out memory blocks.

3.2 Mathematical Representation

We describe how we compute the number of writes suffered by a memory during its lifetime without considering a particular technique. Basically, the number of writes will depend on the number of simulation steps, which depends on the ability of the technique to handle the failures.

A memory can be emulated as a unidimensional array Q of length C_M, where each element contains an integer that represents the remaining lifetime of a memory cell. The length is defined as $C_M = N \cdot N_L \cdot N_P$, where N is the data block size, N_L is the number of blocks in a page, and N_P is the number of pages. The array Q is created by ascendingly sorting the elements from a matrix of integers. This matrix was randomly generated and based on Gaussian distribution with mean μ and standard deviation σ (first hypothesis).

Our simulation handles one element of Q per simulation step. Being that, at the k-th step, all memory cells must have suffered Q_{k-1} writes, except for the $k-1$ memory cells that had already failed; since the elements $Q_0, \ldots, Q_{k-1}$ have a lower remaining lifetime than the elements $Q_k, \ldots, Q_{C_M-1}$.

When the first block in a page becomes unable to recover from a failure, the page is deallocated. In that case, a deallocated page should not be taken into account. Let $F_P(k)$ be the number of deallocated pages in the k-th simulation step and let the wear rate be (1) the percentile reduction of the number of writes. Intuitively, $f_{wr}(0) = 1$ (there is no reduction) and $F_P(0) = 0$ (no failed pages) before the simulation starts. Let c_k be the lifetime difference between the elements located in positions k and $k-1$. Considering $c_0 = 0$, we can describe a simultaneous write in all memory pages as (2).

$$f_{wr}(k) = f_{wr}(k-1) \cdot \frac{N_P - F_P(k)}{N_P - F_P(k-1)} \tag{1}$$

$$W(k) = \sum_{i=1}^{k} c_i \cdot f_{wr}(i) \tag{2}$$

Note that when we say all the pages of the memory, we refer to the fourth hypothesis. The perfect wear-leveling ensures that when we perform c_k writes, all the blocks that do not belong to failed pages have suffered c_k writes. That is why we use the wear rate, which removes the unwritable pages from the counting. Systematically, to compute the number of writes, for the whole memory, we should multiply $W(k)$ by N_L and N_P. However, they are constants and do not need to be computed at the simulation time, thus (2) is enough for our evaluations.

3.3 Bit-Flip Probability Modeling

In a memory that stores one bit per cell (third hypothesis), we can model a write in a memory block as a sequence of n (the block length) independent events, with probability p of success (flipping). We have a success when a bit has its value flipped; otherwise, we have a failure. That is, we model memory writes with a binomial distribution. Therefore, the expected number of flipping bits in a write is $n \cdot p$. By the second and the fifth hypotheses, every bit in a block should last longer than its original lifetime (by a fraction $1/p$).

For the following BFP models, we consider that the probability of success p is unknown. Every memory block has $N + \Delta_X$ bits (where the subindex X is the technique, and can be omitted for intelligibility). Of those bits, $N = 512$ bits have p probability of flipping (data BFP) and Δ_X bits have $P_X(p)$ probability of flipping (code BFP, which is a function p). Thus, we define the **weighted BFP** Υ_X as the weighted arithmetic mean of p and $P_X(p)$. Equation (3) uses Υ_X to compute the total writes in our simulation model for each technique.

$$\mathbb{W}(k) = \frac{W(k)}{\Upsilon_X} \tag{3}$$

3.3.1 The DRM BFP model

By considering that one parity bit covers n bits. We know that a parity bit is 1 when the covered bits have an odd number of bits with value 1. If an even number of covered bits change their value, from 0 to 1 or vice-versa, the parity is not changed because there is compensation. Hence, the BFP of the parity bit depends on the probability of an odd number of flips occurring in the byte.

Now, consider the model of a binomial distribution to describe a memory write. The probability of parity change is the probability of occurring one, or three, or five flips, and so on until $n-1$ flips, for an n-bit memory word (where n is a power of two). Mathematically,

$$P_{\text{PARITY}}(n, p) = \sum_{i=1}^{\frac{n}{2}} B(2 \cdot i - 1; n, p) \tag{4}$$

The notation B(k;n,p) represents the probability mass function for the binomial distribution. In DRM, $\Delta = 64$ bits, i.e., the number of parity bits. We have Δ bits with probability $P_{\text{PARITY}}(b, p)$ of flipping, where b is the byte's length, and N bits with probability p of flipping. The DRM's BFP is:

$$\Upsilon_{\text{DRM}} = p \cdot \frac{N}{N + \Delta} + P_{\text{PARITY}}(b, p) \cdot \frac{\Delta}{N + \Delta} \tag{5}$$

3.3.2 The SECDED BFP model

has the same principle of the model for parity bits in DRM, i.e., the parity bits of SECDED depend on counting an odd number of flips in a write. Nevertheless, the model of SECDED becomes more complex because it is a non-standard Hamming code, thus the quantity of

covered bits by a parity bit is not constant. We formulate an equation that gives us the quantity of covered bits for every parity bit in a (n, k) Hamming code. The quantity $q_h(j)$ of covered bits by the i-th parity bit, located in the j-th codeword's position (such as $j = 2^{i-1}$), is given by (6).

$$k_h(j) = j \cdot \left(\left\lceil \frac{n-(j-1)}{j} \right\rceil - \left\lceil \frac{n-(j-1)}{2 \cdot j} \right\rceil \right)$$

$$q_h(j) = k_h(j) + \max\big(0, n-(j-1)-2 \cdot k_h(j)\big) - 1 \tag{6}$$

In a standard (n, k) Hamming code, for definition, the i-th parity bit covers all codeword bits that have value 1 in the i-th bit of the binary word that describes a bit position in the codeword. Given m parity bits, such that $m = n - k$ is an integer greater than 2, we can alternatively define a standard Hamming code as $(2^m - 1, 2^m - m - 1)$ code [5]. Thus, to describe a bit position in the codeword we need $\lceil \log_2(2^m - 1) \rceil$ bits. We know that $\log_2 2^{m-1} < \lceil \log_2(2^m - 1) \rceil \leq \log_2 2^m = m$ (Property B.1).

For a given bit, in all possibles m-bit words, there is always $2^m/2$ words whose that bit is 1. Therefore, in a (n, k) Hamming code, the i-th parity bit covers $n/2$ bits. With m parity bits, there is $(n - m)/2$ data bits covered by each parity bit (the coverage is redudant). In other words, the data bits coverage is $(2^m - 1 - m)/2 = (k/2)$. We demonstrate that (6) follows this case, see Section B.1.

We were not capable of proofing our equation for every Hamming code. Nevertheless, we provide the Table 3 that stands the results of (6) for the (71, 64) hamming code used in SECDED. Hence, using the quantity of covered bits for each parity bit, we describe the weighted BFP for a (n, k) Hamming code as:

$$P_{\text{HC}}(p) = \frac{1}{n} \cdot \big(k \cdot p + \sum_{i=1}^{n-k} P_{\text{PARITY}}(q_h(2^{i-1}), p)\big)$$

The SECDED has an additional bit of parity covering the n bits of codeword, therefore:

$$\Upsilon_{\text{SECDED}} = \frac{1}{n+1} \cdot \big(n \cdot P_{\text{HC}}(p) + P_{\text{PARITY}}(n, P_{\text{HC}})\big) \tag{7}$$

In spite of $N = 512$ bits and $\Delta = 64$ bits for a memory block with SECDED, it is not necessary modify (7), because the SECDED works individually by blocks of $n = 72$ bits. Therefore, multiplying and weighting the equation by the number of SECDED blocks in a memory block will not change the model.

3.3.3 The ECP BFP model

In this model, the essential is to keep in mind that neither the pointer bits nor the full bit will be frequently written. The worst case scenario for these bits happens at the last ECP entry, which is used as a counter of the used ECP entries. That entry will be updated at most e times in the ECP_e, during the block's lifetime. Since we assume an average cell

endurance greater than a million writes, we can consider that those bits have a negligible BFP. Consequently, N data bits and e spare bits suffer the vast majority of writes. We assume that, in the lifetime of the block, e errors will occur, and considering our fifth hypothesis the e spare bits will receive many writes. Let $\Delta = e + e \cdot \log_2 N + 1$, thus:

$$\Upsilon_{\mathrm{ECP}_e} = p \cdot \frac{N+e}{N+\Delta} \tag{8}$$

3.3.4 The SAFER BFP model

The proposition of this model takes the number of failed bits into account. We do that because, in SAFER, the number of failed bits in a memory block changes the BFP of the data bits. Specifically, in the absence of failures, each bit at a given memory block has a BFP of p upon every new write to the block. However, when a failure occurs, the block is sub-divided into groups of bits, with at most one failed bit (stuck at one or zero) per group. For explanation purposes, suppose we have a group G with a failed bit g_i, and G' will overwrite G upon a new block write. Suppose that the bit g'_i in G' will overwrite g_i. Then, in addition to the original BFP of p, we need to consider the probability $q = 1/2$ that the stuck value of g_i matches the value of g'_i (otherwise, SAFER will invert all bits in G' before overwriting G). If they match, the BFP when G' overwrites G is p and the resulting BFP is $p \cdot q$. If they do not match, the BFP when G' overwrites G is $1-p$ and the resulting BFP is $(1-p) \cdot (1-q)$.

For the SAFER_k (which can correct up to k errors) there are initially k groups, and they require k spare bits to inform whether the data is stored directly or not. These bits have a BFP of p or $(1-q)$. As errors happen, $k-e$ bits will keep a BFP of p, and the other e bits will have BFP of $1-q$, since their state will depend on the probability that g_i does not match g'_i. Furthermore, SAFER needs $\lceil \log_2 k \rceil \cdot \lceil \log_2 \lceil \log_2 n \rceil \rceil + \lceil \log_2 (\lceil \log_2 k \rceil + 1) \rceil$ bits to organize the groups and keep them with only one error. Like the pointers in ECP these bits are written very few times; thus, we assume their BFP is zero. Δ is $k + \lceil \log_2 k \rceil \cdot \lceil \log_2 \lceil \log_2 N \rceil \rceil + \lceil \log_2 (\lceil \log_2 k \rceil + 1) \rceil$ bits.

Being k a power of 2, like N, the number of bits in a group is N/k. This means that with e errors in a memory block, e groups of (N/k) bits have a BFP of $p \cdot q$ or $(1-p) \cdot (1-q)$. The previous observations lead us to the following BFP $(N + (k-e) - (\frac{N}{k} \cdot e)) \cdot p + (\frac{N}{k} \cdot e) \cdot q \cdot p + (\frac{N}{k} \cdot e) \cdot (1-q) \cdot (1-p) + e \cdot (1-q)$, which can be simplified into (9).

$$\Upsilon_{\mathrm{SAFER}}(e) = \frac{(N+k-e) \cdot p + \frac{N}{k} \cdot e \cdot \left(\frac{1}{2} - p\right) + e \cdot \frac{1}{2}}{N+\Delta} \tag{9}$$

Our simulation model cannot handle this bit-flip model, because every memory block will have its own number of errors. Hence, for simplification, we calculate the average number of errors of the memory blocks at each simulation step. Therefore, the computation of writes at each simulation step is done using $P_{\mathrm{SAFER}}(e_\mu(i))$. Where $e_\mu(i)$ is the average number of errors at the i-th simulation step. In this case, in our simulation we replace (3)

with (10).

$$\mathbb{W}(k) = \sum_{i=1}^{k} \frac{c_i \cdot f_{wr}(i)}{\Upsilon_{\text{SAFER}}(e_\mu(i))} \tag{10}$$

3.3.5 The FREE-p BFP model

For a (n, k) BCH code, we define $D = \{d_0, d_1, \ldots, d_{k-1}\}$ as the binary data word with k bits, $G = \{g_0, g_1, \ldots, g_m\}$ as the generator polynomial, where $m = n - k$. Finally, we define $V = \{v_0, v_1, \ldots, v_{m-1}\}$ as the verification code.

To illustrate our model, we take a generic (7,4) BCH code showed in the Fig. 1. The division is a process of continuous addition modulo 2 between the elements of a subset that is an element from $\mathcal{P}(L)$, where $L = \{\mathfrak{L}_1, \mathfrak{L}_2, \mathfrak{L}_3, \mathfrak{L}_4\}$. More formally, we say that the bits of V result from some combination of the bits of G and this combination is represented by some element of $\mathcal{P}(L)$ over the application of $f_\oplus$ (defined by the property B.3), that is, an element belonging to $\mathcal{L} = \bigcup_{P \in \mathcal{P}(L)} f_\oplus(P) = \{0, \mathfrak{L}_1, \mathfrak{L}_2, \mathfrak{L}_3, \mathfrak{L}_4, \mathfrak{L}_1 \oplus \mathfrak{L}_2, \mathfrak{L}_1 \oplus \mathfrak{L}_3, \ldots, \mathfrak{L}_1 \oplus \mathfrak{L}_2 \oplus \mathfrak{L}_3 \oplus \mathfrak{L}_4\}$.

For example, suppose that some division is represented by $\mathfrak{L}_2 \oplus \mathfrak{L}_3$, this implies that the result of $V = \{v_0, v_1, v_2\}$ is equal to $\{0, g_0, g_0 \oplus g_1\}$. We know that $|\mathcal{L}| = 16$ and all the possible values for v_2 are $\{0, g_0, g_1, g_2, g_0 \oplus g_1, g_0 \oplus g_2, g_1 \oplus g_2, g_0 \oplus g_1 \oplus g_2\}$, for v_1 are $\{0, g_0, g_1, g_0 \oplus g_1\}$ and for v_0 are $\{0, g_0\}$. Note that we see 8 combinations for v_2, however there are 16 that actually could happen, because every combination with $\mathfrak{L}_1$ results in a combination that already exists.

Still regarding the (7,4) BCH code, suppose that there is a data word D and its code V stored in a memory block. If a new data word D' and its code V' will be stored, the probability of occurring $d_3 = d'_3$, $d_2 = d'_2$, $d_1 = d'_1$ and $d_0 = d'_0$ (the probability of no bit flips) is $(1-p)^4$. If any bit differs from the corresponding stored bit, the probability of occurring a bit-flip is $1 - (1-p)^4$. Consider the new code bit v'_2. From the above, there are 16 possible results among which exactly half shall be 1 and the other half 0 (see the stated proposition B.2 and its corollary B.2.1). Regardless of v'_2 being 1 or 0, when v'_2 overwrites v_2, there would be a flip in 8 out of 16 chances. However, the probability of zero bit flips (namely, $(1-p)^4$) is not a chance of flipping. This leads us to 8/15 changes of occurring a BFP of $1 - (1-p)^4$, for every the code bit.

Generalizing for a data word $D = \{d_0, d_1, d_2, \ldots, d_{k-1}\}$ and its code $V = \{v_0, v_1, \ldots, v_{l-1}\}$ stored in a memory block, the new data word D' and its code V' that are going to be stored,

	d_3	d_2	d_1	d_0	0	0	0	g_3	g_2	g_1	g_0
$\mathfrak{L}_1$:	g_3	g_2	g_1	g_0							
$\mathfrak{L}_2$:		g_3	g_2	g_1	g_0						
$\mathfrak{L}_3$:			g_3	g_2	g_1	g_0					
$\mathfrak{L}_4$:				g_3	g_2	g_1	g_0				
					v_2	v_1	v_0				

Figure 1: Calculation of a generic verification code for the (7,4) BCH code.

will incur in a bit-flip for any code bit with probability $1-(1-p)^k$, which may happen with probability $2^{k-1}/(2^k-1)$. Therefore,

$$P_{\text{code}}(p,k) = (1-(1-p)^k) \cdot \frac{2^{k-1}}{2^k-1} \tag{11}$$

In the (n,k) BCH code used in FREE-p, the memory parameters are defined by $k=N$ and $n-k=\Delta-1$. We subtract 1 to disregard the extra parity bit, since the total SO Δ is the sum of the code bits from BCH plus the parity bit. The BFP of (n,k) BCH code is given by (12) and for FREE-p is given by (13).

$$P_{\text{BCH}} = \frac{1}{n} \cdot \big(k \cdot p + (n-k) \cdot P_{\text{code}}(p,k)\big) \tag{12}$$

$$\Upsilon_{\text{FREE-p}} = \frac{1}{n+1} \cdot \big(P_{\text{PARITY}}(n, P_{\text{BCH}}) + n \cdot P_{\text{BCH}}\big) \tag{13}$$

4 Results and Discussion

In this section, we present and discuss our results. Specifically, we compare the studied ECTs according to their weighted BFP; wear-out impact on PCM; energy consumption; and trade-off between wear-out and energy consumption.

4.1 Weighted BFP

Table 1 shows the weighted BFPs (Υ_X) for every technique (X) computed using our models for each value of p (the BFP of data bits). The higher the value of Υ_X, the worst the technique performs in terms of PCM wear-out. For SAFER, because its BFP depends on the number of error, Υ is the average BFP across 1000 simulations.

Table 1: Weighted BFPs (Υ_X) for each technique (X), and for different BFPs of data bits (p).

p	Υ_{ECP_6}	Υ_{DRM}	Υ_{SECDED}	$\mu(\Upsilon_{\text{SAFER}_{32}})$	$\Upsilon_{\text{FREE-p}}$
10%	9.0%	13.5%	14.3%	12.6%	14.3%
15%	13.6%	18.6%	18.8%	17.0%	18.8%
20%	18.1%	23.2%	23.3%	21.5%	23.2%
30%	27.1%	31.2%	32.2%	30.3%	32.1%
40%	36.2%	41.1%	41.1%	39.1%	41.1%
50%	45.2%	50.0%	50.0%	48.0%	50.0%
60%	54.2%	58.9%	58.9%	56.8%	58.9%
70%	63.3%	67.8%	67.8%	65.7%	67.9%
80%	72.3%	76.6%	76.7%	74.5%	76.8%
90%	81.4%	84.6%	85.7%	83.3%	85.7%
100%	90.4%	88.9%	89.6%	92.2%	94.7%

We observe that when p is below 50%, the ECC-based techniques (DRM, SECDEC, and FREE-p) have a higher Υ_X than the other techniques. The reason is that, in the ECC-based techniques, the BFP of code bits is higher than that of data bits, which accelerates the wear-out. Note that without intra-row wear leveling, the wear-out induced by bit-flips in the code bits would speed up the failure of memory blocks. For $p \leq 30\%$, SAFER performs as bad as the ECC-based techniques. This happens because SAFER's bit-inversion mechanism increases the BFP of bit groups in the presence of errors. The only technique that shows a consistently low weighted BFP is ECP, which benefits from infrequent updates to its code bits. For $p \geq 50\%$, all techniques exhibit a higher BFP in data bits than in code bits. In ECC-based techniques, this happens because the BFP of code bits is close to 50% (almost independently of p). In SAFER, the BFP of bit groups tend to 50%, due SAFER's bit-inversion mechanism. Finally, in ECP, the infrequent writes to code bits result in lower values of Υ for every p.

4.2 Theoretical and Empirical BFP

For our wear-out analysis of PCM, we feed two parameters into our BFP models: an empirical bit-flip rate ($p = 15\%$) and the theoretical BFP assumed in the literature ($p = 50\%$). We obtained $p = 15\%$ by instrumenting SPEC CPU2006 in our cache memory simulator (based on Intel's PIN Tools). In particular, we simulate a two-level cache (L1: 64 KBytes and L2: 2 MBytes), both levels with 64B blocks, LRU replacement, and a write-back replacement policy. We found $p = 15\%$, which corroborates with the value in [3]. For consistency, we include the row $p = 15\%$ in Table 1.

Using the empirical and theoretical parameters, we study the impact of ECTs on memory lifetime in two ways. First, we simulate the different ECTs using our BFP models (Section 3), which differentiate between the BFPs of data and code bits. The results are shown in Fig. 2 (a) and (b). Second, we study the ECTs without differentiating between the BFPs of data and code bits as it was done in previous works. The results are shown in Fig. 2 (c) and (d). In both studies, we run 1000 simulations of each ECT to account for the randomized lifetimes of PCM cells (Section 3.2). We also set $N = 512$ $N_L = 64$, $N_P = 256$, and the average cell endurance $\mu = 10^8$ with a standard deviation $\sigma = 2.5 \cdot 10^7$. Finally, in Fig. 2 (a) and (c), we show results for the experimental bit-flip rate, whereas in Fig. 2 (b) and (d) we show results for the theoretical BFP. The curves in all figures represent memory lifetime as the number of non-failed pages versus the number of writes to PCM. The lifetime of memory ends when the number of non-failed pages reaches zero.

The curves in Fig. 2 (d) represent the results as they were originally published in [7,8,10]. Although, in those works there was no direct comparison between FREE-p and SAFER, both had been found superior to ECP. The result is similar for the BFP of 15% (Fig. 2 (c)). In Fig. 2 (a) and (b), FREE-p exhibits degradation compared the previous results. In fact, FREE-p performs worse than ECP, which contradicts the previous result. In the Fig. 2 (b) there is no substantial change from (d). However, in (a) we see lifetime degradation for ECC-based techniques as compared to (c). Conversely, ECP improves its memory lifetime, whereas SAFER maintains approximately the same lifetime.

These results highlight our claim that differentiating between the bit-flip rates of data

and code bits enables more fine-grained and accurate studies. It is clear now that ECC can accelerate memory wear-out and reduce its lifetime, thereby supporting the hypothesis by Schechter et al. *[7].*

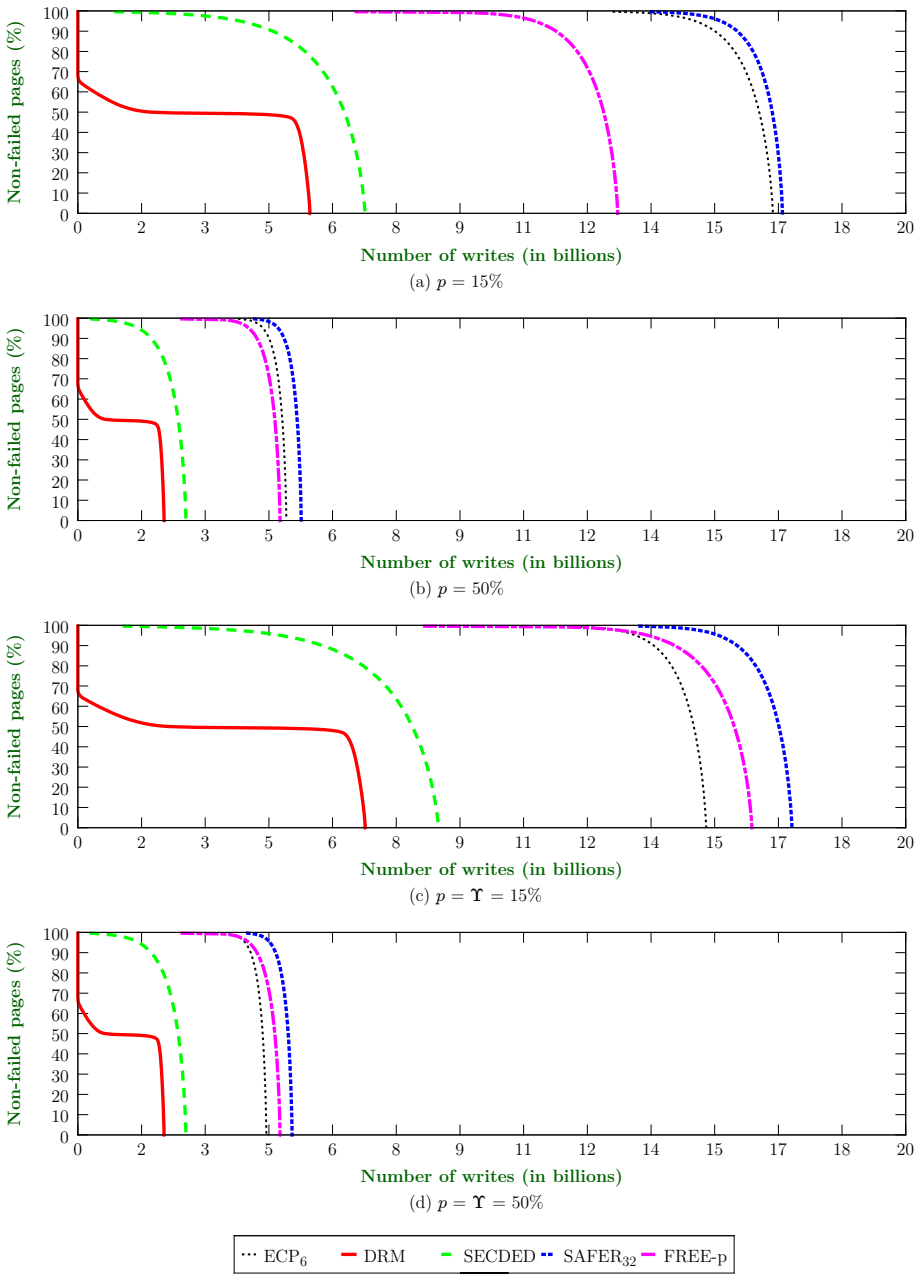

Figure 2: Memory lifetime with our BFP models for (a) and (b); and without our BFP models for (c) and (d).

4.3 Energy Consumption and Bit-Flip Probability

Since each ECT has a unique BFP for every value of p, their expected write energy should be different. To study the write energy we adopt the same cell write energy for all ECT, i.e., $E_{reset} = 19.2$ pJ and $E_{set} = 13.5$ pJ (from Lee *et al.* [4]). By using the expected number of bit flips in a write, we can define the expected write energy of each ECT X as in (14). Δ_X represents the SO for each ECT.

$$\xi_X = (N + \Delta_X) \cdot \Upsilon_X \cdot \big((E_{set} + E_{reset}) \cdot 0.5\big) \tag{14}$$

$$\Lambda_X = \frac{\mathbb{W}_X(\omega)}{\xi_X} \tag{15}$$

In addition to the write energy, we combine endurance and energy consumption into a single metric, to evaluate their trade-off. That metric (Λ) is expressed in (15). The simulation step ω denotes end of simulation, and $\mathbb{W}(\omega)$ represents the total number of writes withstood by the memory. Table 2 shows the Λ of all ECTs normalized to Λ_{ECP_6}. From the definition of Λ, the higher the memory endurance and the lower the write energy imposed by an ECT, the better the ECT's Λ. Based on this table, we conclude that ECP and SAFER exhibit the best trade-off between endurance and write energy, for both empirical bit-flip rate and theoretical BFP.

Table 2: Λ of the ECTs normalized to Λ_{ECP_6}. $p = 15\%$ is our empirical bit-flip rate and $p = 50\%$ is the theoretical BFP.

Υ_X	$\Lambda_X/\Lambda_{\text{ECP}_6}$	
	$p = 15\%$	$p = 50\%$
Υ_{DRM}	0.24	0.37
Υ_{SECDED}	0.30	0.47
$\mu(\Upsilon_{\text{SAFER}_{32}})$	0.81	1.02
$\Upsilon_{\text{FREE-p}}$	0.56	0.88

5 Conclusions

In this work, we introduced a more accurate and fine-grained approach to analyze the impact of ECTs on PCM's lifetime. Our approach also enables a meaningful computation of PCM write energy and its trade-off with memory endurance. This kind of analysis was not possible with previous ECT models, but it is still critical for modern computer systems.

We evaluated five state-of-the-art ECTs using our approach and compared our results to those in the literature. Our results extend and shed light on previous works that used simplifying assumptions; and support the argument that ECC-based techniques speed up the wear-out of PCM.

References

[1] G. Atwood and R. Bez. Current status of chalcogenide phase change memory. In *Device Research Conference Digest, 2005. DRC '05. 63rd*, volume 1, pages 29–33, june 2005.

[2] Engin Ipek, Jeremy Condit, Edmund B. Nightingale, Doug Burger, and Thomas Moscibroda. Dynamically replicated memory: building reliable systems from nanoscale resistive memories. In *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, ASPLOS '10, pages 3–14, New York, NY, USA, 2010. ACM.

[3] Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and B.R. Childers. Improving write operations in mlc phase change memory. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–10, feb. 2012.

[4] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable dram alternative. *SIGARCH Comput. Archit. News*, 37:2–13, June 2009.

[5] Shu Lin and Daniel J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[6] Moinuddin K. Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 14–23, New York, NY, USA, 2009. ACM.

[7] Stuart Schechter, Gabriel H. Loh, Karin Straus, and Doug Burger. Use ecp, not ecc, for hard failures in resistive memories. *SIGARCH Comput. Archit. News*, 38:141–152, June 2010.

[8] Nak Hee Seong, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, and Hsien-Hsin S. Lee. Safer: Stuck-at-fault error recovery for memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 115–124, Washington, DC, USA, 2010. IEEE Computer Society.

[9] International Technology Roadmap for Semiconductor. Process integration, devices, and structures - 2011 Edition. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011PIDS.pdf, 2011. Web Site. Accessed: 17-Mar-2013.

[10] Doe Hyun Yoon, N. Muralimanohar, Jichuan Chang, P. Ranganathan, N.P. Jouppi, and M. Erez. Free-p: Protecting non-volatile memory against both hard and soft errors. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 466–477, feb. 2011.

A Examples of Hamming code coverage

Table 3: Covering of Hamming(71,64)-code. At the first line we have the bit positions in the codeword. Starting from LSB to MSB. The second line discriminates data bits and parity bits, d refers to data bit and h_j refers to parity bit that covers all the bits whose the position in binary word has the $([\log_2 j] + 1)$-th bit equal to 1.

Bit position in the codeword		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bits discrimination		h_1	h_2	d_1	h_4	d_2	d_3	d_4	h_8	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	h_{16}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	d_{26}
Covering of the parity bits	h_1	•		•		•		•		•		•		•		•		•		•		•		•		•		•		•	
	h_2		•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•
	h_4				•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•
	h_8								•	•	•	•	•	•	•	•									•	•	•	•	•	•	•	•
	h_{16}																•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	h_{32}																															
	h_{64}																															

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
h_{32}	d_{27}	d_{28}	d_{29}	d_{30}	d_{31}	d_{32}	d_{33}	d_{34}	d_{35}	d_{36}	d_{37}	d_{38}	d_{39}	d_{40}	d_{41}	d_{42}	d_{43}	d_{44}	d_{45}	d_{46}	d_{47}	d_{48}	d_{49}	d_{50}	d_{51}	d_{52}	d_{53}	d_{54}	d_{55}	d_{56}	d_{57}	h_{64}	d_{58}	d_{59}	d_{60}	d_{61}	d_{62}	d_{63}	d_{64}
	•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•	
		•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•
				•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•
								•	•	•	•	•	•	•	•									•	•	•	•	•	•	•	•								
																•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•								
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•								
																																•	•	•	•	•	•	•	•

B Demonstrations

B.1 Parity Coverage Bits Equation

Property B.1 *Defining k, $k \in \mathbb{Z}$ and $f, f \in \mathbb{R} \mid 0 \leq f < 1$. Thus:*

$$\lceil k - f \rceil = k$$

Demonstration. By the definition, $\lceil k \rceil = k$ and $\lceil k - 1 \rceil = k - 1$ because $k - 1 \in \mathbb{Z}$. Thus, $\lceil k \rceil \geq \lceil k - f \rceil \geq \lceil k - 1 \rceil$. Namely, $k \geq \lceil k - f \rceil \geq k - 1$. As $f < 1$, the greatest integer for $k - f$ é k. Hence, $\lceil k - f \rceil = k$. □

We can show that (6) is correct to a regular (n, k) Hamming code. As argued before, every parity bit covers 2^{m-1} bits including itself, or $2^{m-1} - 1$ excluding itself.

Proposition B.1 *Given the equations $q_h(j) = k_h(j) + \max(0, n - (j-1) - 2 \cdot k_h(j)) - 1$ and $k_h(j) = j \cdot (\lceil \frac{n-(j-1)}{j} \rceil - \lceil \frac{n-(j-1)}{2 \cdot j} \rceil)$, for a regular Hamming code with codeword length of $n = 2^m - 1$ bits, k data bits and $m = n - k$ code bits, for every i-th code bit, at the j-th codeword bit position, where $j = 2^{i-1}$, $q_h(2^{i-1}) = 2^{m-1} - 1$.*

Demonstration. $k_h(j)$ for $j = 2^{i-1}$ and $n = 2^m - 1$ is

$$\begin{aligned}
k_h(2^{i-1}) &= 2^{i-1} \cdot \left(\left\lceil \frac{2^m - 1 - (2^{i-1} - 1)}{2^{i-1}} \right\rceil - \left\lceil \frac{2^m - 1 - (2^{i-1} - 1)}{2 \cdot 2^{i-1}} \right\rceil \right) \\
&= 2^{i-1} \cdot \left(\left\lceil \frac{2^m \cancel{-1} - 2^{i-1} \cancel{+1}}{2^{i-1}} \right\rceil - \left\lceil \frac{2^m \cancel{-1} - 2^{i-1} \cancel{+1}}{2^i} \right\rceil \right) \\
&= 2^{i-1} \cdot \left(\left\lceil \left(2^m - 2^{i-1}\right) \cdot 2^{-i+1} \right\rceil - \left\lceil \left(2^m - 2^{i-1}\right) \cdot 2^{-i} \right\rceil \right) \\
&= 2^{i-1} \cdot \left(\left\lceil 2^{m-i+1} - 1 \right\rceil - \left\lceil 2^{m-i} - 2^{-1} \right\rceil \right)
\end{aligned}$$

Knowing that 2^{m-i+1} is integer, $\lceil 2^{m-i+1} - 1 \rceil = 2^{m-i+1} - 1$. By the property B.1, $\lceil 2^{m-i} - 2^{-1} \rceil = 2^{m-i}$. Hence,

$$\begin{aligned}
k_h(2^{i-1}) &= 2^{i-1} \cdot \left(2^{m-i+1} - 1 - 2^{m-i} \right) \\
&= 2^m - 2^{i-1} - 2^{m-1}
\end{aligned}$$

Resulting in:

$$
\begin{aligned}
q_h(2^{i-1}) &= k_h(2^{i-1}) + \max\left(0, n - (2^{i-1}-1) - 2\cdot k_h(2^{i-1})\right) - 1 \\
&= k_h(2^{i-1}) + \max\left(0, 2^m - 1 - (2^{i-1}-1) - 2\cdot k_h(2^{i-1})\right) - 1 \\
&= 2^m - 2^{i-1} - 2^{m-1} + \max\left(0, 2^m - 1 - (2^{i-1}-1) - 2\cdot (2^m - 2^{i-1} - 2^{m-1})\right) - 1 \\
&= 2^m - 2^{i-1} - 2^{m-1} + \max\left(0, 2^m \cancel{-1} - 2^{i-1} \cancel{+1} - 2^{m+1} + 2^i + 2^m\right) - 1 \\
&= 2^m - 2^{i-1} - 2^{m-1} + \max\left(0, 2^m + 2^m - 2^{m+1} - 2^{i-1} + 2^i\right) - 1 \\
&= 2^m - 2^{i-1} - 2^{m-1} + \max\left(0, 2^m(1+1-2) + 2^i(1-2^{-1})\right) - 1 \\
&= 2^m - 2^{i-1} - 2^{m-1} + \max\left(0, 2^{i-1})\right) - 1 \\
&= 2^m - \cancel{2^{i-1}} - 2^{m-1} + \cancel{2^{i-1}} - 1 \\
&= 2^m - 2^{m-1} - 1 \\
&= 2^m(1-2^{-1}) - 1 \\
&= 2^{m-1} - 1
\end{aligned}
$$

□

B.2 Bit proportion of a Binary Division

Property B.2

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Stifel's relation.

Property B.3

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Easily show by Newton's binomial.

Definition B.1 *$a \oplus b$ is defined as the operation exclusive-or between a and b.*

Definition B.2 *$\bigoplus_{a \in A} a$ is defined as the operation exclusive-or among all elements that belong to the set A. Namely, $\bigoplus_{a \in A} a = \bigoplus_{i=1}^{|A|} a_i = a_1 \oplus a_2 \oplus \cdots \oplus a_{|A|},\ a_i \in A$.*

Definition B.3 *Let B be a set of n bits b_i, $0 \leq i < n$ and $b_i \in \{0,1\}$. Let $\mathcal{P}(B)$ be the power set of B and let be $P \in \mathcal{P}(B)$. We define $\oplus$-power function as*

$$f_{\oplus}(P) = \begin{cases} 0, & \text{se } P = \{\} \\ \bigoplus_{a \in P} a, & \text{otherwise.} \end{cases}$$

We want to show the following statement:

Proposition B.2 *Let B be a set of n bits b_i, $0 \leq i < n$ and $b_i = \{0,1\}$, with $k \geq 1$ bits 1. Let $\mathcal{P}(B)$ be the power set of B and let $f_{\oplus}$ be $\oplus$-power function. Then, the sum of the application of $f_{\oplus}$ over the set $\mathcal{P}(B)$ has half of its cardinality. That is,*

$$\sum_{P \in \mathcal{P}(B)} f_{\oplus}(P) = \frac{|\mathcal{P}(B)|}{2} = 2^{n-1}$$

Demonstration. Let $\pi_1(B)$ be the subset of B containing all k bits b_i of B that represents 1. Let π_0 be the subset of B containing all the $n-k$ bits B that represents 0. Let be $P_i \in \mathcal{P}(B),\ 0 \leq i < |\mathcal{P}(B)|$. Without loss of generality, assume $P_0 = \{\},\ P_1 = \{b_0\},\ P_2 = \{b_1\},\ \ldots,\ P_n = \{b_{n-1}\},\ P_{n+1} = \{b_0, b_1\},\ \ldots,\ P_{2^n-1} = \{b_0, b_1, b_2, \ldots, b_{n-1}\}$. We get $f_{\oplus}(P_0) = 0,\ f_{\oplus}(P_1) = b_0,\ \ldots,\ f_{\oplus}(P_{n+1}) = b_0 \oplus b_1,\ \ldots,\ f_{\oplus}(P_{2^n-1}) = b_0 \oplus b_1 \oplus \cdots \oplus b_{n-1}$.

Among $f_{\oplus}(P_1)$, $f_{\oplus}(P_2)$, ..., $f_{\oplus}(P_n)$, we know that k (or $\binom{k}{1}$) of that operations results 1. Among $f_{\oplus}(P_{n+1})$ and $f_{\oplus}(P_{n+\binom{n}{2}})$, namely, among the results of applying $f_{\oplus}$ in the subsets of $\mathcal{P}(B)$ composed by two elements in B, the results 1 are provide by the subsets composed by one element of π_1 and one of π_0, because we know that is necessary a odd number of bits that represents 1 for the application of $f_{\oplus}$ results 1. Hence, the number of results 1 is $\binom{k}{1} \cdot \binom{n-k}{1}$ (how many ways is possible to select a element of π_1 and one of π_0).

In the same way, the number of results 1 provide by applying $f_{\oplus}$ in the subsets of $\mathcal{P}(B)$ composed by three elements of B, is the number of subsets which there is one element of π_1 and there are two elements of π_0, or the three elements are from π_1. That is, $\binom{k}{1}\cdot\binom{n-k}{2}+\binom{k}{3}$.

Now, supposing that $k \equiv 1(\text{mod } 2)$. The counting of results of applying $f_{\oplus}$ over $\mathcal{P}(B)$ goes as following:

$$
\begin{aligned}
&\binom{k}{1}+\\
&\binom{k}{1}\cdot\binom{n-k}{1}+\\
&\binom{k}{1}\cdot\binom{n-k}{2}+\binom{k}{3}+\\
&\binom{k}{1}\cdot\binom{n-k}{3}+\binom{k}{3}\cdot\binom{n-k}{1}+\\
&\vdots\\
&\binom{k}{1}\cdot\binom{n-k}{n-k}+\binom{k}{3}\cdot\binom{n-k}{n-k-2}+\cdots+\\
&\binom{k}{3}\cdot\binom{n-k}{n-k-1}+\binom{k}{5}\cdot\binom{n-k}{n-k-3}+\cdots+\\
&\binom{k}{3}\cdot\binom{n-k}{n-k}+\binom{k}{5}\cdot\binom{n-k}{n-k-2}+\cdots+\\
&\vdots\\
&\binom{k}{k-2}\cdot\binom{n-k}{n-k}+\binom{k}{k}\cdot\binom{n-k}{n-k-2}+\\
&\binom{k}{k}\cdot\binom{n-k}{n-k-1}+\\
&\binom{k}{k}\cdot\binom{n-k}{n-k}+
\end{aligned}
$$

It is possible to group the sum as

$$
\begin{aligned}
&\binom{k}{1}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right]+\binom{k}{3}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\right.\\
&\left.\cdots+\binom{n-k}{n-k}\right]+\cdots+\binom{k}{k}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right],
\end{aligned}
$$

In another way

$$
\begin{aligned}
&\binom{k}{1}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right]+\binom{k}{3}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\right.
\end{aligned}
$$

$$\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\Bigg]+\cdots+\binom{k}{k}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right].$$

Using B.3,

$$\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}=\sum_{i=0}^{n-k}\binom{n-k}{i}=2^{n-k},$$

Therefore, we get

$$2^{n-k}\cdot\left[\binom{k}{1}+\binom{k}{3}+\cdots+\binom{k}{k}\right]. \tag{16}$$

Now, using B.2, we can replace $\binom{k}{1}$ by $\binom{k-1}{0}+\binom{k-1}{1}$, $\binom{k}{3}$ by $\binom{k-1}{2}+\binom{k-1}{3}$, and so goes until $\binom{k}{k-2}$ by $\binom{k-1}{k-3}+\binom{k-1}{k-2}$ and $\binom{k}{k}$ by $\binom{k-1}{k-1}$. Finally,

$$2^{n-k}\cdot\left[\binom{k-1}{0}+\binom{k-1}{1}+\binom{k-1}{2}+\binom{k-1}{3}+\cdots+\binom{k-1}{k-3}+\binom{k-1}{k-2}+\binom{k-1}{k-1}\right],$$

Again, by B.3 we get

$$\binom{k-1}{0}+\binom{k-1}{1}+\binom{k-1}{2}+\cdots+\binom{k-1}{k-1}=\sum_{i=0}^{k-1}\binom{k-1}{i}=2^{k-1}.$$

(16) can be written as follows:

$$2^{n-k}\cdot 2^{k-1}=2^{n-1}.$$

Now, supposing that $k\equiv 0(\text{mod }2)$. The element $\binom{k}{k}$ is not anymore a even combination of elements of $\pi_1(B)$. Thus,

$$2^{n-k}\cdot\left[\binom{k}{1}+\binom{k}{3}+\cdots+\binom{k}{k-1}\right]. \tag{17}$$

Using B.2, we replace $\binom{k}{1}$ by $\binom{k-1}{0}+\binom{k-1}{1}$, $\binom{k}{3}$ by $\binom{k-1}{2}+\binom{k-1}{3}$, and so goes until $\binom{k}{k-3}$ by $\binom{k-1}{k-4}+\binom{k-1}{k-3}$ and $\binom{k}{k-1}$ by $\binom{k-1}{k-2}$ e $\binom{k-1}{k-1}$. Hence,

$$2^{n-k}\cdot\left[\binom{k-1}{0}+\binom{k-1}{1}+\binom{k-1}{2}+\binom{k-1}{3}+\cdots+\binom{k-1}{k-3}+\binom{k-1}{k-2}+\binom{k-1}{k-1}\right].$$

We should observe that the number of combination summed does not differ with the parity of k. Being k odd, the Stifel's relation is not used in the last element of the sum, because $\binom{k}{k}$ cannot be replaced by $\binom{k-1}{k-1}+\binom{k-1}{k}$. Finally, by B.2, (17) can be written as

$$2^{n-k}\cdot 2^{k-1}=2^{n-1}.$$

□

Corollary B.2.1 *Half of the results of $f_{\oplus}(P_i)$, $P_i \in \mathcal{P}(B),\ 0 \leq i < |\mathcal{P}(B)|$, are 1 and, consequently, half are 0.*

Demonstration. By contradiction. Supposing that the number of results 1 is greater than the number of results 0. We know that $\mathcal{P}(B)$ has 2^n elements and applying $f_{\oplus}$ over $\mathcal{P}(B)$ the sum results in 2^{n-1}. If there is more results 1 than 0, then the sum of results 1 have to be greater than 2^{n-1}, but it is not possible. In the same way, if there is less results 1 than 0, the sum should compute less than 2^{n-1}, but again is not possible. So the number of results 1 is equal to the number of results 0. □