Error-Correcting WOM Codes: Concatenation and Joint Design

Amit Solomon and Yuval Cassuto, *Senior Member, IEEE*

Abstract

We construct error-correcting WOM (write-once memory) codes that guarantee correction of any specified number of errors in q-level memories. The constructions use suitably designed short q-ary WOM codes and concatenate them with outer error-correcting codes over different alphabets, using suitably designed mappings. With a new storage-efficiency measure we call EC-rate, we show that for common error types the codes save redundancy and implementation complexity over straightforward concatenation. In addition to constructions for guaranteed error correction, we extend the error-correcting WOM scheme to binary multi-level coding for random errors, and to soft-decision decoding without using higher-precision readout.

Index Terms

WOM codes, error-correcting WOM codes, codes for memories, concatenated codes, multi-level coding, coded modulation.

I. Introduction

Flash-based non-volatile memories (NVM) are the storage media of choice in most modern information applications, thanks to their fast access and growing densities. However, the Flash technology suffers from the major impediment of not being able to update data in-place. Because removing charges from memory cells cannot be done at a fine granularity, it is not possible to update written data without first erasing a very large data unit. As a result, write performance is degraded and the wear of cells is accelerated. It has been demonstrated and recognized [18], [22] that *WOM (write-once memory) codes* [19] hold promise to mitigate this access limitation by allowing to update the logical data multiple times without need to physically remove charges from the cells. While WOM codes can support a more flexible write access to Flash, a concern is raised about their effect on data reliability. With WOM codes, cells are written multiple times between erases, and are accessed in a less predictable order than when written only once without WOM. These two effects may increase the severity of inter-cell interference, and degrade reliability if not properly addressed.

Our objective in this paper is to facilitate the reliability of WOM codes by making them resilient to errors from noise and interference. Our method is to combine WOM codes with error-correcting (EC) codes to get guaranteed error correction with flexible parameters. In general combining WOM codes with EC codes is a non-trivial task because the EC encoder may not respect the WOM constraints, and the WOM decoder may cause error propagation affecting a large number of EC symbols. While there have been some works addressing EC+WOM combination [24], [21], [15], the problem is yet to be adequately solved for practical deployment in Flash. In particular, prior to our work no scheme provides guaranteed correction of τ errors, for an arbitrary τ. One exception is a construction in [21] that lifts triple-error correcting WOM codes to general τ, but requires replicating the same WOM codeword a number of times linear in τ.

Our contribution in this paper is a new scheme to combine WOM with EC codes. We use short ($n = 2$) WOM codes over the memory's q-ary alphabet (q is the number of levels supported by the physical device), and concatenate them with outer EC codes of any length N. Our contributions divide into two parts: the first part (Section IV) addresses code constructions for guaranteed error correction with worst-case errors, while the second part (Section V) develops additional tools for random error channels. In the first part we propose multiple code constructions for guaranteed τ-error correction (for any τ), which improve storage rates over straightforward concatenation for errors common in Flash (e.g. magnitude-1 or asymmetric magnitude-1). Our construction method is to jointly design the EC and WOM codes, such that the WOM redundancy anyhow expended for rewriting is designed to also contribute to error correction. By that, our work develops concrete formal constructions building on the notion that WOM codes can have intrinsic error resilience, previously observed in [17] and further developed in [13]. The main observation leading to the constructions' good correction capabilities is that the WOM alphabet can be decomposed and mapped to multiple smaller alphabets, and on each alphabet different EC capabilities can be prescribed for the outer EC code. Each of the constructions is specified with four components: 1) the WOM code, 2) the alphabet decomposition and mapping, 3) the EC code, and 4) the construction combining 1-3. To evaluate and compare our constructions, we define the *EC-rate* as a new rate measure that captures the residual redundancy added to a WOM code for error correction. Code families with concrete parameters are obtained by evaluating the EC-rate on parameters of the ubiquitously used BCH codes. In addition to concrete advantage in EC-rates, the lower-alphabet EC codes used in our constructions reduce the implementation complexity over straightforward concatenation. In the second part we complement the code constructions with additional coding-theoretic

Amit Solomon and Yuval Cassuto are with the Viterbi Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa Israel (emails: samitsolomon@gmail.com, ycassuto@ee.technion.ac.il

This work was supported in part by the Israel Science Foundation, and in part by the US-Israel Binational Science Foundation.

Part of the results in the paper were presented at the IEEE International Symposium on Information Theory, June 2018.

tools that can improve error-correction performance with random errors. Firstly, we develop a *multi-level coding* framework for which WOM codes are constructed to work with binary-only EC codes (for low implementation complexity). The WOM alphabet is mapped hierarchically to multiple bits, and information from the WOM decoding function is used to feed an error+erasure multi-stage decoder. Analysis using decoding-error probability bound shows the good performance of the WOM codes constructed especially for multi-level coding. In addition, it shows the significant advantage in decoding performance of decoders that know the current write number of the WOM code. Secondly, we extend the EC-WOM framework to *soft-decision decoding*, where the soft decoder inputs come not from costly higher-precision readout, but from the reliability information provided by the inner WOM decoder. We show that for additive Gaussian noise channels the ordering of symbol reliabilities based on the WOM decoding function has a nice structure.

Our focus in the paper is on $q = 8$-ary WOM codes corresponding to the ubiquitous TLC Flash technology, and offering $t = 4$ guaranteed writes. But the proposed tools and techniques can be extended to other q and t values. While the prior work on error-correcting WOM codes focused on binary codes, our approach leverages the information-theoretic advantage of using higher q WOM codes [7]. This approach can potentially be used to add error correction capabilities to other known q-ary WOM codes, such as those in [2], [10], [11], [16]. Considering that most EC-WOM alternatives are binary, the specific WOM parameters considered in the paper are attractive to use because their rate (without error correction) is within about 10% from the fixed-rate binary WOM capacity [12].

II. Preliminaries

We start by including definitions regarding WOM codes; then we add definitions addressing error correction by WOM codes.

A. WOM codes

A WOM code is our basic object of study in this paper, and we focus on the generalization of the WOM coding model to q-ary alphabets, with $q > 2$. We give the general definition of a WOM code up to restricting it to be *fixed-rate*, that is, the same number of bits is written in each of its writes.

Definition 1. *A (n, q, t, M) (fixed-rate) **WOM code** is a code applied to a size n block of q-ary cells, and guaranteeing t writes of input size M each, while the cell levels do not decrease.*

A WOM code is specified through a pair of functions: the *decoding* and *update* functions.

Definition 2. *The decoding function is defined as $\psi : \{0, \ldots, q-1\}^n \to \{0, \ldots, M-1\}$, which maps the current levels of the n cells to the data input in the most recent write. The update function is defined as $\mu : \{0, \ldots, q-1\}^n \times \{0, \ldots, M-1\} \to \{0, \ldots, q-1\}^n$, which specifies the new cell levels as a function of the current cell levels and the new data value at the input. By the WOM requirement, the i-th cell level output by μ cannot be lower than the i-th cell level in the input, for each i.*

Definition 2 of the WOM-code functions assumes the special case that the function outputs do not depend on the current write number. Later in the paper (Definition 11), we generalize the definition of the decoding function ψ to also depend on the write number.

Definition 3. *The code's **physical state** is defined as the n q-ary levels to which the cells are currently programmed. The code's **logical state** is the data element from $\{0, \ldots, M-1\}$ returned by ψ on the current physical state.*

A convenient way to specify the WOM decoding function is through an n-dimensional matrix of physical states $\{0, \ldots, q-1\}^n$, where at each position the matrix holds the logical state in $\{0, \ldots, M-1\}$ corresponding to the output of the decoding function for that physical state. The update function specifies the target physical state given the current physical state and the input logical value.

Example 1. *Let us consider two WOM-code examples. In Figure 1a we have the decoding function of a $(n = 2, q = 7, t = 3, M = 8)$ WOM code constructed by Construction 3 in [5]. This code is applied on a pair of $q = 7$-level memory cells, enabling $t = 3$ guaranteed writes of size $M = 8$ each. In Figure 1b we have a different $(n = 2, q = 7, t = 3, M = 8)$ code, offering the exact same parameters. Let us assume that for the $t = 3$ writes we want to write the logical values 7, 6 and 2 using the WOM code of Figure 1 (a). For the first write the logical state is 7 and the physical state is $(2, 1)$. When updating the logical state to 6, the physical state becomes $(2, 4)$. For the third write of 2, the physical state becomes $(2, 6)$. The codes of Fig. 1 (a),(b) are equivalent in terms of their specified re-write performance (the same guaranteed t and M using the same coding resources n and q). However, the codes may differ significantly in performance beyond the specified parameters. For example, the code of Figure 1 (a) can be extended to 4 writes when moving from $q = 7$ to $q = 8$, while the code of Figure 1 (b) cannot.*

In the remainder of the paper, we focus on WOM constructions that maximize the *error-correction performance* within their class of (ν, q, t, m) codes. It will be shown that two codes with the same re-write parameters can have very different performance with respect to correcting memory errors. The error-correction performance of a WOM code is now defined.

Definition 4. *A WOM code with parameters (ν, q, t, m) is called τ-**error correcting** if it can correct any error combination in up to τ (out of the ν) q-ary code symbols.*

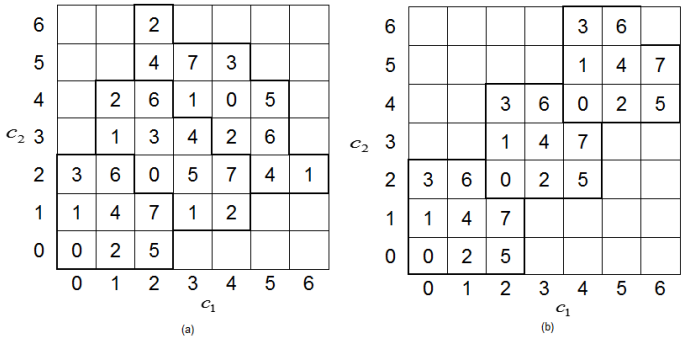
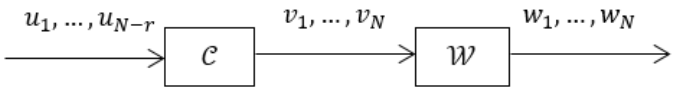


Figure 1: Sample $n = 2$ WOM constructions (from [5]). (a) - Decoding function ψ for a $(2, 7, 3, 8)$ code. (b) - Decoding function ψ for another $(2, 7, 3, 8)$ code. Physical states are represented by (c_1, c_2) and logical states are labeled inside each matrix position.

III. EC-WOM Concatenation

The route to error-correcting WOM codes we pursue in this paper is through *concatenation*. We start in this section with the most straightforward concatenated construction using an inner WOM code and an outer error-correcting (EC) code. Such constructions are called *EC-WOM* codes. Let $\mathcal{W}$ be a (n, q, t, M) WOM code. Denote an EC code $\mathcal{C}$ by $\mathcal{C}_A[N, N-r]$ if it is defined over an alphabet of size A (power of a prime), it has length N, and r redundant A-ary symbols. A pictorial illustration of concatenating the EC code $\mathcal{C}$ with the WOM code $\mathcal{W}$ is given in Figure 2.



.

Figure 2: EC-WOM concatenation. Note that $u_i, v_j \in GF(A)$, whereas $w_i \in \{0, ..., q-1\}^n$

We start with the simplest construction using an inner WOM code and an outer EC code.

Construction 1. *Given a (n, q, t, M) WOM code $\mathcal{W}$ and a $\mathcal{C}_M[N, N-r]$ EC code $\mathcal{C}$, the concatenated code $\mathcal{CW}_0$ is the $\left(nN, q, t, M^{N-r}\right)$ WOM code obtained by taking N copies of $\mathcal{W}$ and inputing to each a symbol of $\mathcal{C}$. In addition, if $\mathcal{C}$ can correct τ M-ary errors, then $\mathcal{CW}_0$ is a τ-error correcting WOM code.*

The concatenation of Construction 1 is extremely simple but it requires to use an EC code over an alphabet as large as the input size of the WOM code. An EC code with large alphabet size implies a high cost of redundancy per each correctable error. So if n is even moderately large the scheme is not attractive because M must grow to lower the WOM redundancy, and with M grows the EC redundancy for the same number of physical q-ary errors. For that reason we limit ourselves to a small n in our proposed error-correcting WOM codes, and in subsequent constructions we show how to further reduce the redundancy compared to Construction 1. It turns out that $n = 2$ is small enough to not suffer from high EC redundancy, and large enough to get good redundancy WOM codes.

A. EC codes and BCH redundancy

We now define a measure of redundancy that is suitable for evaluating the concatenated codes. To compare between different EC-WOM concatenated constructions we use the *EC-rate* of the code, defined next.

Definition 5. *Let $\mathcal{CW}$ be a $\left(nN, q, t, M^{N-r_{\text{eff}}}\right)$ WOM code with some error-correction capability, which is constructed with a (n, q, t, M) inner WOM code. We define the* ***EC-rate*** *of $\mathcal{CW}$ to be*

$$1 - \frac{r_{\text{eff}}}{N}. \tag{1}$$

r_{eff} is the effective redundancy of the EC-code in units of M-ary symbols.

The EC-rate of the code reflects its efficiency to correct errors, and ignoring its redundancy as a WOM code used for re-writing. The EC-rate definition is convenient for comparing between concatenation-based constructions that add certain EC capabilities to WOM codes with a given set of parameters (n, q, t, M).

The constructions we provide in the paper work with general EC codes, but for convenience we present them with BCH EC codes [3] [14]. Beyond their well-known effectiveness in a variety of applications (including non-volatile memories), BCH

codes are especially useful for this study because they can be defined over different alphabet sizes, and there are simple expressions for their redundancy that can capture and compare their performance.

Definition 6. *For alphabet size A, code length N, and designed minimum distance d, define the* ***BCH (approximate) redundancy*** *as*

$$\frac{A-1}{A} \cdot (d-2) \cdot \log_A N \ \ [A-\text{ary symbols}]. \tag{2}$$

The BCH redundancy is the approximate number of parity symbols in the codeword, given in units of A-ary symbols. This definition of redundancy comes from a known approximation of the redundancy of primitive A-ary BCH codes for large N, when A is a power of a prime [23]. We can see that fixing N and d the BCH redundancy as defined decreases with A as $(A-1)/(A \log A)$, but note that when fixing the units (e.g. to bits) the amount of actual redundancy in the code grows with A as $(A-1)/A$.

When using Construction 1 with a BCH code we get the following result combining Definitions 6 and 5.

Example 2. *Take a $(n=2, q=8, t=4, M=8)$ WOM code $\mathcal{W}$ and concatenate it with an outer τ-error correcting (design minimum distance $d=2\tau+1$) BCH code $\mathcal{C}=\mathcal{C}_8[N, N-r]$ using Construction 1. Then we get the $\left(2N, 8, 4, 8^{N-r}\right)$ WOM code $\mathcal{CW}_0$ that is τ-error correcting.*

The outer code $\mathcal{C}$ used in the construction of Example 2 has BCH redundancy

$$\frac{7}{8} \cdot (2\tau-1) \cdot \log_8 N \ \ [8-\text{ary symbols}].$$

Proposition 1. *A τ-error correcting WOM code $\mathcal{CW}_0$ obtained by Construction 1 with an M-ary BCH code $\mathcal{C}$ has EC-rate*

$$1 - \frac{\frac{M-1}{M}(2\tau-1)\log_M N}{N}, \tag{3}$$

and one may recognize the BCH redundancy of the M-ary BCH code as r_{eff} at the numerator of the right term in (3). Specifically for the parameters of Example 2 we get that Construction 1 attains the EC-rate of

$$1 - \frac{7}{8} \cdot \frac{(2\tau-1)\log_8 N}{N}. \tag{4}$$

B. TLC WOM Codes

In the rest of the paper we focus our attention on concatenated constructions using inner WOM codes with the specific parameters $(n=2, q=8, t=4, M=8)$. We do so for two main reasons: 1) fixing the WOM parameters used by the constructions better conveys the qualities of the constructions and their differences, and 2) these particular parameters are convenient to use in a practical non-volatile memory setup ($q=8$ corresponds to the current state-of-the-art multi-level technology, $M=8$ is an integer power of 2 and $n=2$ is small enough for the concatenation to be efficient in redundancy). WOM codes with these parameters are called here *TLC WOM codes*, where TLC is how the non-volatile memory community refers to 8-level memories – here both the physical (q) and the logical (M) number of levels is 8. TLC WOM codes are optimal in the sense that $t=4$ is the maximum possible given $n=2, q=8, M=8$, and also $q=8$ is the minimum possible given $n=2, t=4, M=8$. In addition, the WOM *sum-rate* of TLC WOM codes is within $\sim 10\%$ from the fixed-rate binary WOM capacity [12], which is only attainable with large n and high complexity. It is important to note that the constructions work for more general parameters beyond TLC WOM codes. For example, fixing $n=2, M=8$ and growing t as a function of q is a simple extension building on known art [5].

IV. EC-WOM Concatenated Constructions

In this section we construct concatenated EC-WOM codes that improve over the basic Construction 1 from the previous section. Construction 1 offers EC-WOM codes by using standard M-ary error-correcting codes for symmetric errors. The main drawback of this construction is that in interesting WOM codes M is not too small, which implies high redundancy and complex decoding. In the constructions of this section we show that for the error types common in non-volatile memories, guaranteed error correction can be attained with less redundancy and lower decoding complexity. In particular, the error-correcting codes used in the concatenation will be over alphabets smaller than M, and often the binary alphabet. The key method is to carefully design the inner WOM code such that guaranteed error correction is attained with a more efficient outer error-correcting code.

The error model considered in Definition 4 covers any possible error affecting a code symbol. In non-volatile memories with relatively large number of levels q (e.g. $q=8$), protecting against general q-ary errors is accepted to be a wasteful overkill. Instead, specific dominant error types are addressed, which are more limited in the symbol transitions they cover. We follow the same wisdom here, and refine the error-correction characterization for EC-WOM codes. We consider the following error types defined over the alphabet $Q \triangleq \{0, 1, \ldots, q-1\}$.

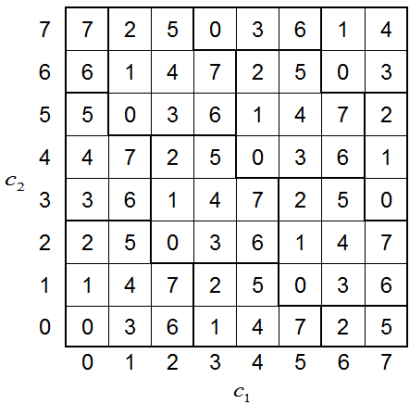


Figure 3: Decoding function of the tiling-based $\mathcal{W}(2, 8, 4, 8)$ code taken from [5].

Definition 7. *Let $c \in Q$ be the symbol written to a memory cell. The cell suffers a **mag-1** error if the read symbol is $c' \in Q$ such that $|c' - c| = 1$.*

Mag-1 errors allow transitions of one level either upward or downward from the correct symbol. We similarly define the asymmetric version Amag-1.

Definition 8. *Let $c \in Q$ be the symbol written to a memory cell. The cell suffers an **Amag-1** error if the read symbol is $c' \in Q$ such that $c' - c = 1$.*

Amag-1 errors allow transitions of one level in the *upward* direction only. Amag-1 errors are common in non-volatile memories such as Flash due to inter-cell interference (ICI): adding charges to a cell due to writing to a neighboring cell.

We can plug in the mag-1 and Amag-1 error types into the definition of τ-error correction (Definition 4) as follows.

Definition 9. *A WOM code with parameters (ν, q, t, m) is called τ **mag-1 error correcting** / τ **Amag-1 error correcting** if it can correct any combination of* mag-1/Amag-1 *errors, respectively, in up to τ (out of the ν) q-ary code symbols.*

Note that in particular a τ mag-1 error correcting code is also τ Amag-1 error correcting, and a τ-error correcting code (from Definition 4) is both τ mag-1 error correcting and τ Amag-1 error correcting.

To efficiently correct mag-1 and Amag-1 errors in WOM codes, in the sequel we specify the constructions in three steps: 1) specification of the inner WOM code, 2) mapping the M-ary logical alphabet of the WOM code $\{0, \ldots, M-1\}$ to a more structured alphabet, and 3) prescribing EC-codes over the structured alphabet with parameters that guarantee correcting τ errors. This construction method has the flavor of *generalized concatenation* (GC) and *coded modulation* (CM) [4] found very useful in previous applications[1]. Unlike the simple Construction 1, in the improved constructions that follow we will need to go beyond specifying just the *parameters* of the inner WOM code, to designing new WOM codes with extra properties that are needed for the concatenation to work.

A. EC-WOM construction for mag-1 errors

1) Tiling WOM code: The inner WOM code we use in this construction is the known tiling-based $(n = 2, q = 8, t = 4, M = 8)$ code from [5], depicted in Figure 3. The figure shows the decoding function of the code, and we omit here the specification of the update function because in this case it is immaterial for proving the error-correction properties (this is not true in general, and in subsequent constructions we *will* rely on the update function for error correction).

2) Mapping 8-ary to a product of 4-ary and binary: The concatenation requires introducing structure to the logical $M = 8$-ary alphabet as follows. We map each number $a \in \{0, \ldots, 7\}$ to its mixed radix representation (ah, al), where the upper symbol $ah \in \{0, 1, 2, 3\}$ and the lower symbol $al \in \{0, 1\}$ satisfy $2ah + al = a$, and are unique. In Figure 4 we apply this mapping to the tiling WOM code of Figure 3, where for clarity we map $\{0, 1, 2, 3\}$ to $\{a, b, c, d\}$. It can be seen in Figure 4b that tiling the space with the tile of Figure 4a gives the property that physical states adjacent horizontally or vertically have the opposite binary value in their lower symbol. This property will be used by the construction to combat mag-1 errors, which correspond horizontal and vertical transitions between adjacent physical states.

3) Outer 4-ary and binary EC codes: For the 4-ary upper symbols we take the $\mathcal{C}_4[N, N - r_1]$ BCH code with design distance $\tau + 1$; for the binary lower symbols we take the $\mathcal{C}_2[N, N - r_2]$ BCH code with design distance $2\tau + 1$. These choices will allow us to correct up to τ errors in the binary lower symbols and up to e' errors and e erasures in the 4-ary upper symbols, for $2e' + e \leqslant \tau$.

[1]GC fits when seeing the WOM as a code, and CM is appropriate when thinking about WOM as modulation.

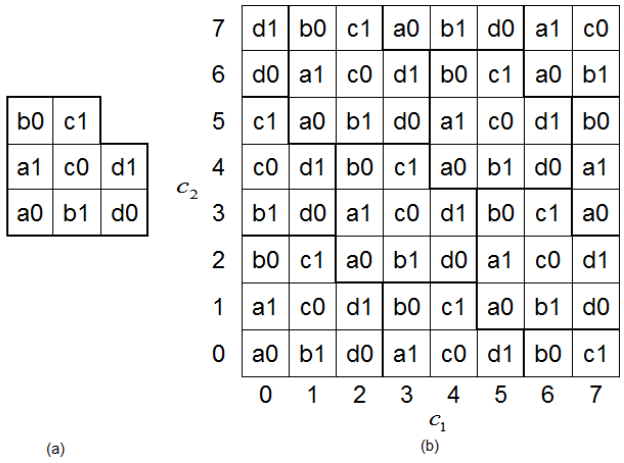


Figure 4: Mapping the 8-ary WOM logical symbols to 4-ary+binary symbols. (a): a single tile, and (b): the entire WOM code.

4) The construction:

Construction 2. *Given the* $(2, 8, 4, 8)$ *WOM code* $\mathcal{W}$ *specified with its logical mapping in Figure 4b and EC codes* $\mathcal{C}_4[N, N-r_1]$, $\mathcal{C}_2[N, N-r_2]$ *specified in Section IV-A.3, the concatenated code* $\mathcal{CW}_1$ *is the* $\left(2N, 8, 4, 4^{N-r_1} \cdot 2^{N-r_2}\right)$ *WOM code obtained by taking* N *copies of* $\mathcal{W}$ *and inputing to each a symbol of* $\mathcal{C}_4$ *to its upper symbol and a symbol of* $\mathcal{C}_2$ *to its lower symbol.*

The essential properties of Construction 2 are given in the following proposition. The proof includes the decoding algorithm.

Proposition 2. *For any* N *and* τ, *a* $\left(2N, 8, 4, 4^{N-r_1} \cdot 2^{N-r_2}\right)$ *code* $\mathcal{CW}_1$ *obtained by Construction 2 is* τ *mag-1 error correcting. Moreover,* $\mathcal{CW}_1$ *has EC-rate*

$$1 - \frac{\left[\frac{3}{4}(\tau-1) + \frac{1}{2}(2\tau-1)\right]\log_8 N}{N}. \tag{5}$$

Proof: To see that $\mathcal{CW}_1$ is τ mag-1 error correcting, define by τ_1 the number of copies of $\mathcal{W}$ that suffered a mag-1 error in exactly one of their two symbols. Similarly, define by τ_2 the number of copies of $\mathcal{W}$ that suffered mag-1 errors in both their symbols. We show that any combination with $\tau_1 + 2\tau_2 \leqslant \tau$ is correctable. We first decode the binary code $\mathcal{C}_2$. From the mapping seen in Figure 4, there will be bit errors in all positions corresponding to the τ_1 mag-1 errors affecting one symbol out of a $\mathcal{W}$ pair. Since $\tau_1 \leqslant \tau$, the decoder of $\mathcal{C}_2$ will locate and correct all these errors. Next we decode the 4-ary code $\mathcal{C}_4$, where every position with error in $\mathcal{C}_2$ is erased in the input of $\mathcal{C}_4$'s decoder. The decoder sees τ_1 erasures and τ_2 errors, and can correct them because of the restriction $\tau_1 + 2\tau_2 \leqslant \tau$ (recall that $\mathcal{C}_4$ has design minimum distance $\tau + 1$). At this point all up to τ symbols in error are recovered by $\mathcal{C}_2$ and $\mathcal{C}_4$ jointly.

To prove the EC-rate, we observe that

$$4^{N-r_1} \cdot 2^{N-r_2} = 8^{N-\frac{2}{3}r_1-\frac{1}{3}r_2}. \tag{6}$$

Hence in Definition 5 we have $r_{\text{eff}} = \frac{2}{3}r_1 + \frac{1}{3}r_2$ 8-ary symbols. Substituting in r_{eff} the BCH redundancies from (2)

$$r_1 = \frac{3}{4}(\tau-1)\log_4 N, \;\; r_2 = \frac{1}{2}(2\tau-1)\log_2 N,$$

we get (5) for the EC-rate. ∎

Comparing to Construction 1, we see that Construction 2 gives EC-rate higher by $\frac{3}{8}\frac{\log_8 N}{N}$. Consequently, if errors are predominantly mag-1 errors, with a more clever concatenation we can correct the same number of errors with less redundancy compared to the straightforward concatenation. Another important advantage of Construction 2 is in its much simpler decoding. While Construction 1 requires a BCH decoder for τ errors over the finite field GF(8), Construction 2 only needs to correct τ binary errors in C_2, and in the worst case only $\tau/2$ errors over GF(4) in C_4. Moreover, in typical decoding instances, the code C_4 will mostly need to deal with erasures, because in most error patterns very few copies of $\mathcal{W}$ suffer errors in both symbols.

B. Improved EC-WOM construction for mag-1 errors

While Construction 2 offered an improvement of the EC-rate over the basic concatenation, the improvement is quite small, and in particular the difference $\frac{3}{8}\frac{\log_8 N}{N}$ does not grow with τ. In this sub-section we aim at improving the EC-rate further, and more significantly. To get the desired improvement in EC-rate we need two ingredients: 1) a new suitably designed $(2, 8, 4, 8)$ WOM code, and 2) a slight refinement of the error model to distinguish between mag-1 errors not in the same $\mathcal{W}$ pair (more common) and those falling in the same pair (less common). Also, from now on we will rely on the fact that the decoder knows the current write count, that is, how many times (out of the t) the WOM has been written so far. This assumption is standard

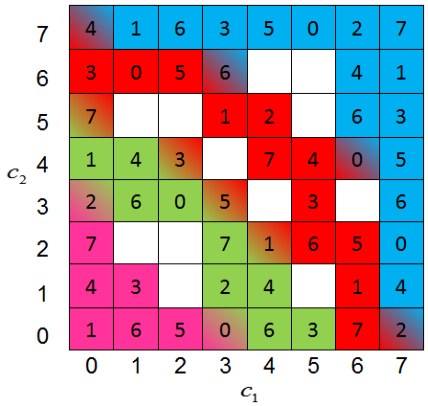


Figure 5: A more suitably designed WOM code: $\mathcal{W}_2\left(2,8,4,8\right)$. The numbers as usual specify the decoding function and the colors represent at which write(s) the physical state can be reached. Solid color represents a single write and mixed color represents two consecutive writes.

in the WOM literature, and does not incur redundancy or complexity costs when N is at least moderately large (storing the write counter is amortized over many parallel WOM blocks).

1) 3-Manhattan WOM: In order to improve the EC-rate, we now show a more suitably designed $(n=2,q=8,t=4,M=8)$ WOM code. The decoding function of the WOM code is shown in Figure 5. The physical states are colored according to which write(s) reach them: magenta, green, red, blue for writes 1-4, respectively. The physical states with mixed colors are shared between two adjacent writes; for example physical state $(0,7)$ is shared between writes 3 and 4. With the colors of the states provided, the update function can be specified in a straightforward manner. It can be verified that from any physical state in one write's color it is possible to reach any logical state in the next write's color. This property of restricting the physical state to belong to the corresponding write's color will be useful for later proving the error-correction properties of the construction.

Definition 10. *Given two physical states $\boldsymbol{x},\boldsymbol{y}$ of a (n,q,t,M) WOM code, the* ***Manhattan distance*** *between the physical states is defined as $D_1\left(\boldsymbol{x},\boldsymbol{y}\right)=\sum_{i=1}^{n}\left|x_i-y_i\right|$, where x_i,y_i are the values written to the i-th cell of $\boldsymbol{x},\boldsymbol{y}$ respectively.*

2) Mapping 8-ary to a product of 4-ary and binary: We use an 8-ary to a product of 4-ary and binary mapping, like in Construction 2 (Section IV-A.2) to map the 8-ary alphabet to a product of 4-ary and binary alphabets. This mapping gives the WOM the following property: the Manhattan distance between two physical states *in the same write* that have the same 4-ary symbol and opposite binary bit is at least 3. This property can be seen in Figure 6 showing the WOM code with the product alphabet. For example, in write 2 physical state $(3,1)$ with value $b0$ and physical state $(5,0)$ with value $b1$ are at Manhattan distance 3, and no $b1$ state is closer to $(3,1)$ in write 2 (there is a closer $b1$ state in $(1,1)$, but it does not belong to write 2.)

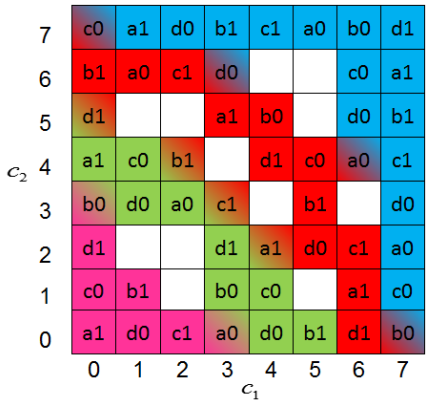


Figure 6: The code of Figure 5 after mapping 8-ary to 4-ary+binary. The code+mapping have the 3-Manhattan property used for error correction.

3) Outer 4-ary and binary EC codes: For the 4-ary symbols we take the $\mathcal{C}_4[N,N-r]$ BCH code with design distance $2\tau+1$; the binary bits are left uncoded (i.e., we use the trivial code $\mathcal{C}_2[N,N]$). These choices will allow us to correct up to τ errors in the 4-ary symbols and recover the binary bit.

4) The construction:

Construction 3. *Given the $(2, 8, 4, 8)$ WOM code $\mathcal{W}_2$ specified with its logical mapping in Figure 6 and EC codes $\mathcal{C}_4[N, N-r]$, $\mathcal{C}_2[N, N]$ specified in Section IV-B.3, the concatenated code $\mathcal{CW}_2$ is the $\left(2N, 8, 4, 4^{N-r} \cdot 2^N\right)$ WOM code obtained by taking N copies of $\mathcal{W}_2$ and inputing to each a symbol of $\mathcal{C}_4$ to its upper symbol and a symbol of $\mathcal{C}_2$ to its lower symbol.*

The concatenated EC-WOM code specified in Construction 3 has the following properties. The proof includes the decoding algorithm.

Proposition 3. *For any N and τ, a $\left(2N, 8, 4, 4^{N-r} \cdot 2^N\right)$ code $\mathcal{CW}_2$ obtained by Construction 3 is τ mag-1 error correcting, assuming no copy of $\mathcal{W}_2$ suffers two mag-1 errors. Moreover, $\mathcal{CW}_2$ has EC-rate*

$$1 - \frac{3}{4} \cdot \frac{(2\tau - 1) \cdot \log_8 N}{N}. \tag{7}$$

Proof: Suppose τ copies of $\mathcal{W}_2$ suffered a mag-1 error in exactly one of their two cells. Assuming the decoder knows the current write number, a mag-1 error resulting in a physical state outside this write's color can be detected by the decoder, and mapped to a 4-ary *erasure* in $\mathcal{C}_4$. Other mag-1 errors map to 4-ary errors in $\mathcal{C}_4$, because adjacent physical states within the same write's color have different 4-ary symbols (check in Figure 6). Thus all τ 4-ary symbols in the erroneous cells can be recovered since $\mathcal{C}_4$ is designed with distance $2\tau + 1$. To recover the lower bits of the cells with mag-1 errors we use the Manhattan-distance-3 property shown in Section IV-B.2 that guarantees that not both $x0$ and $x1$ states are adjacent to the read cell state, where x is the 4-ary symbol recovered by $\mathcal{C}_4$.

To prove the EC-rate, we note that $4^{N-r} \cdot 2^N = 8^{N-\frac{2}{3}r}$, hence in Definition 5 we need to substitute $r_{\text{eff}} = \frac{2}{3}r$. We calculate the BCH redundancy r for $\mathcal{C}_4$ from Definition 6 and get $r = \frac{3}{4}(2\tau - 1)\log_4 N$. Substituting into r_{eff} we obtain (7). ∎

Examining the EC-rate attained for the improved Construction 3, we see a significant improvement over both Construction 1 and Construction 2. The new EC-rate is higher than Construction 1 by $\frac{1}{8}\frac{(2\tau-1)\log_8 N}{N}$, now an improvement that *grows with* τ. This advantage comes at the cost of excluding error patterns with both cells in error at the same copy of $\mathcal{W}_2$ (most errors of this type are still correctable, but not all; see Figure 6 $c1$ in state $(3, 3)$ changing to $d1$ in state $(4, 4)$: the decoder will miscorrect to $c0$ in state $(5, 4)$). The uncorrectable error combinations are unlikely in a random error pattern, because when $\tau \ll N$ hitting two cells of the same $\mathcal{W}_2$ copy has a very small probability. That said, we next extend Construction 3 to also guarantee a specified number of double mag-1 errors in the same $\mathcal{W}_2$ copy. The extension is done by adding an EC-code to the binary symbols as well, instead of the trivial $\mathcal{C}_2[N, N]$ used previously in Section IV-B.3.

3') Refined outer 4-ary and binary EC codes: For correcting τ_1 single mag-1 errors (one in a $\mathcal{W}_2$ copy) and τ_2 double mag-1 errors (both in the $\mathcal{W}_2$ copy), use for the 4-ary upper symbols the code $\mathcal{C}_4[N, N - r_1]$ with design distance $2(\tau_1 + \tau_2) + 1$. For the lower binary symbols use the code $\mathcal{C}_2[N, N - r_2]$ designed with distance $2\tau_2 + 1$.

4') The refined construction:

Construction 4. *This construction is the same as Construction 3, but with its EC-codes changed to $\mathcal{C}_4[N, N - r_1]$, $\mathcal{C}_2[N, N - r_2]$ specified in Section IV-B.3'.*

Now with the refined EC-codes we have the following correction capabilities. The proof includes the decoding algorithm modified accordingly.

Proposition 4. *For any N and τ_1, τ_2, a $\left(2N, 8, 4, 4^{N-r_1} \cdot 2^{N-r_2}\right)$ code $\mathcal{CW}_2'$ obtained by Construction 4 corrects any error combination where at most τ_2 $\mathcal{W}_2$ copies suffered double mag-1 errors and at most $\tau_1 + \tau_2$ $\mathcal{W}_2$ copies suffered mag-1 errors (double or single). Moreover, $\mathcal{CW}_2'$ has EC-rate*

$$1 - \frac{3}{4} \cdot \frac{\left(2\tau_1 + \frac{10}{3}\tau_2 - \frac{5}{3}\right) \cdot \log_8 N}{N}. \tag{8}$$

Proof: As in Proposition 3, $\mathcal{C}_4$ will recover all the 4-ary symbols in $\mathcal{W}_2$ copies that suffered mag-1 errors, single or double. After setting the bits in the binary symbols according to the closest physical states to the read outputs, the decoder of $\mathcal{C}_2$ will be able to correct the at most τ_2 bit errors from double-error $\mathcal{W}_2$ copies.

To prove the EC-rate, we have $r_{\text{eff}} = \frac{2}{3}r_1 + \frac{1}{3}r_2$, and substituting the BCH redundancy for $\tau_1 + \tau_2$ in $\mathcal{C}_4$ and τ_2 in $\mathcal{C}_2$ we get

$$r_1 = \frac{3}{4}(2\tau_1 + 2\tau_2 - 1)\log_4 N, \;\; r_2 = \frac{1}{2}(2\tau_2 - 1)\log_2 N.$$

From this and some rearrangement (8) follows. ∎

Construction 4 offers the nice compromise of correcting a number of double errors needed for good error coverage, while keeping the EC-rate well above that of Construction 2 when this number is not too high. Furthermore, it can be shown that the refined Construction 3 can correct τ_2 *arbitrary* errors in $\mathcal{W}_2$ copies, not just double mag-1 errors.

In an effort to further increase the EC-rate and reduce complexity, we next construct EC-WOM codes that correct the weaker but similarly motivated Amag-1 errors.

C. EC-WOM construction for Amag-1 errors

In the following construction we endow a WOM code with guaranteed error correction while *exclusively using binary codes* in the concatenation. This is a major implementation advantage in practice, as binary codes (in particular BCH codes) are much easier to implement.

1) Tiling WOM code: The inner WOM code we use in this construction is the known tiling-based $(n = 2, q = 8, t = 4, M = 8)$ code from [5], same as in Section IV-A. We use a different mapping as specified next.

2) Mapping 8-ary to three bits: We map the 8-ary symbols to a 3-bit binary representation as specified in Figure 7 showing the decoding function. Note that this representation is *not* the standard 8-ary-to-binary mapping of Figure 3. We order the bits' significance from right to left (the LSB is the right bit), and note the following properties of this mapping. 1) Between two physical states adjacent horizontally or vertically, *exactly* one of the two higher bits is flipped, and between two physical states adjacent on the main diagonal (bottom-left to top-right), both high bits are flipped. 2) The LSB flips along the secondary diagonal (top-left to bottom-right). These properties will be used later on for decoding.

c_2 \ c_1	0	1	2	3	4	5	6	7
7	101	111	100	000	011	001	010	110
6	001	010	110	101	111	100	000	011
5	100	000	011	001	010	110	101	111
4	110	101	111	100	000	011	001	010
3	011	001	010	110	101	111	100	000
2	111	100	000	011	001	010	110	101
1	010	110	101	111	100	000	011	001
0	000	011	001	010	110	101	111	100

Figure 7: Mapping the 8-ary logical symbols of the tiling-based WOM code to three binary symbols.

3) Outer binary EC codes: We use the code $\mathcal{C}_2\left[2N, 2N - r_1\right]$ with design distance $2\tau + 1$ for the two upper bits, and for the lower bit we use the code $\mathcal{C}_2\left[N, N - r_2\right]$ designed with distance $\tau + 1$.

4) The construction:

Construction 5. *Given the $(2, 8, 4, 8)$ WOM code $\mathcal{W}$ specified with the logical mapping in Figure 7 and EC codes specified in Section IV-C.3, the concatenated code $\mathcal{CW}_3$ is the $\left(2N, 8, 4, 2^{2N-r_1} \cdot 2^{N-r_2}\right)$ WOM code obtained by taking N copies of $\mathcal{W}$ and inputing to each two bits of $\mathcal{C}_2\left[2N, 2N - r_1\right]$ as the higher bits and a bit of $\mathcal{C}_2\left[N, N - r_2\right]$ to its lower bit.*

The properties of Construction 5 result in the following proposition. The proof includes the decoding algorithm.

Proposition 5. *For any N and τ, a $\left(2N, 8, 4, 2^{3N-r_1-r_2}\right)$ code $\mathcal{CW}_3$ obtained by Construction 5 is τ Amag-1 error correcting. Moreover, $\mathcal{CW}_3$ has EC-rate*

$$1 - \frac{3}{4} \cdot \frac{\left(2\tau - \frac{4}{3}\right) \log_8 N + \frac{4}{9}\tau - \frac{2}{9}}{N}. \tag{9}$$

Proof: Let τ_1 denote the number of copies of $\mathcal{W}$ that suffered an Amag-1 error in exactly one of their two cells, and τ_2 denote the number of copies of $\mathcal{W}$ that suffered Amag-1 errors in both their cells. We show that any combination with $\tau_1 + 2\tau_2 \leqslant \tau$ is correctable. We first decode the binary code $\mathcal{C}_2\left[2N, 2N - r_1\right]$ of the two higher bits. From the mapping seen in Figure 7, each Amag-1 error results in an error in the two upper bits, thus the $2N$ upper bits suffer at most $\tau_1 + 2\tau_2$ errors. Since $\tau_1 + 2\tau_2 \leqslant \tau$, the decoder of $\mathcal{C}_2\left[2N, 2N - r_1\right]$ can correct all these errors. Next we decode $\mathcal{C}_2\left[N, N - r_2\right]$ of the lower bit. For each copy of $\mathcal{W}$ that suffered a single Amag-1 error we input an erasure to the lower bit's decoder. All other lower bits are correct. Since the decoder sees at most $\tau_1 \leqslant \tau$ erasures, it can recover the lower bits successfully.

To prove the EC-rate, we observe that

$$2^{3N-r_1-r_2} = 8^{N-\frac{1}{3}(r_1+r_2)}. \tag{10}$$

Hence in Definition 5 we have $r_{\text{eff}} = \frac{1}{3}(r_1 + r_2)$ 8-ary symbols. Substituting in r_{eff} the binary BCH redundancies from (2)

$$r_1 = \frac{1}{2}(2\tau - 1)\log_2 2N, \;\; r_2 = \frac{1}{2}(\tau - 1)\log_2 N,$$

and after some rearrangement we get (9) for the EC-rate. ∎

As N grows, r_{eff} of Construction 5 tends to $\left(\frac{3}{2}\tau - 1\right)\log_8 N$ (see (9)), which is the lowest redundancy among all the constructions given so far in the paper. The closest redundancy to this is achieved by Construction 3, but without guarantee to correct double errors in the same WOM copy. Another benefit of using Construction 5 in the presence of Amag-1 errors is

that both codes are binary codes, which implies simpler implementation with BCH codes, and also greater flexibility to use alternative lower complexity binary codes like LDPC codes. A summary and comparison of the constructions in this section is given in table I.

TABLE I: Comparison of EC-WOM constructions

Const.	Error type	EC-rate	EC alphabets	Gap from Construction 1
1	any	$1-\frac{7}{8}\cdot\frac{(2\tau-1)\log_8 N}{N}$	$GF(8)$	–
2	mag-1	$1-\frac{\left(\frac{3}{4}(\tau-1)+\frac{1}{2}(2\tau-1)\right)\log_8 N}{N}$	$GF(4), GF(2)$	$\frac{3}{8}\cdot\frac{\log_8 N}{N}$
3	single mag-1	$1-\frac{3}{4}\cdot\frac{(2\tau-1)\log_8 N}{N}$	$GF(4)$	$\frac{1}{8}\cdot\frac{(2\tau-1)\log_8 N}{N}$
4	single mag-1, any	$1-\frac{3}{4}\cdot\frac{\left(2\tau_1+\frac{10}{3}\tau_2-\frac{5}{3}\right)\log_8 N}{N}$	$GF(4), GF(2)$	$\frac{1}{8}\cdot\frac{(14\tau-12\tau_1-20\tau_2+3)\log_8 N}{N}$
5	Amag-1	$1-\frac{3}{4}\cdot\frac{\left(2\tau-\frac{4}{3}\right)\log_8 N+\frac{4}{9}\tau-\frac{2}{9}}{N}$	$GF(2), GF(2)$	$\frac{1}{24}\cdot\frac{(6\tau+3)\log_8 N-8\tau+4}{N}$

V. EC-WOM Codes for Random Errors

After constructing EC-WOM codes for guaranteed worst-case errors in Section IV, we now move to study EC-WOM codes for random errors. Our objective is to improve and analyze the WOM error correction over realistic memory channel models. To keep implementation complexity low, we only use binary EC-codes throughout the section, and use multiple codes through the concept of *multi-level coding* [20]. To not confuse the levels in the multi-level coding hierarchy with cell levels, we call the former "bit-levels" instead of just "levels". As in the previous parts of the paper, in its reminder too we assume a basic readout of cells returning a "hard" estimate in $\{0,\ldots,q-1\}$ for each cell level. A finer ("soft") readout is costly in non-volatile storage devices, hence we cannot assume its availability.

A. Multi-level coding

In multi-level coding, one maps each high-order symbol (e.g., a modulation symbol in communications, a q-ary cell level in standard Flash, or a M-ary WOM logical symbol in this paper) to multiple bits, each coded by a separate binary EC-code. Each individual EC-code is designed for the channel induced on its bit-level by the high-order channel. Multi-level coding orders the bit-levels in a hierarchy, and applies *multi-stage decoding*, whereby the decoder outputs of a bit-level are supplied to the decoders of the bit-levels above it in the hierarchy. We now define the specific variant of multi-stage decoding we use in this section, first without taking into account the WOM code feeding the multi-stage decoder.

The multi-stage decoder: The inputs to the multi-stage decoder are m bit-levels ordered $1,\ldots,m$, each having N inputs from the alphabet $\{0,1,\perp\}$. Each bit-level code with some minimum distance d_i is decoded by an error+erasure (hard-decision) bounded-distance decoder. In bit-level 1, the inputs to the decoder are equal to the inputs to the multi-stage decoder in coordinates that have 0 or 1, and are erasures in coordinates that have $\perp$. For each bit level $i>1$, we in addition input erasures in coordinates that have 0 or 1 and at least one bit-level in $\{1,\ldots,i-1\}$ corrected an error in this coordinate.

The most distinctive feature of the multi-stage decoder here compared to prior multi-stage decoder definitions is that it gets $\perp$ (erasure) inputs *despite using only hard estimates* from the physical readout. Previously (e.g. in M-PSK/QAM communications) erasure inputs were only possible with soft channel outputs, but here the $\perp$s coming from the inner WOM code offer a significant decoding-performance boost even with hard channel outputs. The multi-stage decoder used here is a hybrid between the "hard" and "soft" decision variants [20], in that it uses hard-decision decoding but propagates bit errors upwards as a soft indication of an unreliable coordinate. Our choice to erase bits with a corrected error in any lower bit-level is a simple special case of a more complex erasing problem [6]. We justify this simple choice in the next sub-section discussing the WOM constructions for multi-level coding. We now move to introduce the WOM codes into the multi-level coding framework. For this we first revisit and refine the definition of the WOM decoding function from Section II.

Definition 11. *The WOM decoding function is redefined as* $\psi:\{0,\ldots,q-1\}^n\times\{1,\ldots,t\}\to\{0,\ldots,M-1\}\cup\perp$, *where the second argument* $k\in\{1,\ldots,t\}$ *is the write number, and the output* $\perp$ *represents a hole (invalid state). In addition, let* $m=\log_2 M$ *and a* M*-ary to binary mapping be given. Define the WOM* binary *decoding function as* $\psi_b:\{0,\ldots,q-1\}^n\times\{1,\ldots,t\}\to\{0,1\}^m\cup\perp^m$.

Note that the addition of the $k\in\{1,\ldots,t\}$ argument to the decoding function is done for the purpose of error detection, so as usual in WOM any physical state is mapped to at most one (non $\perp$) logical state. As an example for Definition 11 we look at the WOM code of Figure 5 and the physical state $\boldsymbol{x}=(2,0)$. We get $\psi(\boldsymbol{x},1)=5$, and $\psi_b(\boldsymbol{x},1)=101$ according to the standard 8-ary to binary mapping. Alternatively, $\psi(\boldsymbol{x},2)=\perp$ and $\psi_b(\boldsymbol{x},2)=\perp\perp\perp$, because $\boldsymbol{x}$ is not possible in write number 2. We also use the convention of $\psi(\boldsymbol{x},k)=\perp\ \forall\boldsymbol{x}\notin\{0,\ldots,q-1\}^n$ (in case the physical readout returns a value outside the q legal levels).

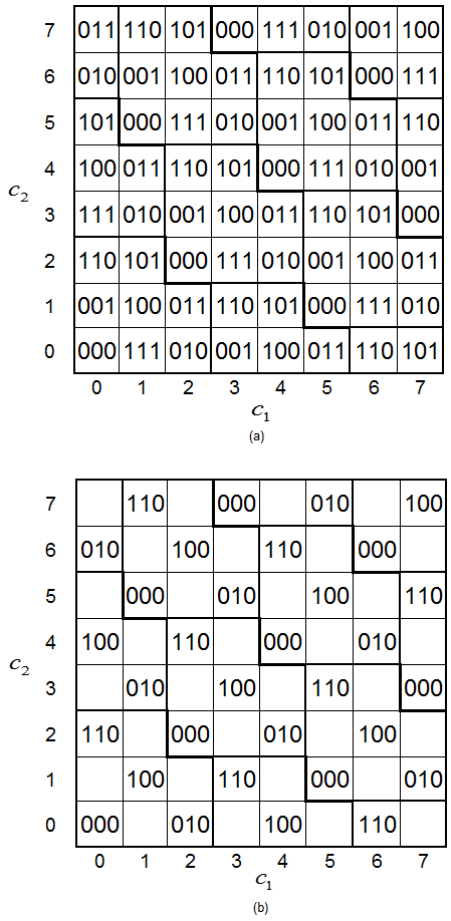


Figure 8: Multi-level coding WOM decoding function and mapping (tiling WOM). (a) The full decoding function and mapping, and (b) the residual decoding function after the LSB was decoded to 0 by bit-level 1.

Our focus in this paper is on WOM codes with logical alphabet size $M = 8$, translating to $\log_8 = 3$ bit-levels in the multi-level coding hierarchy (but the same techniques can be extended to more bit-levels for higher M values). We start our discussion of WOM multi-level coding by listing the concatenation components, as done in each sub-section of Section IV.

1) Low error-propagation WOM codes: The WOM codes used in this section are suitably designed to minimize the propagation of errors from the q-ary memory alphabet to the binary EC-code alphabet. That is, memory read/write noise induces wrong q-ary values in the read physical states, and we wish these errors to be expressed in few bit errors seen by the binary EC decoders. Lower error propagation can be achieved by introducing invalid physical states in the decoding function, which are physical states not corresponding to any logical state (and thus will never be reached by the update function). These invalid states are called in the sequel *holes*. To ease the design of such WOM codes, we later define the WOM code's *propagation index* as a single-letter measure of its error-propagation performance.

2) Mapping 8-ary to three bits: The 8-ary logical alphabet of the WOM code is mapped to 3 bits with the following property: physical states adjacent horizontally or vertically have the opposite binary value in their lowest bit (LSB). Thanks to this property, a successful decoding of the LSB will result in the next two bit-levels having equivalent channels without erroneous physical states at Manhattan distance 1. The upper two bits are mapped such that physical states adjacent diagonally have on average a small number of bits difference. Manhattan distance 1 and diagonal errors are, respectively, the first and second dominant errors in memory channels such as those with additive Gaussian-distributed noise. An example of such a mapping is shown in Figure 8 for the case of the tiling WOM code: (a) shows the tiling-based $(2, 8, 4, 8)$ WOM decoding function and its mapping, while (b) shows the subset of the physical states with LSB equal to 0. Assuming that the LSB was correctly decoded to 0 by bit-level 1's decoder, (b) shows that a physical error with Manhattan distance 2 is necessary to introduce an error in any of the upper two bits (recall that according to the specification of the multi-stage decoder, a Manhattan distance 1 error will be found by bit-level 1 and result in erasures and not errors in the upper bits.)

3) Multi-level binary EC codes: Each bit-level is encoded with a different length-N binary code according to the reliability of the bit determined by the noise and by the WOM code and its 8-ary to binary mapping. Bit-level i is encoded with an EC code with minimum distance d_i. We later explore how to divide a total redundancy budget among the bit-levels to get

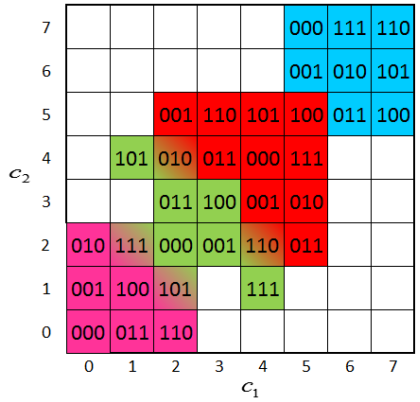


Figure 9: The MLcode1 construction.

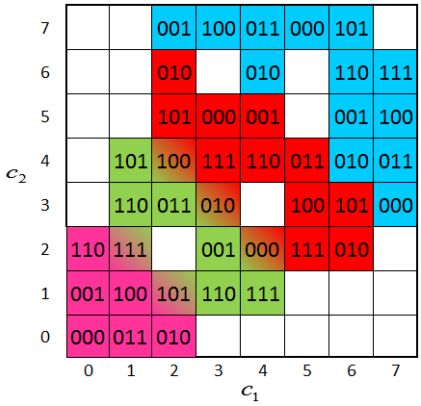


Figure 10: The MLcode2 construction.

the best overall error correction. The multi-stage decoder specified earlier in the sub-section is used, and its $m = 3$ inputs in coordinate l are $\psi_b(\boldsymbol{x}_l, k)$, where $\boldsymbol{x}_l$ is the WOM read physical state in that coordinate and k is the write number. Recall from Definition 11 that the inputs at coordinate l will be $\perp\perp\perp$ in the following cases: 1) a physical state that is a hole in the WOM decoding function, and 2) a physical state that is outside the region reachable at the current write k.

B. WOM constructions

In addition to the tiling WOM code and its mapping shown in Figure 8, we construct three additional WOM codes – all with parameters $(n, q, t, M) = (2, 8, 4, 8)$ – to work better in the multi-level coding setting. The codes are specified in Figures 9, 10, and 11. As before, the decoding functions are given explicitly, while the update functions can be readily inferred from the colors representing the regions of the $t = 4$ writes. The codes are specified with their binary mappings. We name the constructions MLcode1 (Figure 9), MLcode2 (Figure 10), and MLcode3 (Figure 11). MLcode1 is based on [13] but with injected holes, and MLcode3 is based on Construction 3 but with more holes. Examining the WOM constructions in Figures 8,9,10,11 justifies the specification of the multi-stage decoder in Section V-A: any bit corrected $0 \to 1$ or $1 \to 0$ by bit-level $i = 1$ (LSB) or $i = 2$ implies that the readout is mostly useless for bit-levels $j > i$, because we can find in most cases two physical states at the same Manhattan distance from the read physical state with opposite binary values at bit-level j. For example, in Figure 9 if we read physical state $(4, 4)$ and bit-level 1 corrects the LSB $0 \to 1$, then in Manhattan distance 1 from $(4, 4)$ we have all possibilities for the upper two bits. Hence erasing bit-levels $j > i$ by the multi-stage decoder is a justified decision. Better correction performance can be obtained if the multi-stage decoder uses the knowledge of the WOM code to decide what bits to erase. For example, in Figure 11 if we read physical state $(1, 1)$ and bit-level 1 corrects the LSB $0 \to 1$, then the only two valid physical states at Manhattan distance 1 from $(1, 1)$ have 0 in the second bit, and thus may not be erased by the decoder of bit-level 2. While this may offer improvement in practice, in the analysis we assume the basic "WOM-independent" multi-stage decoder defined in Section V-A.

Our goal is to analyze the error-correcting performance of the different WOM constructions under multi-level coding with multi-stage decoding. Such an analysis tool will help in choosing the best WOM code and in designing the EC codes used in each bit-level.

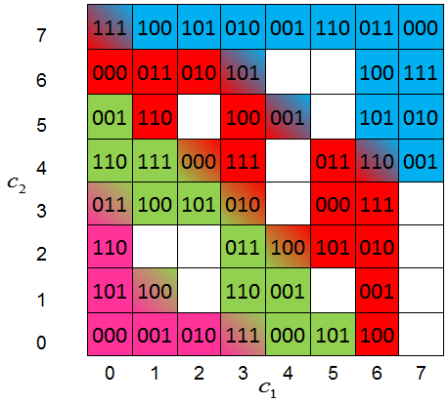


Figure 11: The MLcode3 construction.

C. Bounding the decoding-error probability

Our main tool to analyze multi-level coding of WOM codes is through the calculation of the decoding-error probability. In multi-level coding, a decoding-error event happens when at least one of the bit-level decoders either returns the wrong codeword or declares failure. We denote the decoding-error probability by P_E. In addition, we define the following.

Definition 12. *Let p_i,q_i denote the probability that a bit input to the i-th bit-level decoder is an error, erasure, respectively. p_i and q_i correspond to a given write k, but we keep this dependence implicit. Let p_{E_i} be the probability of decoding error at bit-level i, conditioned that all bit-levels $1,\ldots,i-1$ were decoded correctly.*

Since calculating the exact decoding-error probability is difficult, we instead calculate an upper bound that will offer a performance guarantee. We calculate an upper bound on P_E using the well-known union-bound technique [20]

$$P_E \leqslant \sum_{i=1}^{m} p_{E_i}.$$

Given the input error,erasure probabilities p_i,q_i to an error+erasure bounded-distance decoder of a code with minimum distance d_i, the bit-level's decoding-error probability is obtained through the standard trinomial formula

$$p_{E_i} = 1 - \sum_{l,j:2l+j\leqslant d_i-1} \frac{N!}{l!j!(N-l-j)!} p_i^l q_i^j (1-p_i-q_i)^{N-l-j}.$$

We now need to calculate p_i and q_i for each bit-level i, and they will depend on the chosen WOM code. Moreover, in the input erasure probability q_i we need to include erasures due to errors in lower bit-levels, hence the required conditioning in Definition 12 on correct decoding of the lower bit-levels.

Calculating p_i and q_i for bit-level i is now explained. For a binary string a we use in the sequel a_i to denote its i-th bit, and a_i^j to denote its sub-string $a_i,\ldots,a_j$, for any $i<j\leqslant m$. By Definition 12 we have

$$p_i \triangleq \mathsf{P}_i(\text{error}),\ q_i \triangleq \mathsf{P}_i(\text{erasure}),$$

where the subscript i corresponds to bit-level i. Our first step toward calculating p_i and q_i is the marginalization over the correct bits in the bit-levels up to i

$$\begin{aligned}\mathsf{P}_i(\text{error}) =& \sum_{(b_1,\ldots,b_i)\,\in\,\{0,1\}^i} \mathsf{P}_i(\text{error}|b_1,\ldots,b_i)\cdot \mathsf{P}(b_1,\ldots,b_i)\\ =&\frac{1}{2^i} \sum_{(b_1,\ldots,b_i)\,\in\,\{0,1\}^i} \mathsf{P}_i(\text{error}|b_1,\ldots,b_i),\end{aligned}$$

where the second equality follows from the assumption that bit assignments follow the uniform distribution. A similar calculation is done for $\mathsf{P}_i(\text{erasure})$, but we skip it to avoid duplicity. Then each probability in the sum is further marginalized over the correct physical state of the WOM code feeding the bit-level decoder

$$\begin{aligned}&\mathsf{P}_i(\text{error}|b_1,\ldots,b_i) =\\ &\sum_{\boldsymbol{x}\,\in\,\{0,\ldots,q-1\}^n} \mathsf{P}_i(\text{error}|b_1,\ldots,b_i,\boldsymbol{x})\cdot \mathsf{P}(\boldsymbol{x}|b_1,\ldots,b_i).\end{aligned} \tag{11}$$

The first term in the sum are the probabilities $\mathsf{P}_i(\text{error}|b_1,\ldots,b_i,\boldsymbol{x})$ that can be calculated from the WOM decoding function and the channel distribution governing the physical-state transitions. Note that $b_1,\ldots,b_i$ are fully determined given $\boldsymbol{x}$ from the WOM decoding function and mapping, so we may also write this probability simply as $\mathsf{P}_i(\text{error}|\boldsymbol{x})$. The second term in the sum are the probabilities $\mathsf{P}(\boldsymbol{x}|b_1,\ldots,b_i)$ that can be calculated from the WOM update function: there may be multiple physical states in write k with bit values $b_1,\ldots,b_i$, and to know the distribution of $\boldsymbol{x}$ one needs to enumerate all combinations of logical values written in the write sequences $1,\ldots,k$ that end with these i bit values in the k-th write. This distribution is in general not uniform (even though logical values are uniform), as one $\boldsymbol{x}$ may be reached by the update function in more write sequences than another with the same logical value. We clarify that $\mathsf{P}(\boldsymbol{x}|b_1,\ldots,b_i)$ does *not* depend on the channel, only on the WOM code.
To calculate $\mathsf{P}_i(\text{error}|b_1,\ldots,b_i,\boldsymbol{x})$ we sum the channel's transition probabilities from $\boldsymbol{x}$ to all states $\boldsymbol{x}'$ such that $(\psi_b(\boldsymbol{x}',k))_1^{i-1} = (\psi_b(\boldsymbol{x},k))_1^{i-1}$ and $(\psi_b(\boldsymbol{x}',k))_i = 1-(\psi_b(\boldsymbol{x},k))_i$. Similarly, to calculate $\mathsf{P}_i(\text{erasure}|b_1,\ldots,b_i,\boldsymbol{x})$ we sum the transition probabilities from $\boldsymbol{x}$ to all states $\boldsymbol{x}'$ such that $\psi(\boldsymbol{x}',k)=\perp$, or that $(\psi_b(\boldsymbol{x}',k))_1^{i-1} \neq (\psi_b(\boldsymbol{x},k))_1^{i-1}$, where two strings are not equal if at least one bit differs between them. These conditions on $\boldsymbol{x}'$ imply that as we go up in the bit-level hierarchy, q_i grows while p_i diminishes.

Example 3. *Suppose cells are read with additive zero-mean white Gaussian noise (AWGN) with standard deviation σ. Define $Q(s) \triangleq \frac{1}{\sqrt{2\pi}}\int_s^\infty \exp(-u^2/2)du$ as the usual Gaussian tail function. For the WOM code in Figure 10, if $\boldsymbol{x}=(1,1)$ is written in the first write ($k=1$), then in bit-level $i=1$ we have*

$$\mathsf{P}_1(\text{error}|0,(1,1)) = 4\left[Q\left(\frac{0.5}{\sigma}\right)-Q\left(\frac{1.5}{\sigma}\right)\right]\left[Q\left(\frac{-0.5}{\sigma}\right)-Q\left(\frac{0.5}{\sigma}\right)\right].$$

This is the probability that the channel takes us from $\boldsymbol{x}=(1,1)$ to one of its 4 horizontal/vertical neighbors. Similarly,

$$\mathsf{P}_1(\text{erasure}|0,(1,1)) = 1-\mathsf{P}_1(\text{error}|0,(1,1)) - \left[Q\left(\frac{-0.5}{\sigma}\right)-Q\left(\frac{0.5}{\sigma}\right)\right]^2 - 3\left[Q\left(\frac{0.5}{\sigma}\right)-Q\left(\frac{1.5}{\sigma}\right)\right]^2.$$

Erasure is the outcome for $\boldsymbol{x}=(1,1)$ in all $\boldsymbol{x}'$ states except (negative terms from left to right): states that give an error outcome, when $\boldsymbol{x}'=\boldsymbol{x}$, and the three diagonal neighbors of $\boldsymbol{x}=(1,1)$ that are within the region reachable at write number $k=1$. Moreover, in the code of Figure 10 the distribution $\mathsf{P}(\boldsymbol{x}|b_1,\ldots,b_i)$ is uniform for any i in write $k=1$ because in that write there is a 1-1 mapping between the logical and physical states. However, in write $k=4$ we expect to have

$$\mathsf{P}(\boldsymbol{x}=(2,7)|1,0,0) < \mathsf{P}(\boldsymbol{x}=(6,5)|1,0,0),$$

because $(2,7)$ is reachable from only three states of write $k=3$ while $(6,5)$ is reachable from all states except one.

D. Decoding-error probability results for 4 WOM codes

We now use the analysis of Section V-C to calculate and plot the decoding-error performance of the 4 WOM codes in Figures 8,9,10,11. Recall that all the codes have WOM parameters $(n,q,M,t)=(2,8,8,4)$; in this sub-section we take the block length (number of parallel WOM-code copies) to be $N=255$. The channel noise is AWGN with σ values between 0.22 and 0.26, and the distance profile (d_1,d_2,d_3) of the bit-level codes was optimized empirically to get the best performance for a given total redundancy budget. The results are plotted in Figures 12,13,14,15, each one showing the same upper-bound calculation for a different write number $k\in\{1,2,3,4\}$ (recall from Section V-C that p_i and q_i depend on the WOM code and the write number k). Each figure shows the decoding-error probability upper bound for the 4 WOM codes, and in comparison to non WOM-coded information stored on the same 255×2 cells with the same total EC redundancy (the non-WOM redundancy allocation to bit-levels was optimized separately). It is observable that different WOM codes differ considerably in error probability even though they have the same parameters and redundancy. It is also observed that WOM-coded memory enjoys higher reliability than non-WOM, meaning that the same redundancy that is invested in the re-writing feature also serves us in improving reliability. An interesting property is that the ordering of WOM codes in reliability is not absolute but depends on the write number k: for example MLCode3 is the best in $k=3$ but MLCode2 is the best in $k=4$. The tiling construction has the worst reliability in all writes, hence it is not recommended for use. We next show in Figures 16-19 a comparison of the error probability bound of MLCode3 between a decoder that knows the write number and a decoder that does not know the write number. A decoder that does not know the write number cannot use the k-aware WOM decoding function of Definition 11, and thus misses opportunities to supply the EC-decoder with $\perp$ inputs when errors occur. The plots reveal a significant performance gap in favor of knowing the write number. The smaller gap for $k=4$ (Figure 19) can be explained by looking at MLCode3's specification in Figure 11, and observing that at the 4-th write there are few Manhattan-1 neighbors belonging to the 3-rd write.

We next look for a simpler characterization of the WOM code's reliability without the complexity of (11) that requires the enumeration of all pairs of physical states $\boldsymbol{x}$,$\boldsymbol{x}'$. For that we define the *propagation index* in the next sub-section.

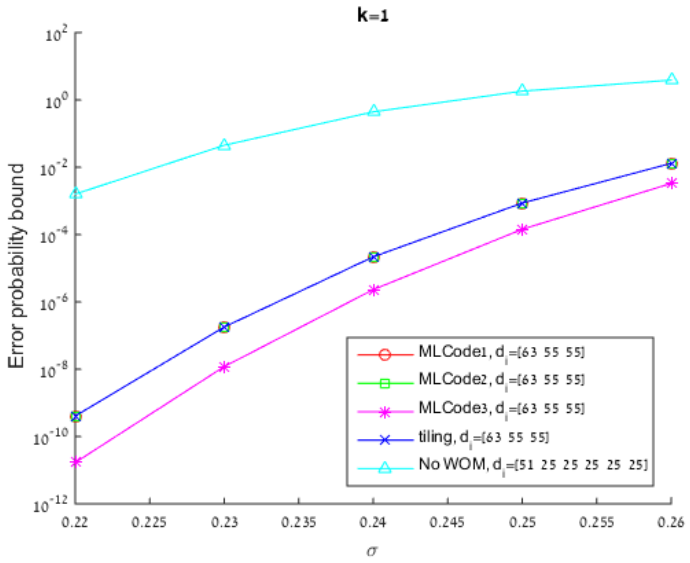


Figure 12: Error probability bound, $k = 1$.

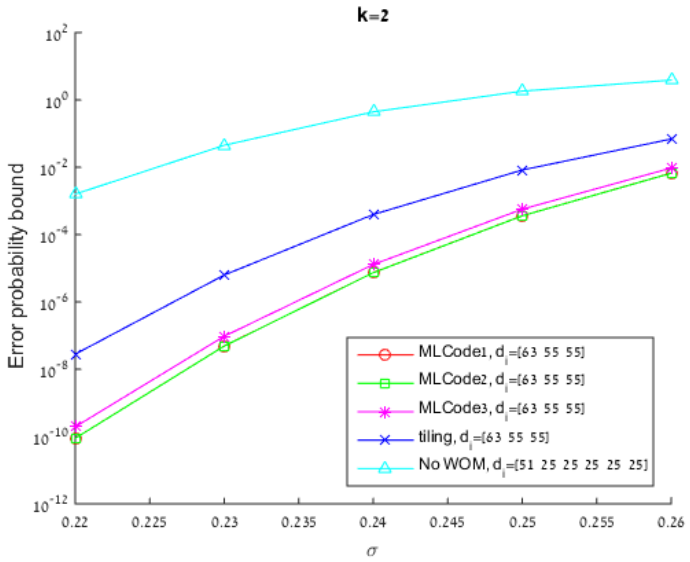


Figure 13: Error probability bound, $k = 2$.

E. Propagation index of a WOM code

In the realistic scenario where noise and inter-cell interference are not too severe, we expect mag-1 errors to be the dominant error type. For that case we propose a simpler characterization of the WOM code's reliability – one that only enumerates the physical states' close neighborhoods in the WOM decoding function.

Definition 13. *Given a valid physical state $\boldsymbol{x}$ and another physical state $\boldsymbol{y}$ of a (n, q, t, M) WOM code we define the* ***error function*** *between $\boldsymbol{x}$ and $\boldsymbol{y}$ in the k-th write as*

$$\nu(\boldsymbol{x}, \boldsymbol{y}; k) = \begin{cases} 1 & \psi(\boldsymbol{x}, k) \neq \psi(\boldsymbol{y}, k) \neq \perp \\ 0.5 & \psi(\boldsymbol{y}, k) = \perp \\ 0 & \psi(\boldsymbol{x}, k) = \psi(\boldsymbol{y}, k) \neq \perp \end{cases}$$

We define the ***average error*** *of $\boldsymbol{x}$ in the k-th write as*

$$\epsilon(\boldsymbol{x}, k) = \frac{1}{2n} \sum_{l \in \{1,\ldots,n\}, j \in \{-1,1\}} \nu(\boldsymbol{x}, \boldsymbol{x} + j \cdot e_l; k),$$

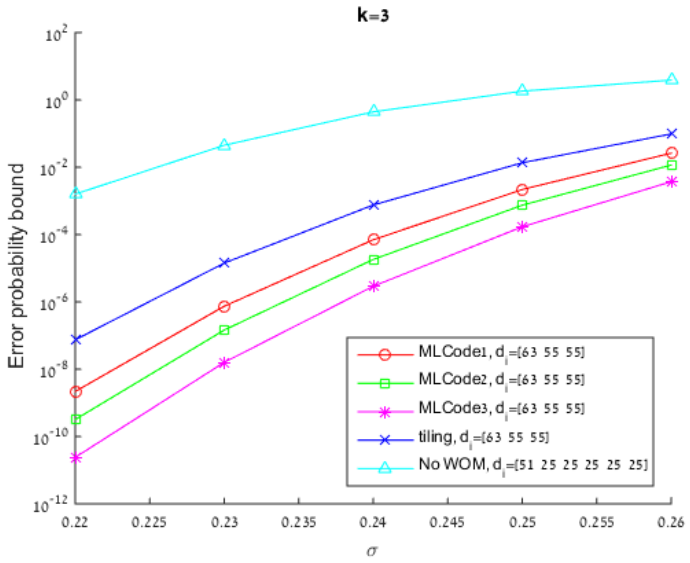


Figure 14: Error probability bound, $k = 3$.

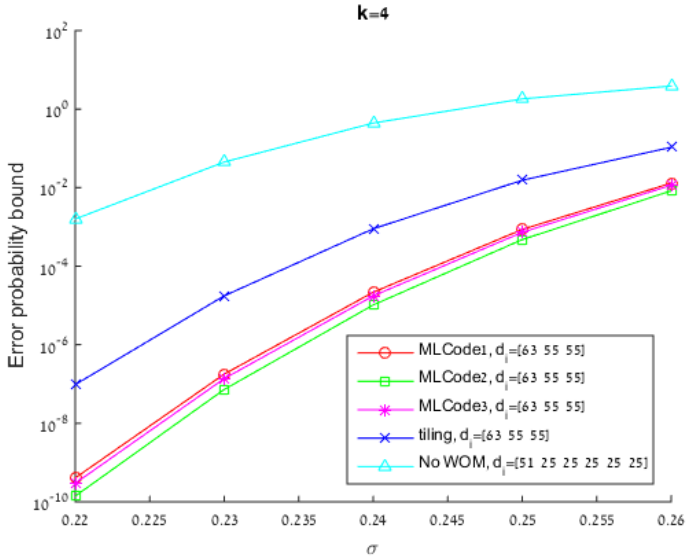


Figure 15: Error probability bound, $k = 4$.

where e_l is the unit vector that is equal to 1 in coordinate l and to 0 in the other $n-1$ coordinates.

The average error measures the average "damage" a mag-1 error causes to a physical state $\boldsymbol{x}$. Note that the above definition of error uses the M-ary WOM alphabet and thus is independent of the binary mapping. The WOM code's propagation index is next defined as an expectation of the average error over the physical states $\boldsymbol{x}$.

Definition 14. *Given a (n, q, t, M) WOM code we define the **propagation index (PI)** in the k-th write as*

$$\eta_k = \sum_{\boldsymbol{x} \in \{0,\ldots,q-1\}^n} p_{\boldsymbol{x},k} \cdot \epsilon\left(\boldsymbol{x}, k\right),$$

*where $p_{\boldsymbol{x},k}$ is the probability to be in physical state $\boldsymbol{x}$ at the k-th write. We define the **geometric propagation index (GPI)** as the propagation index when $p_{\boldsymbol{x},k}$ is the uniform distribution among the physical states of the k-th write. We define the **probabilistic propagation index (PPI)** as the propagation index when $p_{\boldsymbol{x},k}$ is the distribution on the physical states induced by the uniform distribution on the logical states in the sequence of k writes.*

While PPI better captures the behavior of the WOM code in real write sequences, GPI is easier to calculate and does not depend on the update function of the WOM code.

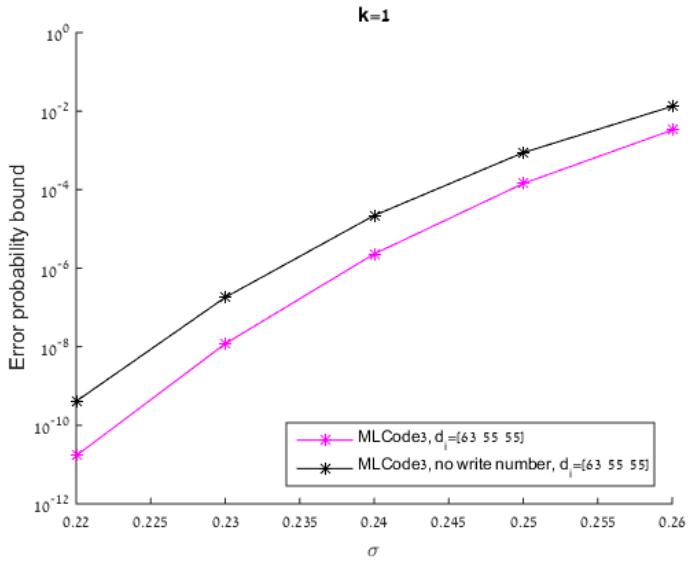


Figure 16: Error probability bound for two decoders, MLCode3, $k = 1$.

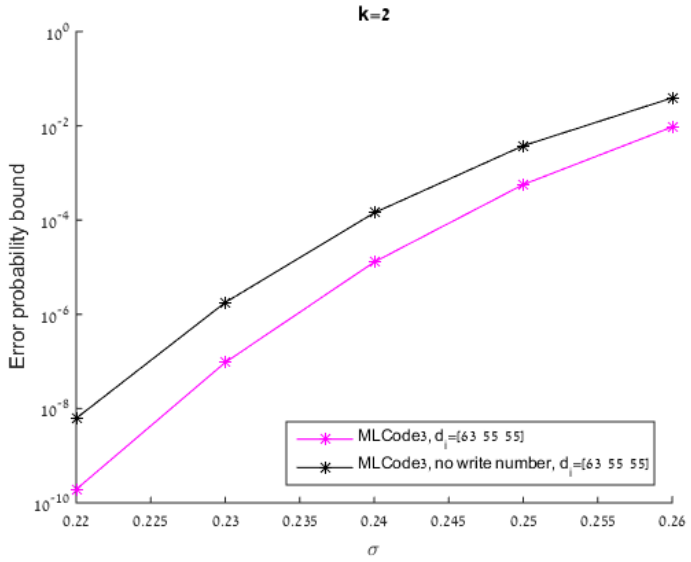


Figure 17: Error probability bound for two decoders, MLCode3, $k = 2$.

These definitions are demonstrated in the following example.

Example 4. *We are given the WOM code in Figure 20 and wish to calculate its propagation indexes. The GPI and PPI of the WOM code in Figure 20 are*

$$GPI = \begin{cases} 0.8125 & k = 1 \\ 0.8 & k = 2 \\ 0.833 & k = 3 \end{cases},$$

and

$$PPI = \begin{cases} 0.8125 & k = 1 \\ 0.8125 & k = 2 \\ 0.8633 & k = 3 \end{cases}.$$

To calculate the PPI we need a precise specification of the update function without ambiguity when there are multiple update choices, which is included in Figure 20.

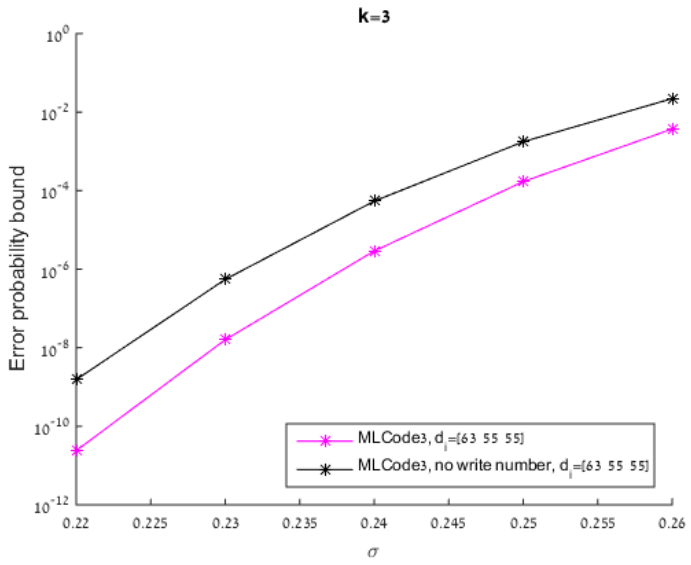


Figure 18: Error probability bound for two decoders, MLCode3, $k = 3$.

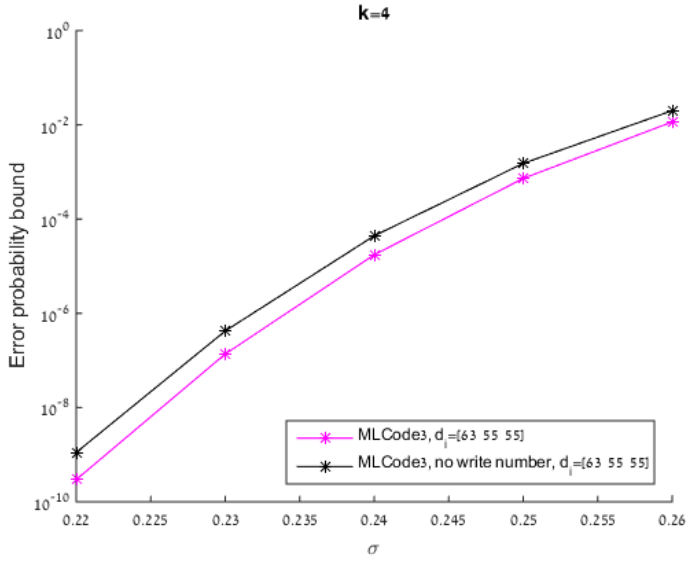


Figure 19: Error probability bound for two decoders, MLCode3, $k = 4$.

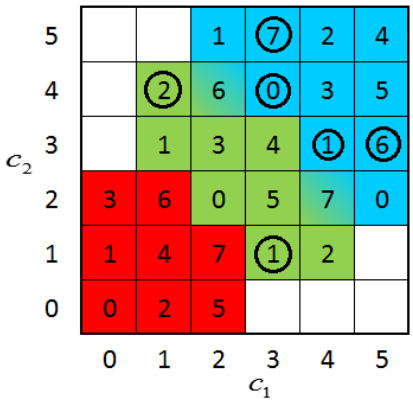


Figure 20: Sample WOM code decoding and update function. When there are multiple choices to update to a given logical value in the same write, the physical state marked with a circle is chosen. For example: when writing in the second write, $\mu(\boldsymbol{x}, 1) = (1, 3)$ if $\boldsymbol{x} \in \{(0, 2), (1, 2)\}$, and $(3, 1)$ otherwise.

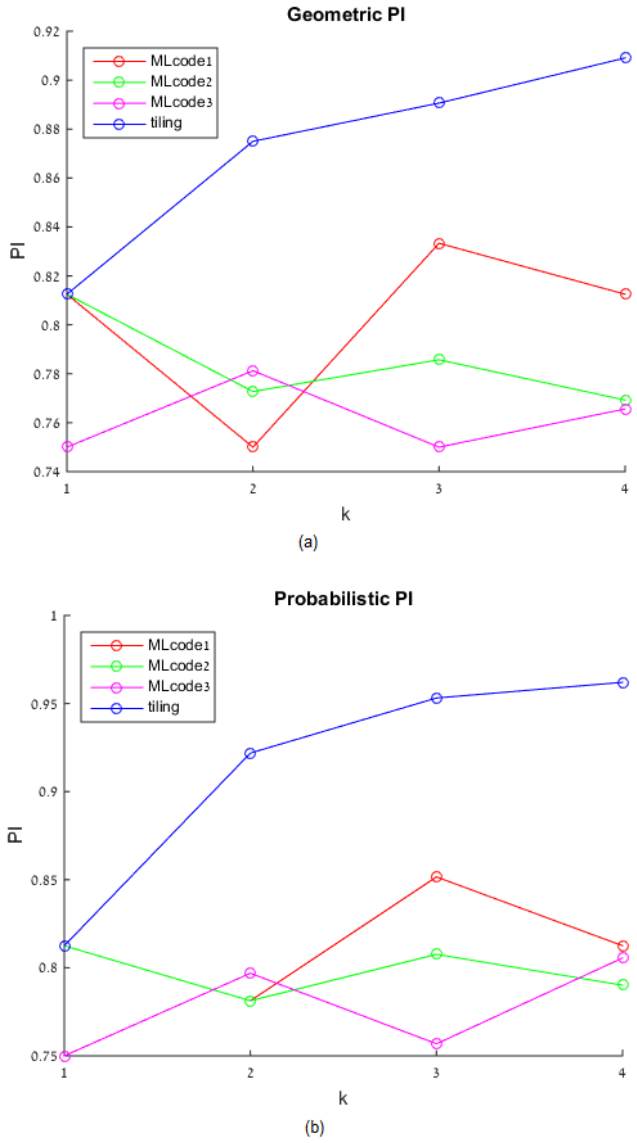


Figure 21: GPI (a) and PPI (b) of the 4 WOM codes analyzed in Section V-D.

We plot in Figure 21 the GPI in (a) and PPI in (b) of the 4 WOM codes analyzed in Section V-D. We see that both GPI and PPI predict for writes $k = 1, 2, 3$ the relative order of the WOM codes in terms of decoding-error probability bound. But only the more refined PPI manages to predict the correct order in write $k = 4$. That means that the advantage of MLCode2 in the 4-th write originates from its update function giving preference to more reliable physical states.

F. EC-WOM with soft-decision decoding

So far in the paper we only used hard-decision error-and-erasure decoders for the EC-codes in the EC-WOM constructions. We next show that EC-WOM constructions can also benefit from *soft-decision* decoders, and more attractively, without need to obtain additional information from the memory channel beyond the quantized levels $\{0, \ldots, q-1\}$. This can be done based on the observation that the *reliability* of a read physical state $\boldsymbol{x} \in \{0, \ldots, q-1\}^n$ depends on its neighborhood in the WOM code's decoding function. While in hard decision $\boldsymbol{x}$ is only characterized by whether $\psi(\boldsymbol{x}, k) = \perp$ or not, in soft decision we can characterize the reliability of $\boldsymbol{x}$ based also on $\psi(\boldsymbol{x}', k)$ of its neighboring physical states $\boldsymbol{x}'$. For example, a bit $(\psi_b(\boldsymbol{x}, k))_i = b$ is more reliable if three of its Manhattan-1 neighbors have $(\psi_b(\boldsymbol{x}', k))_i = \perp$, than if all four neighbors have $(\psi_b(\boldsymbol{x}', k))_i = 1 - b$. Our characterization of the reliability of read physical states relies on the following natural definition of reliability.

Definition 15. *Let $\boldsymbol{x}$ be some (not necessarily valid) WOM physical state. We define the* ***reliability*** *of the i-th bit of $\boldsymbol{x}$ in the k-th*

write as

$$r_i(\boldsymbol{x},k)=\frac{\left|\sum_{\boldsymbol{y}}\lambda(\boldsymbol{y},k)\,\mathsf{P}(\boldsymbol{y}\to\boldsymbol{x})\right|}{\sum_{\boldsymbol{y}}|\lambda(\boldsymbol{y},k)|\,\mathsf{P}(\boldsymbol{y}\to\boldsymbol{x})}, \tag{12}$$

where

$$\lambda(\boldsymbol{y},k)=\begin{cases}1 & (\psi_b(\boldsymbol{y},k))_i=0\\ -1 & (\psi_b(\boldsymbol{y},k))_i=1\\ 0 & (\psi_b(\boldsymbol{y},k))_i=\perp\end{cases},$$

and $\mathsf{P}(\boldsymbol{y}\to\boldsymbol{x})$ *is the probability that the channel takes the input physical state* $\boldsymbol{y}$ *to the output physical state* $\boldsymbol{x}$.

Note that the sums in (12) are over all physical states $\boldsymbol{y}$, including $\boldsymbol{y}=\boldsymbol{x}$. In practice we will only sum over $\boldsymbol{y}\in Ne(\boldsymbol{x})$, that is, physical states in the *neighborhood* of $\boldsymbol{x}$ which is defined as the physical states with non-negligible $\mathsf{P}(\boldsymbol{y}\to\boldsymbol{x})$. In that case $\boldsymbol{x}$ is called the *center of the neighborhood* $Ne(\boldsymbol{x})$. Definition 15 for reliability is natural because the numerator adds transition probabilities with their sign reflecting the logical bit values, while the denominator normalizes with the sum of absolute values of the same probabilities. In general $0\leqslant r_i(\boldsymbol{x},k)\leqslant 1$; in particular $r_i(\boldsymbol{x},k)=1$ if and only if all valid physical states in $Ne(\boldsymbol{x})$ have the same logical bit.

Example 5. *Consider the WOM code shown in Figure 10, and let* $\boldsymbol{x}=(6,1)$. $\boldsymbol{x}$ *is not valid, in particular in the third write, i.e.,* $\psi(\boldsymbol{x},3)=\perp$. *Suppose the channel is defined to cause a mag-1 error with probability* p *in a uniformly selected direction, and no error with probability* $1-p$. *Then because* $\boldsymbol{x}$ *has only one valid physical state* $\boldsymbol{y}=(6,2)$ *in* $Ne(\boldsymbol{x})$, *we have that* $r_i(\boldsymbol{x},3)=1,\forall i\in\{1,2,3\}$. *This means that* $\boldsymbol{x}$ *gives certain bit values, even though it is itself an invalid state.*

To obtain simple and robust soft-decision decoders, we focus next on decoders that only require the *ordering* of the symbol reliabilities at the decoder input, without assigning numerical values to individual-symbol reliabilities. Order-based soft-decision decoders such as GMD [8] and Ordered Statistics [9] are known to offer very good decoding performance. To capture the more significant behaviors of EC-WOM codes with respect to reliability ordering, we next assume the channel is AWGN with a small σ. With small σ we can take as a good approximation the neighborhood of $\boldsymbol{x}$ to be $\boldsymbol{x}$ and its 4 Manhattan-1 neighbors (2 horizontal and 2 vertical). Formally, from now on we assume $Ne(\boldsymbol{x})=\{\boldsymbol{y}:D_1(\boldsymbol{x},\boldsymbol{y})\leqslant 1\}$. Now we can define the following.

Definition 16. *Let* $\boldsymbol{x}$ *be some (not necessarily valid) WOM physical state. We define the* ***neighborhood profile*** *of the* i*-th bit of* $\boldsymbol{x}$ *in the* k*-th write as the tuple* $g_i(\boldsymbol{x},k)=[l_0,l_1]$, *where* l_0 *is the number of zeros,* l_1 *is the number of ones, and* $4-l_0-l_1$ *is the number of* $\perp$*s in the multiset* $\{(\psi_b(\boldsymbol{y},k))_i:\boldsymbol{y}\in Ne(\boldsymbol{x})\setminus\boldsymbol{x}\}$.

Clearly if we know both $(\psi_b(\boldsymbol{x},k))_i$ and $g_i(\boldsymbol{x},k)$ we can calculate the reliability $r_i(\boldsymbol{x},k)$. In Figure 22 we plot the reliability values as a function of the neighborhood profile $[l_0,l_1]$ for $\boldsymbol{x}$ with $(\psi_b(\boldsymbol{x},k))_i=0$ in (a), and for $\boldsymbol{x}$ with $(\psi_b(\boldsymbol{x},k))_i=\perp$ in (b). The values for $\boldsymbol{x}$ with $(\psi_b(\boldsymbol{x},k))_i=1$ are symmetric to (a), only exchanging the roles of l_0 and l_1. With a neighborhood center of $\perp$ the roles of l_0,l_1 are symmetric, so the profiles missing from (b) can be completed by exchanging l_0,l_1. In the plots the reliabilities are sorted from highest to lowest.

Looking at the x-axis of Figure 22 we have a way to order decoder inputs by reliability for soft-decision decoding: the most reliable neighborhood profiles are those with reliability 1 in Figure 22 (a),(b); then the other profiles in decreasing order in (a), and lastly those with lowest reliability in (b). While Figure 22 is specific for $\sigma=0.3$, the next results show that the ordering of neighborhood profiles is the same for all values of σ below some threshold. Let $[l_0,l_1]_0$ be the reliability of neighborhood profile $[l_0,l_1]$ when the neighborhood center's logical bit is 0, and $[l_0,l_1]_\perp$ be the reliability when the center is $\perp$. Then we have the following two propositions.

Proposition 6. *For any* $\sigma\leqslant 0.519$ *we have the following*

$$[4,0]_0=[3,0]_0=[2,0]_0=[1,0]_0=[0,0]_0= \tag{13}$$
$$[4,0]_\perp=[3,0]_\perp=[2,0]_\perp=[1,0]_\perp=1, \tag{14}$$
$$[3,1]_0\geqslant[2,1]_0\geqslant[1,1]_0\geqslant[0,1]_0, \tag{15}$$
$$[2,2]_0\geqslant[1,2]_0\geqslant[0,2]_0, \tag{16}$$
$$[1,3]_0\geqslant[0,3]_0\geqslant[0,4]_0, \tag{17}$$

and

$$[3,1]_\perp\geqslant[2,1]_\perp\geqslant[2,2]_\perp=[1,1]_\perp=[0,0]_\perp=0. \tag{18}$$

The proof of Proposition 6 is straightforward: the equalities are implied directly from (12) and the inequalities follow from the monotonicity in one of l_0 or l_1 while the other is fixed. The condition on σ ensures that in the entire neighborhood the

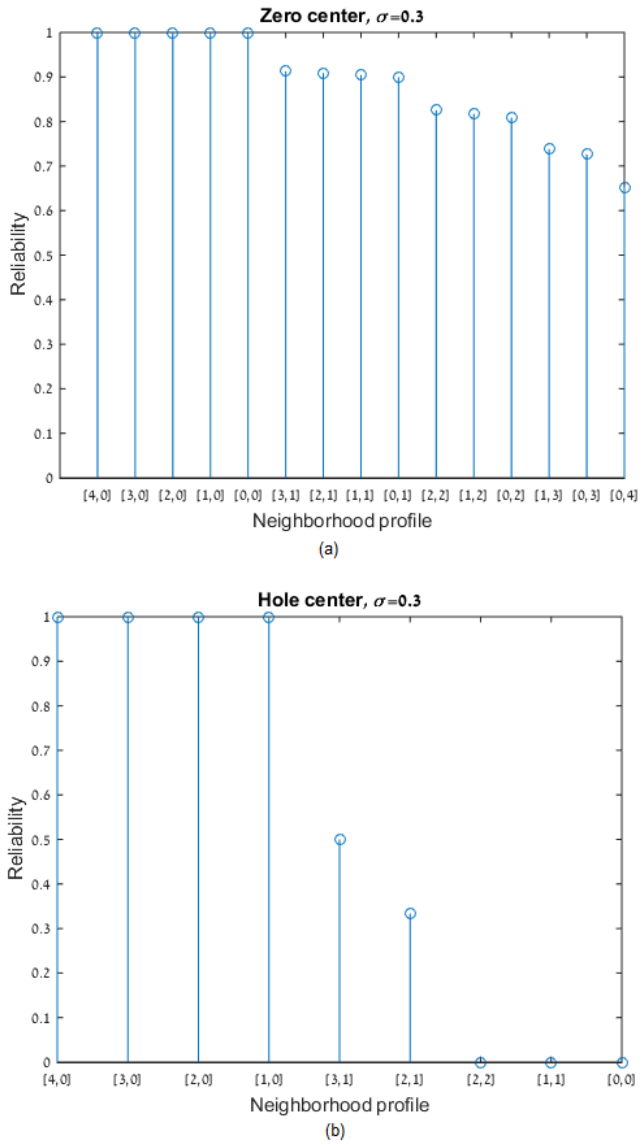


Figure 22: Reliability as a function of neighborhood profiles for $\sigma = 0.3$, when the center of the neighborhood is 0 (a), and $\perp$ (b). Symmetric cases are omitted from (b).

most reliable binary value equals to the value in the center of the neighborhood (and thus in (15)-(17) a higher l_0 or lower l_1, while the other in fixed, implies higher reliability). The cutoff value of $\sigma = 0.519$ is when $[0,4]_0$ (the lowest reliability of a neighborhood profile) becomes negative.

Let z_1 denote the probability that a particular cell is read with a mag-1 error (recall Definition 7), and let z_0 denote the probability that the cell is read with no error. With the AWGN noise model we have $z_1 = Q(0.5/\sigma) - Q(1.5/\sigma)$ and $z_0 = Q(-0.5/\sigma) - Q(0.5/\sigma)$. For $\boldsymbol{y}$ that is a particular Manhattan-1 neighbor of $\boldsymbol{x}$ we have $\mathsf{P}(\boldsymbol{y} \to \boldsymbol{x}) = z_1 z_0$. We also have $\mathsf{P}(\boldsymbol{x} \to \boldsymbol{x}) = z_0^2$. We can now prove a less obvious ordering relation among the neighborhood reliabilities.

Proposition 7. *For any $\sigma \leqslant 0.795$ we have*

$$[0,1]_0 \geqslant [2,2]_0, \tag{19}$$

and

$$[0,2]_0 \geqslant [1,3]_0. \tag{20}$$

Moreover, for any $\sigma \leqslant 0.341$ we have,

$$[0,4]_0 \geqslant [3,1]_\perp. \tag{21}$$

Proof: Recall that $\mathsf{P}(\boldsymbol{y} \to \boldsymbol{x}) = z_1 z_0$, $\mathsf{P}(\boldsymbol{x} \to \boldsymbol{x}) = z_0^2$. Substituting these probabilities in (12) we get the reliabilities $[0,2]_0 = \frac{z_0^2 - 2z_0 z_1}{z_0^2 + 2z_0 z_1}$ and $[1,3]_0 = \frac{z_0^2 - 2z_0 z_1}{z_0^2 + 4z_0 z_1}$. This readily establishes (20) for any σ. To show (19) we calculate $[0,1]_0 = \frac{z_0 - z_1}{z_0 + z_1}$ and $[2,2]_0 = \frac{z_0}{z_0 + 4z_1}$. Then we get that $[0,1]_0 \geqslant [2,2]_0$ so long that $z_0 \geqslant 2z_1$, or equivalently when $\sigma \leqslant 0.795$. Lastly, to prove (21) we calculate $[0,4]_0 = \frac{z_0 - 4z_1}{z_0 + 4z_1}$ and $[3,1]_\perp = \frac{1}{2}$, and get the inequality so long that $\sigma \leqslant 0.341$. ∎

The following corollary is a useful tool for order-based soft-decision decoders such as GMD and Ordered Statistics, because it establishes a universal ordering of neighborhood profiles for noise levels that are below a reasonable threshold.

Corollary 8. *For any $\sigma \leqslant 0.341$ the order of neighborhood profiles in non-increasing reliability is*

$$
\begin{aligned}
&[4,0]_0, [3,0]_0, [2,0]_0, [1,0]_0, [0,0]_0,\\
&[4,0]_\perp, [3,0]_\perp, [2,0]_\perp, [1,0]_\perp,\\
&[3,1]_0, [2,1]_0, [1,1]_0, [0,1]_0, [2,2]_0,\\
&[1,2]_0, [0,2]_0, [1,3]_0, [0,3]_0, [0,4]_0,\\
&[3,1]_\perp, [2,1]_\perp, [2,2]_\perp, [1,1]_\perp, [0,0]_\perp.
\end{aligned}
$$

and recall that $[l_0, l_1]_1$ can be entered in that order by the symmetry $[l_0, l_1]_1 = [l_1, l_0]_0$.

VI. Conclusion

The key message to take from this study of EC-WOM codes is the interesting inter-relations between the WOM and error-correction features of the memory. When WOM and error correction are considered jointly, some of the WOM redundancy can be directed for better error correction, reducing the overall cost of obtaining both features. Concrete examples of this are given in the constructions and analyses of this paper. Another benefit of concatenating WOM and EC codes is studied in the second part of the paper: obtaining soft information for EC decoding from the WOM decoder, without need to increase the raw measurement precision of the memory cells. There are many interesting problems left open by this paper: 1) improving the constructions of this paper for the same WOM parameters, 2) extending the constructions to additional WOM parameters (also beyond $n = 2$, which looks non-trivial) and other error models, 3) concatenating WOM codes with other types of codes like LDPC or polar codes, and 4) studying the fundamental limits of WOM+EC combination with low complexity. Even further, an interesting direction beyond the scope of the current scheme is how to design the EC+WOM code to "recover" from a write error detected in an intermediate write, so that subsequent writes can still be served.

VII. Acknowledgement

We thank Evyatar Hemo and Brian Kurkosky for valuable discussions.

References

[1] A. Bhatia, M. Qin, A. Iyengar, B. Kurkoski, P. Siegel, "Lattice-based WOM codes for multilevel flash memories". IEEE Journal on Selected Areas in Communications, 32(5), pp. 933-945, 2014.

[2] N. Bitouze, A. Graell i Amat, E. Rosnes, "Using short synchronous WOM codes to make WOM codes decodable". IEEE Transactions on Communications, 62(7), pp. 2156-2169, 2014.

[3] R. Bose, D. Ray-Chaudhuri, "On a class of error correcting binary group codes". Information and Control, 3(1), pp. 68-79, 1960.

[4] M. Bossert, "Channel Coding for Telecommunications". John Wiley & Sons, Inc, 1999.

[5] Y. Cassuto, E. Yaakobi, "Short Q-ary fixed-rate WOM codes for guaranteed rewrites and with hot/cold write differentiation". IEEE Transactions on Information Theory, 60(7), pp. 3942-3958, 2014.

[6] K. Fazel, A. Chouly, "Multistage decoding using erasing technique for multilevel coding". In IEEE Global Telecommunications Conference, Globecom 1991.

[7] F. Fu, A.H. Vinck, "On the capacity of generalized write once memory with state transitions described by an arbitrary directed acyclic graph ". IEEE Transactions on Information Theory, 45(1), pp. 308-312, 1999.

[8] G. Forney, "Generalized minimum distance decoding". IEEE Transactions on Information Theory, 12(2), pp. 125-131, 1966.

[9] M.P. Fossorier, S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics". IEEE Transactions on Information Theory, 41(5), pp.1379-1396, 1995.

[10] R. Gabrys, E. Yaakobi, L. Dolecek, S. Kayser, P.H. Siegel, A. Vardy, J.K. Wolf, "Non-binary WOM-codes for multilevel flash memories". In IEEE Information Theory workshop, ITW 2011.

[11] K. Haymaker, C.A. Kelley, "Geometric WOM codes and coding strategies for multilevel flash memories". arXiv:1206.5762v1, May 2012.

[12] C. Heegard, "On the capacity of permanent memory". IEEE Transactions on Information Theory, 31(1), pp. 34-42, 1985.

[13] E. Hemo, Y. Cassuto, "d-imbalance WOM codes for reduced inter-cell interference in multi-level NVMs". IEEE Journal on Selected Areas in Communications, 34(9), pp. 2378-2390, 2016.

[14] A. Hocquenghem, "Codes correcteurs derreurs". Chiffres, 2(2), pp. 147-56, 1959.

[15] A. Jiang, Y. Li, E. Gad, M. Langberg, J. Bruck, "Joint rewriting and error correction in write-once memories". In IEEE International Symposium on Information Theory, ISIT 2013.

[16] A. Jiang, V. Bohossian, J. Bruck, "Rewriting codes for joint information storage in flash memories". IEEE Transactions on Information Theory, 56(10), pp. 5300-5313, 2010.

[17] B. Kurkoski, "A note on using lattices for error-correction and rewriting in flash memories". In Proceedings of the 33rd Symposium on Information Theory and its Applications, SITA 2010 (Nagano, Japan).

[18] S. Odeh, Y. Cassuto, “NAND flash architectures reducing write amplification through multi-write codes”. In IEEE conference on Mass Storage Systems and Technologies, MSST 2014.
[19] R. Rivest, A. Shamir, “How to reuse a write-once memory”. Information and Control, 55(1-3), pp. 1-19, 1982.
[20] U. Wachsmann, R.F. Fischer, J.B. Huber, “Multilevel codes: Theoretical concepts and practical design rules”. IEEE Transactions on Information Theory, 45(5), pp. 1361-1391, 1999.
[21] E. Yaakobi, P. Siegel, A. Vardy, J. Wolf, “Multiple error-correcting WOM-codes”. IEEE Transactions on Information Theory, 58(4), pp. 2220-2230, 2012.
[22] G. Yadgar, E. Yaakobi, A. Schuster, “Write once, get 50% free: saving SSD erase costs using WOM codes”. In USENIX conference on File and Storage Technologies, FAST 2015.
[23] S. Yekhanin, I. Dumer, “Long nonbinary codes exceeding the Gilbert-Varshamov bound for any fixed distance”. IEEE Transactions on Information Theory, 50(10), pp. 2357-2362, 2004.
[24] G. Zemor, G. Cohen, “Error-correcting WOM-codes”. IEEE Transactions on Information Theory, 37(3), pp. 730-734, 1991.