Rewriting Flash Memories by Message Passing

Eyal En Gad, **Wentao Huang**, **Yue Li** and **Jehoshua Bruck**
California Institute of Technology, Pasadena, CA 91125
{eengad,whuang,yli,bruck}@caltech.edu

***Abstract*—This paper constructs WOM codes that combine rewriting and error correction for mitigating the reliability and the endurance problems in flash memory. We consider a rewriting model that is of practical interest to flash applications where only the second write uses WOM codes. Our WOM code construction is based on binary erasure quantization with LDGM codes, where the rewriting uses message passing and has potential to share the efficient hardware implementations with LDPC codes in practice. We show that the coding scheme achieves the capacity of the rewriting model. Extensive simulations show that the rewriting performance of our scheme compares favorably with that of polar WOM code in the rate region where high rewriting success probability is desired. We further augment our coding schemes with error correction capability. By drawing a connection to the conjugate code pairs studied in the context of quantum error correction, we develop a general framework for constructing error-correction WOM codes. Under this framework, we give an explicit construction of WOM codes whose codewords are contained in BCH codes.**

I. Introduction

Flash memory has become a leading storage media thanks to its many excellent features such as random access and high storage density. However, it also faces significant reliability and endurance challenges. In flash memory, programming cells with lower charge levels to higher levels can be done efficiently, while the opposite requires erasing the whole block containing millions of cells. Block erasure degrades cell quality, and current flash memory can survive only a small number of block erasures. To mitigate the reliability and the endurance issues, this paper studies write-once memory (WOM) codes that combine erasure-free information rewriting and error correction.

WOM was first studied by Rivest and Shamir [18]. In the model of WOM, new information is written by only increasing cell levels. Compared to traditional flash, WOM-coded flash achieves higher reliability when the same amonut of information is written, or writes more information using the same number of program/erase (P/E) cycles. We illustrate these benefits using Fig. 1, where we show the bit error rates (BERs) of the first write and the next rewrite measured for the scheme of this paper in a 16nm flash chip. When using the standard setting for error correcting codes (ECCs), flash memory can survive 14000 P/E cycles without an ECC decoding failure. Using a code constructed in this paper that allows user to write 35% more information, we only need 10370 P/E cycles to write the information. Notice that the raw BER at 10370 P/E cycles is much lower than that at 14000 P/E cycles, hence ECC decoding will have much lower failure rate, which leads to higher reliability. On the other hand, if we use WOM until ECC fails at 14000 P/E cycles, the total amount information that is written requires 18900 P/E cycles to write in traditional flash. WOM codes can also be used for *scrubbing* the memory. In this use, the memory is read periodically, to correct errors

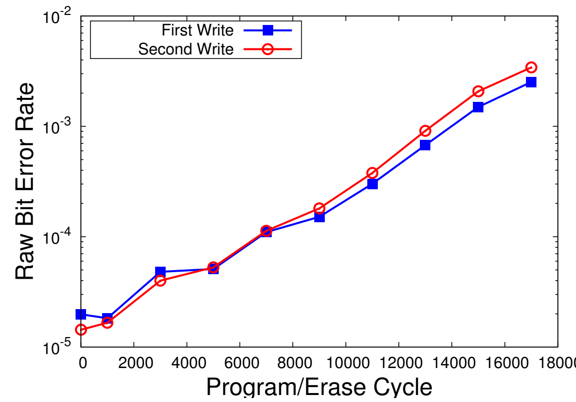

Fig. 1. The raw BERs when using the proposed rewriting scheme.

that were introduced over time. The errors are corrected using an ECC, and the corrected data is written back using a WOM code (see [13]). Many WOM constructions were proposed recently. Codes with higher rates were discovered [12][20], and codes that achieve capacity have also been found [1]. In this paper, we propose an alternative construction of WOM codes. Our scheme differs from the WOM codes mentioned above mainly in two aspects. First, we focus on a specific rewriting model with two writes, where only the second write uses WOM codes. Such rewriting scheme has no code rate loss in the first write, and recent experimental study has demonstrated its effectiveness on improving the performance of solid state drives [22]. Note that, the model of this rewriting scheme is not only an instance of the general WOM model [8], but also an instance of the model studied by Gelfand and Pinsker [5]. Second, our construction is based on binary erasure quantization with low-density-generator-matrix (LDGM) codes. The encoding is performed by iterative quantization studied by Martinian and Yedidia [15], which is a message-passing algorithm similar to the decoding of low-density-parity-check (LDPC) codes. As LDPC codes have been widely adopted by commercial flash memory controllers, the hardware architectures of message-passing algorithms have been well understood and highly optimized in practice. Therefore, our codes are implementation-friendly for practitioners. Extensive simulations show that the rewriting performance of our scheme compares favorably with that of the capacity-achieving polar WOM code [1] in the rate region where a low rewriting failure rate is desired. For instance, we show that our code allows user to write 40% more information by rewriting with very high success probability. We note that the iterative quantization algorithm of [15] was used in [2] in a different way for the problem of information embedding, which share some similarity with our model.

Moreover, our code construction is extended with error correction. The need for error correction is observed in our experiments. As shown in Fig. 1, the BERs of both writes increase rapidly with the number of block erasures. Constructions of error-correcting WOM codes have been studied in recent literature. Error-correcting WOM codes have been proposed in [3][4][11][21][23]. Different from the existing constructions above, we use conjugate code pairs studied in the context of quantum error correction [7]. As an example, we construct LDGM WOM codes whose codewords also belong to BCH codes. Therefore, our codes allows to use any decoding algorithm of BCH codes. The latter have been implemented in most commercial flash memory controllers. We also present two additional approaches to add error correction, and compare their performance.

II. Rewriting and Erasure Quantization

A. Rewriting Model

We consider a model that allows two writes on a block of n cells. A cell has a binary state chosen from $\{0,1\}$, with the rewriting constraint that state 1 can be written to state 0, but not vice versa. All cells are initially set to be in state 1, and so there is no writing constraint for the first write. A vector is denoted by a bold symbol, such as $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$. The state of the n cells after the first write is denoted by the vector $\boldsymbol{s}$. We focus only on the second write, and we assume that after the first write, the state of the cells is i.i.d., where for each i, $\Pr\{s_i = 1\} = \beta$. We note that the special case of $\beta = 1/2$ is of practical importance, since it approximates the state after a normal page programming in flash memory[1]. The second write is concerned with how to store a message $\boldsymbol{m} \in \mathbb{F}_2^k$ by changing $\boldsymbol{s}$ to a new state $\boldsymbol{x}$ such that 1) the rewriting constraint is satisfied, and 2) $\boldsymbol{x}$ represents $\boldsymbol{m}$. This is achieved by the encoding operation of a rewriting code, defined formally in the following.

Definition 1. *A* rewriting code *C_R is a collection of disjoint subsets of $\mathbb{F}_2^n$.*

Each element of C_R corresponds to a different message. Consider $M \in C_R$ that corresponds to a message $\boldsymbol{m}$, then for all $\boldsymbol{x} \in M$, we say that $\boldsymbol{x}$ is *labeled* by $\boldsymbol{m}$. The decoding function maps the set of labeled vectors into their labels, which are also the messages. To encode a message $\boldsymbol{m}$ given a state $\boldsymbol{s}$, the encoder needs to find a vector $\boldsymbol{x}$ with label $\boldsymbol{m}$ that can be written over $\boldsymbol{s}$. If the encoder does not find such vector $\boldsymbol{x}$, it declares a failure. The rewriting rate of C_R is defined by $R_{\text{WOM}} = k/n$. The rewriting capacity, which characterizes the maximum amount of information that can be stored per cell in the second write, is known to be β bits [8].

We are interested in rewriting codes with rates close to the capacity, together with efficient encoding algorithms with low failure probability. The main observation in the design of the proposed rewriting scheme of this paper is that the rewriting problem is related to the problem of binary erasure quantization (BEQ), introduced in the next subsection.

[1] In flash memory, the message to be written can be assumed to be random due to data compression and data randomization used in memory controllers.

B. Binary Erasure Quantization

The BEQ problem is concerned with the quantization of a binary *source sequence* $\boldsymbol{s}'$, for which some bits are erased. Formally, $\boldsymbol{s}' \in \{0,1,*\}^n$, where $*$ represents erasures. $\boldsymbol{s}'$ needs to be quantized (compressed) such that every non-erased symbol of $\boldsymbol{s}'$ will maintain its value in the reconstructed vector. A reconstructed vector with such property is said to have *no distortion* from $\boldsymbol{s}'$. In this paper we use linear BEQ codes, defined as follows:

Definition 2. *A linear BEQ code C_Q is a subspace of $\mathbb{F}_2^n$. Each $\boldsymbol{c} \in C_Q$ is called a codeword of C_Q. The dimension of C_Q is denoted by r.*

Each codeword of C_Q is labeled by a different r-bits sequence $\boldsymbol{u}$. Given a BEQ code C_Q and a source sequence $\boldsymbol{s}'$, a quantization algorithm Q is invoked to find a label $\boldsymbol{u}$ whose codeword $\boldsymbol{c} \in C_Q$ has *no distortion* from $\boldsymbol{s}'$. If such label is found, it is denoted by $\boldsymbol{u} = Q(\boldsymbol{s}')$, and is considered as the compressed vector. Otherwise, a quantization failure is declared, and $Q(\boldsymbol{s}') = \mathit{Failure}$. The reconstruction uses a generator matrix G_Q of C_Q to obtain the codeword $\boldsymbol{c} = \boldsymbol{u}G_Q$.

C. Reduction from Rewriting to Erasure Quantization

In this subsection we show that the problem of rewriting can be efficiently reduced to that of BEQ. Let C_Q be a linear quantization code, and let H be a parity-check matrix of C_Q.

Construction 3. *A rewriting code C_R is constructed as the collection of all cosets of C_Q in $\mathbb{F}_2^n$. A decoding function for C_R is defined by a parity check matrix H of C_Q, such that a vector $\boldsymbol{x} \in \mathbb{F}_2^n$ is decoded into its syndrome*

$$DEC_H(\boldsymbol{x}) = \boldsymbol{x}H^T. \tag{1}$$

Since the dimension of C_Q is r, it has 2^{n-r} cosets. Therefore the rate of C_R is $R_{\text{WOM}} = \frac{n-r}{n}$, implying that $k = n - r$. We define some notation before introducing the reduction algorithm. Let $(H^{-1})^T$ be a left inverse for H^T, meaning that $(H^{-1})^T H^T$ is the $k \times k$ identity matrix. Define a function $BEC : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}^n$ as:

$$BEC(\boldsymbol{w},\boldsymbol{v})_i = \begin{cases} w_i & \text{if } v_i = 0 \\ * & \text{if } v_i = 1 \end{cases} \quad ,\forall i = 1,...,n$$

$BEC(\boldsymbol{w},\boldsymbol{v})$ realizes a binary erasure channel that erases entries in $\boldsymbol{w}$ whose corresponding entries in $\boldsymbol{v}$ equal 1. We are now ready to introduce the encoding algorithm for the rewriting problem.

Theorem 4. *Algorithm 1 either declares a failure or returns a vector $\boldsymbol{x}$ such that $\boldsymbol{x}$ is rewritable over $\boldsymbol{s}$ and $\boldsymbol{x}H^T = \boldsymbol{m}$.*

Proof: Suppose failure is not declared and $\boldsymbol{x}$ is returned by Algorithm 1. We first prove that $\boldsymbol{x}$ is rewritable over $\boldsymbol{s}$. Consider i such that $s_i = 0$. Then it follows from the definition of BEC that $s'_i = z_i$. Remember that $Q(\boldsymbol{s}')$ returns a label $\boldsymbol{u}$ such that $\boldsymbol{c} = \boldsymbol{u}G_Q$ has no-distortion from $\boldsymbol{s}'$. Therefore, $c_i = s'_i = z_i$, and $x_i = c_i + z_i = z_i + z_i = 0 = s'_i$. So $\boldsymbol{x}$

Algorithm 1 $x = ENC(G_Q, \boldsymbol{m}, \boldsymbol{s})$: Encoding for Rewriting

```
z ← m(H^{-1})^T
s' ← BEC(z, s)
u ← Q(s')
if u = FAILURE then
    return FAILURE
else
    return x ← uG_Q + z
end if
```

can be written over $\boldsymbol{s}$. To prove the second statement of the theorem, notice that

$$\begin{aligned} \boldsymbol{x}H^T &= (\boldsymbol{u}G_Q + \boldsymbol{z})H^T = \boldsymbol{u}G_Q H^T + \boldsymbol{m}(H^{-1})^T H^T \\ &= \boldsymbol{m}(H^{-1})^T H^T = \boldsymbol{m}. \end{aligned}$$

■

III. Rewriting with Message Passing

In this section we discuss how to choose a quantization code C_Q and quantization algorithm Q to obtain a rewriting scheme of good performance. Our approach is to use the iterative quantization scheme of Martinian and Yedidia [15], where C_Q is an LDGM code, and Q is a message-passing algorithm. This approach is particularly relevant for flash memories, since the hardware architecture of message-passing algorithms is well understood and highly optimized in flash controllers.

The algorithm Q can be implemented by a sequential or parallel scheduling, as described in [15, Section 3.4.2]. For concreteness, we consider the sequential algorithm denoted by ERASURE-QUANTIZE in [15]. Since the performance of ERASURE-QUANTIZE depends on the chosen generator matrix, we abuse notation and denote it by $Q(G_Q, \boldsymbol{s}')$. Algorithm $Q(G_Q, \boldsymbol{s}')$ is presented in Appendix A, for completeness.

Finally, we need to describe how to choose a generator matrix G_Q that work well together with Algorithm Q. We show next that a matrix G_Q with good rewriting performance can be chosen to be a *parity-check matrix* that performs well in message-passing decoding of erasure channels. This connection follows from the connection between rewriting and quantization, together with a connection between quantization and erasure decoding, shown in [15]. These connections imply that we can use the rich theory and understanding of the design of parity-check matrices in iterative erasure decoding, to construct good generating matrices for rewriting schemes. To make the statement precise, we consider the standard iterative erasure-decoding algorithm denoted by ERASURE-DECODE$(H, \boldsymbol{y})$ in [15], where H is an LDPC matrix and $\boldsymbol{y}$ is the output of a binary erasure channel.

Theorem 5. *For all $\boldsymbol{m} \in \mathbb{F}_2^k$ and $\boldsymbol{z}', \boldsymbol{s} \in \mathbb{F}_2^n$, $ENC(G_Q, \boldsymbol{m}, \boldsymbol{s})$ fails if and only if ERASURE-DECODE$(G_Q, BEC(\boldsymbol{z}', \boldsymbol{s} + \boldsymbol{1}_n))$ fails, where $\boldsymbol{1}_n$ is the all-one vector of length n.*

The proof of Theorem 5 is available in Appendix B. The running time of the encoding algorithm ENC is analyzed formally in the following theorem.

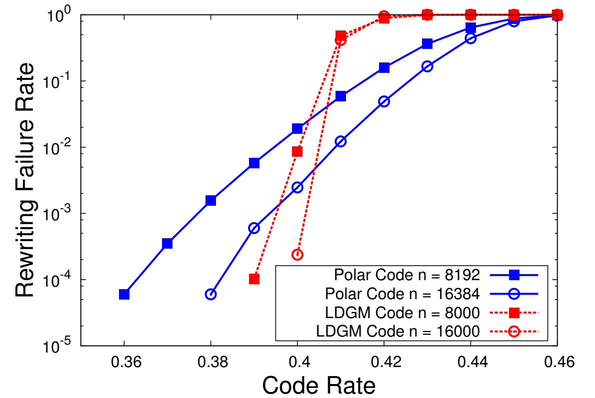

Fig. 2. Rewriting failure rates of polar and LDGM WOM codes.

Theorem 6. *The algorithm $ENC(G_Q, \boldsymbol{m}, \boldsymbol{s})$ runs in time $\mathcal{O}(nd)$ where n is the length of $\boldsymbol{s}$ and d is the maximum degree of the Tanner graph of G_Q.*

The proof of Theorem 6 is available in Appendix C. Theorems 5 and 6, together with the analysis and design of irregular LDPC codes that achieve the capacity of the binary erasure channel [17], imply the following capacity-achieving results.

Corollary 7. *There exists a sequence of rewriting codes which can be efficiently encoded by Algorithm* 1 *and efficiently decoded by Equation* (1) *that achieves the capacity of the rewriting model β.*

The proof of Corollary 7 is available in Appendix D.

The finite-length performance of our rewriting scheme is evaluated using extensive simulation with the choice of $\beta = 0.5$ and G_Q to be the parity-check matrix of a Mackay code [14]. The rewriting failure rates of our codes with lengths $n = 8000$ and 16000 that are relevant to flash applications are compared with those of the polar WOM codes of lengths 2^{13} and 2^{14} [1]. Fig. 2 shows the rewriting failure rates of both codes at different rewriting rate, where each point is calculated from 10^5 experiments. Remember that the capacity of the model is 0.5. The results suggest that our scheme achieves a decent rewriting rate (e.g. 0.39) with low failure rate (e.g. $< 10^{-4}$). Moreover, our codes provide significantly lower failure rates than polar WOM codes when the rewriting rate is smaller, because of the good performance in the waterfall region of message-passing algorithm.

IV. Error-Correcting Rewriting Codes

The construction of error-correcting rewriting codes is based on a pair of linear codes (C_1, C_Q), that satisfies the condition $C_1 \supseteq C_Q$, meaning that each codeword of C_Q is also a codeword of C_1. Define C_2 to be the dual of C_Q, denoted by $C_2 = C_Q^{\perp}$. A pair of linear codes (C_1, C_2), that satisfies $C_1 \supseteq C_2^{\perp}$ is called *a conjugate code pair*, and it is useful in quantum error correction and cryptography [7]. For the flash memory application, we let C_1 be an error-correction code, while $C_2^{\perp} = C_Q$ is a BEQ code. The main idea in the construction of error-correcting rewriting codes is to label *only* the codewords of C_1, according to their membership in the cosets of C_Q. The construction is defined formally as follows:

Construction 8. *For $\boldsymbol{c} \in C_1$, let $\boldsymbol{c} + C_Q$ be the coset of C_Q in C_1 that contains $\boldsymbol{c}$. Then the error-correcting rewriting code is constructed to be the collection of cosets of C_Q in C_1.*

Next we define the matrices $(H^{-1})^T$ and H^T to be used in encoding and decoding. Let G_1 and G_Q be generator matrices of the codes C_1 and C_Q, respectively, such that each row of G_Q is also a row of G_1. Since C_1 contains C_Q, such matrix pair always exists. Define $(H^{-1})^T$ to be constructed by the rows of G_1 that are *not* rows of G_Q. Let H^T be a right inverse of $(H^{-1})^T$.

The encoding is performed according to Algorithm 1, with the matrix $(H^{-1})^T$ defined above. Note that in Step 1, $\boldsymbol{z}$ is a codeword of C_1, since each row of $(H^{-1})^T$ is also a row of G_1. In addition, in Step 7, $\boldsymbol{u}G_Q$ is also a codeword of C_1 (unless $Q(G_Q, \boldsymbol{s}')$ fails), since C_Q is contained in C_1. Therefore, $\boldsymbol{x} = \boldsymbol{u}G_Q + \boldsymbol{z}$ is a codeword of C_1. The decoding can begin by the recovery of $\boldsymbol{x}$ from its noisy version, using the decoder of C_1. The message $\boldsymbol{m}$ can then be recovered by the product $\boldsymbol{x}H^T$.

A similar framework was described in [10], which proposed a construction of a repetition code contained in a Hamming code, with a Viterbi encoding algorithm. In this paper we make the connection to the quantum coding literature, which allows us to construct stronger codes.

A. Conjugate Codes Construction

We look for a conjugate pair (C_1, C_2) such that C_1 is a good error-correcting code, while $C_2^{\perp}$ is a good LDGM quantization code. Theorem 5 implies that C_2 needs to be an LDPC code with a good performance over a binary erasure channel (under message passing decoding). Constructions of conjugate code pairs in which C_2 is an LDPC code are studied in [6][9][19]. Sarvepalli *et al.* [19] construct a pair of codes such that C_1 is a BCH code and C_2 is a Euclidean geometry LDPC code, which is particularly useful for our purpose. This is because BCH codes are used extensively for error correction in flash memories. Below we first briefly review the construction of Euclidean geometry LDPC codes and then discuss the application of the results in [19] to our settings.

Denote by $\text{EG}(m, p^s)$ the Euclidean finite geometry over $\mathbb{F}_{p^s}$ consisting of p^{ms} points. Note that this geometry is equivalent to the vector space $\mathbb{F}_{p^s}^m$. A μ-dimensional subspace of $\mathbb{F}_{p^s}^m$ or its coset is called a *μ-flat*. Let J be the number of μ-flats that do not contain the origin, and let $\alpha_1, ...\alpha_{p^{sm}-1}$ be the points of $\text{EG}(m, p^s)$ excluding the origin. Construct a $J \times p^{sm} - 1$ matrix H_{EG} in the way that its (i, j)-th entry equals 1 if the i-th μ-flat contains α_j, and equals 0 otherwise. H_{EG} is the parity check matrix of the (Type-I) Euclidean geometry LDPC code $C_{EG}(m, \mu, s, p)$. $C_{EG}(m, \mu, s, p)$ is a cyclic code and by analyzing the roots of its generator polynomial, the following result is obtained [19].

Proposition 9 *$C_{EG}^{\perp}(m, \mu, s, p)$ is contained in a BCH code of design distance $\delta = p^{\mu s} - 1$.*

Hence we may choose C_2 to be $C_{EG}(m, \mu, s, p)$ and C_1 to be a BCH code with distance equal to or smaller than δ. Some possible code constructions are shown in Table I. Their

TABLE I
ERROR-CORRECTING REWRITING CODES CONSTRUCTED FROM PAIRS OF CONJUGATE BCH AND EG-LDPC CODES.

(m, μ, s, p)	$C_1[n, k, \delta]$	$C_2[n, k]$	Rewriting Rate
(4,1,2,2)	[255,247,3]	[255,21]	0.0510
(3,1,2,2)	[65,57,3]	[65,13]	0.1111
(3,1,3,2)	[511,484,7]	[511,139]	0.2192
(3,1,4,2)	[4095,4011,15]	[4095,1377]	0.3158

encoding performance, with respect to the probability β that a cell in the state is writable, is shown in Fig. 3. Note from Fig. 3 that a code with smaller rewriting rate achieves a fixed failure rate at a smaller value of β. In particular, the codes corresponding to the top three rows of Table I achieve very small failure rate at $\beta = 0.5$, the point of practical interest. These results also show that the slope of the figures becomes sharper when the length of the codes increases, as expected. Out of the three codes that can be rewritten with $\beta = 0.5$, $C_{EG}(3, 1, 3, 2)$ poses the best rate and error-correction capability.

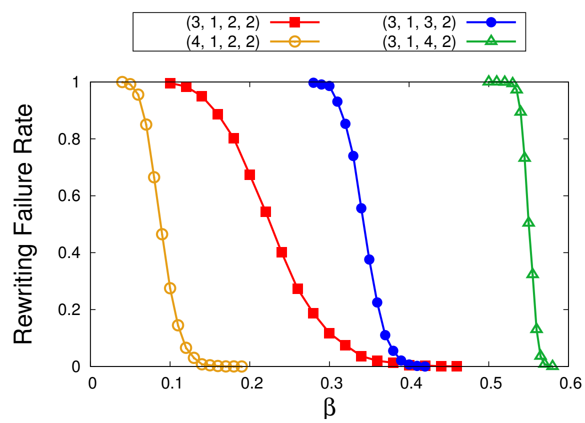

Fig. 3. Encoding performance of the codes in Table I.

V. ALTERNATIVE APPROACHES FOR ERROR CORRECTION

In this section we present two alternative approaches to combine rewriting codes with error correction.

A. Concatenated Codes

In this scheme, we concatenate an LDGM rewriting code with a systematic error-correcting code. The outer code is an LDGM rewriting code without error-correction capability, as in Section III. The systematic ECC is used as the inner code. The concatenated scheme is used in the second write. The scheme requires the first write to *reserve* some bits to store the redundancy of the ECC in the second write.

In the second write, the encoder begins by finding a vector $\boldsymbol{x}$ that can be written over the current state. After $\boldsymbol{x}$ is written, the systematic ECC calculates the redundancy bits required to protect $\boldsymbol{x}$ from errors. The redundancy bits are then written into the reserved cells. The decoding of the second write begins by recovering $\boldsymbol{x}$ using the systematic ECC and its redundancy bits. After $\boldsymbol{x}$ is recovered, the decoder of the rewriting code recovers the stored message from $\boldsymbol{x}$.

We note that reserving bits for the second write have a negative effect on the performance of the system, since it reduces the total amount of information that could be stored in the memory on a given time. Therefore, the next subsection extends the concatenation scheme using a chaining technique, with the aim of reducing the number of bits required to be reserved for the second write.

B. Code Chaining

The chaining approach is inspired by a similar construction in polar coding [16]. The idea is to chain several code blocks of short length. In the following we use a specific example to demonstrate the idea. We use a BCH code for error correction, since its performance can be easily calculated. We note, however, that LDPC codes may be used in practice, such that the circuit modules may be shared with the rewriting code, to reduce the required area. The performance of LDPC code in the considered parameters is similar to that of BCH codes.

A typical BCH code used in flash memory has the parameters $[8191, 7671, 81]$, where the length is 8191, the dimension is 7671, and the minimum distance is 81. If this code is used in a concatenated scheme for the second write, the first write needs to reserve $8191 - 7671 = 520$ bits for redundancy.

To reduce the amount of required reserved bits, we consider the chaining of 8 systematic BCH codes with the parameters $[1023, 863, 33]$. The encoding is performed sequentially, beginning with the rewriting encoding that finds a vector $\boldsymbol{x}_1$ of 863 bits. The vector $\boldsymbol{x}_1$ represents a message $\boldsymbol{m}_1$ of 310 bits, according to an $[863, 310]$-LDGM rewriting code. Once $\boldsymbol{x}_1$ is found, the BCH encoder finds $1023 - 863 = 160$ redundancy bits to protect $\boldsymbol{x}_1$, as in the concatenated scheme. The encoder then "chains" the redundancy bits forward, by encoding them, together with 150 new information bits, into another block of 863 bits, using the $[863, 310]$-LDGM code. Let $\boldsymbol{m}_2$ denote the vector of 310 bits encoded into the second block. $\boldsymbol{m}_2$ contains the 160 redundancy bits of $\boldsymbol{x}_1$, together with the additional 150 information bits. Note that once $\boldsymbol{m}_2$ is decoded, the redundancy bit of $\boldsymbol{x}_1$ are available, allowing the recovery $\boldsymbol{x}_1$, and then $\boldsymbol{m}_1$. The encoding continues in this fashion 8 times, to write over a total of 8 blocks, each containing 863 cells. The 160 redundant bits used to protect $\boldsymbol{x}_8$ are stored in the reserved cells. The decoding is done in the reverse order, where each decoded vector contains the redundancy bits of the previous block.

C. Comparison

We compare the different error-correction approaches, and discuss their trade-offs. The first code in the comparison is a conjugate code pair, described in Section IV. We use a conjugation of a $[511, 484, 7]$-BCH code containing a $[511, 372]$-LDGM code, dual to the $(3, 1, 3, 2)$-Euclidean geometry LDPC code in Table I. The second code in the comparison is a concatenation of an outer $[7671, 2915]$-LDGM Mackay rewriting code with an inner $[8191, 7671, 81]$-BCH code. The third code is a chaining of 8 blocks of $[863, 310]$-LDGM Mackay codes, each concatenated with a $[1023, 863, 33]$-BCH code. We compare the decoding BER P_D, the fraction α of bits required to be reserved, and the rewriting rate R_{WOM} of the codes. The encoding failure rate of each of the three codes for $\beta = 0.5$ is below 10^{-3}. P_D is estimated with a standard flash memory assumption of a raw BER of 1.3×10^{-3}. To achieve a comparable code length, the conjugated code is assumed to be used 16 times in parallel, with a total length of $511 \times 16 = 8176$. The comparison is summarized in Table II.

TABLE II
ERROR-CORRECTING REWRITING CODES OF LENGTH ≈ 8200.

Code	P_D	α	R_{WOM}
Conjugated	10^{-5}	0%	0.21
Concatenated	10^{-16}	6.3%	0.35
Chained	10^{-16}	2%	0.19

Flash systems require P_D below 10^{-15}. We see in Table II that conjugated code still do not satisfy the reliability requirement. We also see that concatenated codes that satisfy the reliability requirement need a large fraction of reserved space. The chained code reduces the fraction of reserved space to 2%, with a rate penalty in the second write.

References

[1] D. Burshtein and A. Strugatski, "Polar write once memory codes," *IEEE Trans. Inf. Theor.*, vol. 59, no. 8, pp. 5088–5101, Aug. 2013.

[2] V. Chandar, E. Martinian, and G. W. Wornell, "Information embedding codes on graphs with iterative encoding and decoding," in *Proc. ISIT*, July 2006, pp. 866–870.

[3] E. En Gad *et al.*, "Polar coding for noisy write-once memories," in *Proc. ISIT*, June 2014, pp. 1638–1642.

[4] A. Gabizon and R. Shaltiel, "Invertible zero-error dispersers and defective memory with stuck-at errors," in *APPROX-RANDOM*, 2012, pp. 553–564.

[5] S. Gel'fand and M. Pinsker, "Coding for channel with random parameters," *Problems of Control Theory*, vol. 9, no. 1, pp. 19–31, 1980.

[6] M. Hagiwara and H. Imai, "Quantum quasi-cyclic ldpc codes," in *Proc. ISIT*, June 2007, pp. 806–810.

[7] M. Hamada, "Conjugate codes for secure and reliable information transmission," in *Proc. ITW*, Oct 2006, pp. 149–153.

[8] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inf. Theor.*, vol. 31, no. 1, pp. 34–42, January 1985.

[9] L. Ioffe and M. Mézard, "Asymmetric quantum error-correcting codes," *Phys. Rev. A*, vol. 75, p. 032345, Mar 2007.

[10] A. Jacobvitz, R. Calderbank, and D. Sorin, "Writing cosets of a convolutional code to increase the lifetime of flash memory," in *Proc. Allerton*, Oct 2012, pp. 308–318.

[11] A. Jiang *et al.*, "Joint rewriting and error correction in write-once memories," in *Proc. ISIT*, 2013, pp. 1067–1071.

[12] A. Jiang, M. Langberg, M. Schwartz, and J. Bruck, "Trajectory codes for flash memory," *IEEE Trans. Inf. Theor.*, vol. 59, no. 7, pp. 4530–4541, July 2013.

[13] Y. Li, A. Jiang, and J. Bruck, "Error correction and partial information rewriting for flash memories," in *Proc. ISIT*, June 2014, pp. 2087–2091.

[14] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theor.*, vol. 45, no. 2, pp. 399–431, Mar 1999.

[15] E. Martinian and J. S. Yedidia, "Iterative quantization using codes on graphs," in *Proc. Allerton*, 2003.

[16] M. Mondelli, S. Hassani, R. Urbanke, and I. Sason, "Achieving Marton's region for broadcast channels using polar codes," in *Proc. ISIT*, June 2014, pp. 306–310.

[17] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theor.*, vol. 48, no. 12, pp. 3017–3028, Dec 2002.

[18] R. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and Control*, vol. 55, no. 1-3, pp. 1–19, 1982.

[19] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, "Asymmetric quantum codes: constructions, bounds and performance," *Proc. Roy. Soc. of Lond. A Mat.*, vol. 465, no. 2105, pp. 1645–1672, 2009.

[20] E. Yaakobi *et al.*, "Codes for write-once memories," *IEEE Trans. Inf. Theor.*, vol. 58, no. 9, pp. 5985–5999, September 2012.

[21] E. Yaakobi, P. Siegel, A. Vardy, and J. Wolf, "Multiple error-correcting wom-codes," *IEEE Trans. Inf. Theor.*, vol. 58, no. 4, pp. 2220–2230, April 2012.

[22] G. Yadgar, E. Yaakobi, and A. Schuster, "Write once, get 50% free: Saving ssd erase costs using wom codes," in *USENIX FAST*, 2015.

[23] G. Zemor and G. D. Cohen, "Error-correcting wom-codes," *IEEE Trans. Inf. Theor.*, vol. 37, no. 3, pp. 730–734, May 1991.

Appendix A
Iterative Quantization Algorithm

We denote $G_Q = (g_1, \ldots, g_n)$ such that g_j is the j-th column of G_Q.

Algorithm 2 $\boldsymbol{u} = Q(G_Q, \boldsymbol{s}')$.

$\boldsymbol{v} \leftarrow \boldsymbol{s}'$
while $\exists j$ such that $v_j \neq *$ **do**
  if $\exists i$ such that $\exists! j$ for which $G_Q(i,j) = 1$ and $v_j \neq *$ **then**
    Push (i,j) into the Stack.
    $v_j \leftarrow *$.
  else
    return FAILURE
  end if
end while
$\boldsymbol{u} \leftarrow \boldsymbol{0}_{n-k}$
while Stack is not empty **do**
  Pop (i,j) from the Stack.
  $u_i \leftarrow \boldsymbol{u} \cdot \boldsymbol{g}_j + s'_j$
end while
return $\boldsymbol{u}$

Appendix B
Proof of Theorem 5

Proof: As in Algorithm 1, let $\boldsymbol{z} = \boldsymbol{m}(H^{-1})^T$ and $\boldsymbol{s}' = \mathrm{BEC}(\boldsymbol{z}, \boldsymbol{s})$. Now according to Algorithm 1, $\mathrm{ENC}(G_Q, \boldsymbol{m}, \boldsymbol{s})$ fails if and only if $Q(G_Q, \boldsymbol{s}')$ fails. According to [15, Theorem 4], $Q(G_Q, \boldsymbol{s}')$ fails if and only if ERASURE-DECODE$(G_Q, \mathrm{BEC}(\boldsymbol{z}', \boldsymbol{s} + \boldsymbol{1}_n))$ fails. This completes the proof. ■

Appendix C
Proof of Theorem 6

Proof: We first show that Step 1 of Algorithm 1 runs in time $\mathcal{O}(n)$ if $(H^{-1})^T$ is chosen in the following way. For any C_Q, its parity check matrix H can be made in to systematic form, i.e., $H = (P\ I)$, by row operations and permutation of columns. Then $(H^{-1})^T$ can be chosen as $(\boldsymbol{0}_{k \times n-k}\ I_k)$, and so $\boldsymbol{z} = \boldsymbol{m}(H^{-1})^T = (\boldsymbol{0}_{n-k}\ \boldsymbol{m})$.

By [15, Theorem 5], Step 3 of Algorithm 1 runs in time $\mathcal{O}(nd)$. By the definition of d, the complexity of Step 7 is also $\mathcal{O}(nd)$. Therefore $\mathcal{O}(nd)$ dominates the computational cost of the algorithm. ■

Appendix D
Proof of Corollary 7

Proof: Let $\bar{\boldsymbol{s}} = \boldsymbol{s} + \boldsymbol{1}_n$. Then it follows from Theorem 5 that for all G_Q, $\boldsymbol{m} \in \mathbb{F}_2^k$, $\boldsymbol{z}' \in \mathbb{F}_2^n$,

$$\Pr\{ENC(G_Q, \boldsymbol{m}, \boldsymbol{s}) = Failure\} = \Pr\{\textsf{ERASURE-DECODE}(G_Q, \mathrm{BEC}(\boldsymbol{z}', \bar{\boldsymbol{s}})) = Failure\},$$

where $\boldsymbol{s}$ is distributed i.i.d. with $\Pr\{s_i =\} = \beta$. The right-hand side is the decoding-failure probability of an LDPC code with parity-check matrix G_Q over a binary erasure channel, using message-passing decoding. The erasure probability of the channel is $1 - \beta$, because $\Pr\{\bar{s}_i = 1\} = 1 - \Pr\{s_i = 1\}$. The capacity of a binary erasure channel with erasure probability $1 - \beta$ is β. This is also the capacity of the rewriting model. In addition, the rate of an LDPC code with parity-check matrix G_Q is equal to the rate of a rewriting code constructed by the cosets of C_Q. It is shown in [17] how to construct a sequence of irregular LDPC codes that achieves the capacity of the binary erasure channel. Such sequence, used for rewriting codes, achieves the rewriting capacity. ■

Appendix E
Handling Encoding Failures

The encoding failure event could be dealt with in several ways. A simple solution is to try writing on different invalid pages, if available, or to simply write into a fresh page, as current flash systems do. If the failure rate is small enough, say below 10^{-3}, the time penalty of rewriting failures would be small. For an alternative solution, we state a reformulation of [15, Theorem 3].

Proposition 10. *For all $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}_2^k$ and $\boldsymbol{s} \in \mathbb{F}_2^n$, $ENC(G_Q, \boldsymbol{m}, \boldsymbol{s})$ fails if and only if $ENC(G_Q, \boldsymbol{m}', \boldsymbol{s})$ fails.*

Proof: As in Algorithm 1, let $\boldsymbol{z} = \boldsymbol{m}(H^{-1})^T$ and $\boldsymbol{s}' = \mathrm{BEC}(\boldsymbol{z}, \boldsymbol{s})$. Note that $\mathrm{ENC}(G_Q, \boldsymbol{m}, \boldsymbol{s})$ fails if and only if $Q(G_Q, \boldsymbol{s}')$ fails. By Algorithm 2, the failure of $Q(G_Q, \boldsymbol{s}')$ is determined only according to the locations of erasures in $\boldsymbol{s}'$, and does not depend on the values of the non-erased entries of $\boldsymbol{s}'$. Since $\boldsymbol{s}' = \mathrm{BEC}(\boldsymbol{z}, \boldsymbol{s})$, the locations of erasures in $\boldsymbol{s}'$ are only determined by the state $\boldsymbol{s}$. This completes the proof. ■

Proposition 10 implies that whether a page is rewritable does not depend on the message to be written. This property suggests that the flash controller can check whether a page is rewritable right after it is being invalidated, without waiting for a message to arrive. An invalid page could be marked as 'unrewritable', such that data would be rewritten only into rewritable pages. This policy would guarantee that the rewriting of a new message always succeed. However, this policy also implies that the message passing algorithm would run more than once for the rewriting of a page.