Error Correction and Partial Information Rewriting for Flash Memories

Yue Li[†*], **Anxiao (Andrew) Jiang**[†], and **Jehoshua Bruck**[*]
[†]*Texas A&M University, College Station, TX 77843, USA*
[*]*California Institute of Technology, Pasadena, CA 91125, USA*
{*yli, bruck*}*@caltech.edu* *ajiang@cse.tamu.edu*

***Abstract*—This paper considers the partial information rewriting problem for flash memories. In this problem, the state of information can only be updated to a limited number of new states, and errors may occur in memory cells between two adjacent updates. We propose two coding schemes based on the models of trajectory codes. The bounds on achievable code rates are shown using polar WOM coding. Our schemes generalize the existing rewriting codes in multiple ways, and can be applied to various practical scenarios such as file editing, log-based file systems and file synchronization systems.**

I. Introduction

Flash memories are asymmetric in the sense that programming a cell from 0 to 1 is efficient while bringing a cell from 1 to 0 needs to erase a whole block of cells, which is costly and degrades the quality of cells. Moreover, to achieve higher storage density, the geometry of flash chips continuously shrink, making data more prone to noise. This paper studies codes that combine error correction and rewriting for mitigating the endurance and the reliability issues in flash memories.

A basic way to model flash memories is through write-once memory (WOM), where a cell level can be increased from low to high but not vice versa. The design of WOM codes has been extensively studied in literature. This includes linear codes, tabular codes, codes based on projective geometry, coset coding etc. High-rate codes have been developed [9], and codes that achieve capacity were proposed very recently [1] [8]. To make the WOM model more practical, the study on error correcting WOM (EC-WOM) codes has been initiated. Codes that correct a few errors (e.g. 1, 2, or 3) have been proposed [10] [11]. More recently, WOM codes that correct an arbitrary number of errors were developed [3] [6].

This paper follows the partial information rewriting model of trajectory codes [5], where the current information can be changed to a limited number of new states during each update.

Definition 1. *(Partial Rewriting) Let $\mathcal{G}(V,E)$ be a directed general rewriting graph that is strongly connected. Let each vertex $v \in V$ denote a message $M \in \{0,1\}^{\log_2|V|}$ and let $\pi : \{0,1\}^{\log_2|V|} \to V$ be a one-to-one mapping defined by $\pi(M) \triangleq v$. Let each edge $e \in E$ denote the change between the messages allowed by each update. Let D be the maximum out degree of each vertex, where $D \geqslant 1$. Partial rewriting stores a sequence of N messages $(M_0, \cdots, M_{N-1})$ such that*

(a) $\pi(M_j) \in V$ *for* $j \in \{0,1,\cdots,N-1\}$.
(b) $(\pi(M_j) \to \pi(M_{j+1})) \in E$ *for* $j \in \{0,1,\cdots,N-2\}$.

This work was supported in part by the NSF Grant CCF-1217944 and a grant from Intellectual Ventures.

The model of partial rewriting can be found in many practical storage applications such as file editing, log-based file systems and file synchronization systems. In such applications, data tend to be frequently updated while each update only makes small changes on the data. Partial rewriting increases the number of block erasures in flash memories and degrades memory performance. Note that a noiseless channel is assumed in the study of trajectory code for partial rewriting [5].

The contributions of this paper are general coding schemes for partial rewriting with noise, where errors from a binary symmetric channel may occur in cells between two adjacent updates. We propose two specific constructions based on trajectory codes. We show the bounds on achievable code rates based on our previous work on polar EC-WOM codes [6]. Our work generalize the existing rewriting codes in multiple ways.

II. Preliminary Concepts

This section introduces the noisy WOM model, which is an instance of the noisy partial rewriting model in this paper. It then revisits the polar EC-WOM codes [6] and the trajectory codes [5], which the codes of this paper are mainly based on.

A. The Model of Rewriting with Noise

A code for rewriting and error correction consists of t encoding functions $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_t$ and t decoding functions $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_t$. Let there be N binary cells. Let $[z] \triangleq \{1,2,\cdots,z\}$ for integer z. For $i \in [N]$ and $j \in [t]$, let $s_{i,j}, s'_{i,j} \in \{0,1\}$ denote the level of the i-th cell immediately before and after the j-th write, respectively. The WOM constraint requires for all i and j, $s'_{i,j} \geqslant s_{i,j}$. Let $c_{i,j} \in \{0,1\}$ denote the level of the i-th cell at any time after the j-th write and before the $(j+1)$-th write, when reading of the message M_j can happen. The error $c_{i,j} \oplus s'_{i,j} \in \{0,1\}$ is the error in the i-th cell caused by the binary symmetric channel denoted by $\mathrm{BSC}(p_e)$ with error probability p_e. (Here $\oplus$ is an XOR function.) For $j \in [t]$, the encoding function $\mathcal{E}_j$ changes the cell levels from $\mathbf{s}_j = (s_{1,j}, s_{2,j}, \cdots, s_{N,j})$ to $\mathbf{s}'_j = (s'_{1,j}, s'_{2,j}, \cdots, s'_{N,j})$ given the initial cell state $\mathbf{s}_j$ and the message to store M_j. (Namely, $\mathcal{E}_j(\mathbf{s}_j, M_j) = \mathbf{s}'_j$.) When the reading of M_j happens, the decoding function $\mathcal{D}_j$ recovers the message M_j given the noisy cell state $\mathbf{c}_j = (c_{1,j}, c_{2,j}, \cdots, c_{N,j})$. (Namely, $\mathcal{D}_j(\mathbf{c}_j) = M_j$.)

For $j \in [t]$, define *instantaneous rate* of the j-th write as $R_j \triangleq \frac{\mathcal{M}_j}{N}$, where $\mathcal{M}_j$ is the number of bits in M_j. The *sum-rate* is defined as $R_{\mathrm{sum}} \triangleq \sum_{j\in[t]} R_j$. When there is no noise, the maximum sum-rate (i.e. capacity) of WOM is known to be $\log_2(t+1)$ bits per cell. However, for the noisy WOM described above, the exact capacity is still largely unknown [4].

B. Polar EC-WOM Codes

Polar EC-WOM codes combine rewriting and error correction with efficient encoding and decoding algorithms [6]. The scheme extends the constructions by Burshtein and Strugatski [1], which is based on polar codes and achieves the capacity of noiseless WOM with techniques related to lossy source coding [7]. In [1], a WOM channel is designed for each write such that the WOM constraint is satisfied and the capacity of the channel can match the instantaneous rate. A polar code is then constructed for each WOM channel and used for encoding. The polar WOM code was extended to correct errors [6]. The code in [6] is shown to have good performance when the frozen sets of the polar codes respectively constructed for the encoding channel and the decoding channel have substantial overlapping, the existence of which is analyzed both analytically and experimentally.

C. Trajectory Codes

Trajectory codes are rewriting codes that are asymptotically optimal for noiseless partial rewriting [5]. Given the rewriting graph $\mathcal{G}(V,E)$, let $L=|V|$, divide a group of n binary cells into C subgroups. For $i \in \{0,1,\cdots,C-1\}$, let the i-th subgroup have n_i cells and be referred to as *register* r_i, namely, $n=\sum_{i=0}^{C-1} n_i$. A register stores a t-write WOM code. Let $\boldsymbol{s}_j=(s_{1,j},s_{1,j},\cdots,s_{n,j})$ and $\boldsymbol{s'}_j=(s'_{1,j},s'_{1,j},\cdots,s'_{n,j})$ be the cell states immediately before and after storing M_j. A trajectory code has Ct encoders $\mathbf{E}_0,\mathbf{E}_1,\cdots,\mathbf{E}_{Ct-1}$ and decoders $\mathbf{D}_0,\mathbf{D}_1,\cdots,\mathbf{D}_{Ct-1}$, supporting $N=Ct$ message updates. For $j\in\{0,1,\cdots,Ct-1\}$, the encoder

$$\mathbf{E}_j:\{0,1\}^n\times\{0,1\}^{\log_2 L}\times\mathcal{G}(V,E)\rightarrow\{0,1\}^n$$

computes the new cell states from the message, the current state and $\mathcal{G}(V,E)$ (namely, $\mathbf{E}_j(\boldsymbol{s}_j,M_j,\mathcal{G}(V,E))=\boldsymbol{s'}_j$) and the decoder

$$\mathbf{D}_j:\{0,1\}^n\times\mathcal{G}(V,E)\rightarrow\{0,1\}^{\log_2 L}$$

reads the message M_j from the current cell state at any time between the j-th and the $(j+1)$-th updates. (Namely, $\mathbf{D}_j(\boldsymbol{s'}_j,\mathcal{G}(V,E))=M_j$.)

The Ct updates are performed using a differential scheme: the first message M_0 is stored in r_0. To write message M_1, we compute the label $\Delta_1\in\{0,1\}^{\log_2 D}$ of the edge $\pi(M_0)\rightarrow\pi(M_1)$ in $\mathcal{G}(V,E)$, and store in r_1. (Instead of labeling edges globally, each outgoing edge of a vertex is given a local label that costs $\log_2 D$ bits, where D is the *maximum out degree*.) The next $C-2$ updates can be written in the same way. After r_{C-1} is used, an update cycle is completed, and the register r_0 will be rewritten with the new $\log L$-bit message for the next update. The iteration continues until the last update is finished. The construction implies the constraint that for all j, and for all i such that the i-th cell belongs to $r_{j \bmod C}$, we have $s'_{i,j}\geqslant s_{i,j}$. The code rate of trajectory codes is

$$R\triangleq\frac{Ct\log_2 L}{n}\text{ bits/cell.}\tag{1}$$

III. Error Correcting Trajectory Codes

We study the coding problem for joint partial rewriting and error correction, where the partial rewriting model is extended by allowing cell states to be changed by noise between two adjacent updates. In flash memories, the noise is from various sources such as interference and charge leakage [2].

A. Error Model and WOM Parameters

Before we present the code construction, we first introduce the related model and parameters. Let the noise channel for the errors received by a register between two adjacent updates (e.g. the time period after storing M_3 and before storing M_4) be a binary symmetric channel $\text{BSC}(p)$ with $p\in(0,\frac{1}{2})$. We assume that errors start occurring in a register after the register is written for the first time. The assumption is motivated by practical flash memories, where the major errors for rewriting are introduced by cell-to-cell interference that happens mainly when cells are being programmed [2]. Following the model of trajectory codes, the noise channel that a register goes through at the time immediately before its next WOM rewrite is $\text{BSC}(p^{*C})$, where p^{*C} is the overall error probability of C cascaded $\text{BSC}(p)$ computed using $p^{*i}\triangleq\frac{1-(1-2p)^i}{2}$.

In WOM, it is common to use some parameters to control the amount of information that is written in each write. For $j\in[t]$, let the parameter $\alpha_{i,j-1}$ be the fraction of cells that are at state 0 immediately before the j-th write of the register r_i's WOM code. We have $\alpha_{i,0}=1$. Let the parameter $\epsilon_{i,j}\leqslant\frac{1}{2}$ be the fraction of cells at state 0 that will be raised to 1 by the j-th rewrite. We have $\epsilon_{i,t}=\frac{1}{2}$. The parameters of the WOM codes used in our setting of partial rewriting also depends on the error probability p. When $n_i\rightarrow\infty$, the values of $\alpha_{i,1}$, $\alpha_{i,2}$, $\cdots$, $\alpha_{i,t-1}$ are computed by $\alpha_{i,j}=[\alpha_{i,j-1}(1-\epsilon_{i,j})]*p^{*C}$, where $a*b\triangleq a(1-b)+(1-a)b$, and the parameters $\epsilon_{i,1},\epsilon_{i,2},\cdots,\epsilon_{i,t}$ are specified by users.

B. Code Construction

Our first construction is a natural extension of trajectory codes, where each register independently corrects the errors in it. The recovered messages are used by the next update. We formally present the construction by defining the encoder and the decoder in the following.

Construction 2. *For $i\in\{0,1,\cdots,C-1\}$, let register r_i use a t-write EC-WOM code, correcting the errors from $\text{BSC}(p^{*(C-i)})$. Let $l=j \bmod C$, and let $\boldsymbol{s}_j$ be the states of the n cells right before the j-th update, and let $\boldsymbol{s'}_j$ denote the cell states at any time between the j-th and the $(j+1)$-th update. (Therefore, $\boldsymbol{s}_j$ is the value of $\boldsymbol{s'}_{j-1}$ at a particular moment.) For $j=0,1,\cdots,Ct-1$, we have*

Encoder $\mathbf{E}_j(\boldsymbol{s}_j,M_j,\mathcal{G}(V,E))=\boldsymbol{s'}_j$
If $l=0$, rewrite r_0 with M_j. Otherwise, do;

(1) Recover message $\hat{M}_{j-1}=\mathbf{D}_{j-1}(\boldsymbol{s}_j,\mathcal{G}(V,E))$.
(2) Compute the label Δ_j s.t. $(\pi(\hat{M}_{j-1})\xrightarrow{\Delta_j}\pi(M_j))\in E$. (Here π is specified in Definition 1.)
(3) Store the label Δ_j in register r_l using rewriting (i.e. using the EC-WOM code for r_l).

Decoder $\mathrm{D}_j(\boldsymbol{s'}_j, \mathcal{G}(V,E)) = \hat{M}_j$

(1) Decode r_0 and obtain the estimated message $\hat{M}_{j-l}$. Let $\hat{v}_{j-l} = \pi(\hat{M}_{j-l})$.

(2) For k from 1 to l, decode r_k and obtain the estimated edge label $\hat{\Delta}_{j-l+k}$. (Note that in the rewriting graph $\mathcal{G}(V,E)$, the edge from message $M_{j-l+k-1}$ to message $M_{j-l+k-1}$ has the label Δ_{j-l+k}).

(3) Compute $\hat{M}_j$. Start from the vertex $\hat{v}_{j-l}$, traverse $\mathcal{G}(V,E)$ along the path marked by $\hat{\Delta}_{j-l+1}, \hat{\Delta}_{j-l+2}, \cdots, \hat{\Delta}_j$, which leads to $\hat{v}_j$. Output $\hat{M}_j = \pi^{-1}(\hat{v}_j)$.

Example 3. *We now show a simple example for $n = 6$ cells. (In practice, the code usually has thousands of cells.) Let the cells be divided into $C = 2$ registers with $n_0 = 4$, $n_1 = 2$, and let $t = 2$, $L = 4$ and $D = 2$. Assume that between two adjacent updates, an error occurs in each register. Let the WOM codes of r_0 and r_1 correct 2 and 1 errors, respectively. Let the rewriting graph $\mathcal{G}$ whose vertex and edge sets be defined as $V = \{v_0, v_1, v_2, v_3\}$, $E = \left\{ v_0 \underset{(0)}{\overset{(0)}{\rightleftarrows}} v_1, v_1 \underset{(0)}{\overset{(1)}{\rightleftarrows}} v_2, v_2 \underset{(0)}{\overset{(1)}{\rightleftarrows}} v_3, v_3 \underset{(1)}{\overset{(1)}{\rightleftarrows}} v_0 \right\}$, where the vertex v_i representing the symbol i and having two outgoing edges locally labeled with (0) and (1). Let the sequence of messages be $(0,3,2,1)$, which corresponds to the path $v_0 \overset{(1)}{\to} v_3 \overset{(0)}{\to} v_2 \overset{(0)}{\to} v_1$ in the graph. Assume that the changes on the states of r_0 and r_1 during the updates are the ones shown in the table below. Here j^- and j^+ denote the moments immediately before and after the j-th update, respectively. A bit marked with underlines indicates an error. Note that at the moment $j = 2$, although performing the update does not require recovery of the messages written at moments $j = 0$ and $j = 1$, those messages can still be recovered until the moment $j = 2^-$ if needed.*

j	r_0	r_1	**Comments**
0^-	$(0,0,0,0)$	$(0,0)$	*Initialization*
0^+	$(0,1,0,0)$	$(0,0)$	*Wrote data "0" in r_0*
1^-	$(0,1,\underline{1},0)$	$(0,0)$	*An error occurs in r_0*
1^+	$(0,1,\underline{1},0)$	$(1,0)$	*Decoded r_0, wrote "(1)" in r_1*
2^-	$(0,\underline{0},\underline{1},0)$	$(\underline{0},0)$	*Errors occur in r_0 and r_1*
2^+	$(1,0,1,0)$	$(\underline{0},0)$	*Rewrote r_0 to store "2"*
3^-	$(1,0,\underline{0},0)$	$(\underline{0},\underline{1})$	*Errors occur in r_0 and r_1.*
3^+	$(1,0,\underline{0},0)$	$(0,1)$	*Decoded r_0, wrote "(0)" in r_1*

C. Analysis of the Correctness of the Construction

To see the correctness of the coding scheme, we use induction. (Here we assume the number of cells goes to infinity.) Let us assume that the first j messages have been stored successfully, and we show that M_{j-1} can be recovered reliably at any time between the $(j-1)$-th and the j-th update, and the j-th message can be stored successfully. Let the index of the register be written as $l = j \bmod C$. If $l = 0$, we are at the first write of a new cycle, and do not need to recover M_{j-1} to store M_j; if $l > 0$, we perform the update by storing the difference Δ_j between M_{j-1} and M_j in r_l. To do so, we first recover the value of M_{j-1} by decoding the registers r_0, $r_1, \cdots, r_{l-1}$ which have respectively received errors from the channels $\mathrm{BSC}(p^{*l})$, $\mathrm{BSC}(p^{*(l-1)}), \cdots, \mathrm{BSC}(p)$ at the time of the decoding. As their WOM codes respectively correct errors from $\mathrm{BSC}(p^{*C})$, $\mathrm{BSC}(p^{*(C-1)}), \cdots, \mathrm{BSC}(p^{*(C-l+1)})$ which are degraded versions of their current noise channels, these registers can be decoded, outputting the messages written by the last l updates (which include $\hat{M}_{j-l}$ stored in r_0, and the labels $\hat{\Delta}_{j-l+1}, \hat{\Delta}_{j-l+2}, \cdots, \hat{\Delta}_{j-1}$ from $r_1, \cdots, r_{l-1}$). Given $\mathcal{G}(V,E)$ we can determine the value of $\hat{M}_{j-1}$, and further compute the label Δ_j of the edge from $\pi(M_{j-1})$ to $\pi(M_j)$. By storing the label Δ_j into r_l, the j-th update succeeds.

D. Code Analysis

We analyze the code performance for Construction 2. Let $L = |V|$. For $j \in [t]$, let $R_{i,j} > 0$ be the achievable instantaneous rate of the j-th write of the EC-WOM code in r_i. As each register uses a constant-rate WOM code (here register r_0 stores $\log_2 L$ bits per write, and the other registers each stores $\log_2 D$ bits per write), for $i \in [C-1]$ we have

$$n_0 = \frac{\log_2 L}{\min_{j\in[t]} R_{0,j}},\ n_i = \frac{\log_2 D}{\min_{j\in[t]} R_{i,j}}. \tag{2}$$

Substituting Eq. (2) in Eq. (1) gives the rate of the code

$$R = \frac{t \cdot C}{\frac{1}{\min_{j\in[t]} R_{0,j}} + \frac{\log_2 D}{\log_2 L}\sum_{i=1}^{C-1} \frac{1}{\min_{j\in[t]} R_{i,j}}}.$$

Note that the EC-WOM in Construction 2 is general. To be specific, we can use the polar EC-WOM code in [6] for each register, and derived the bounds on R. We first revisit some results from [6] that are needed to derive the bounds to the instantaneous rates for the polar EC-WOM code.

Let the WOM channel used for performing the j-th write/encoding of the polar EC-WOM be $\mathrm{WOM}(\alpha_{j-1}, \epsilon_j)$ with the parameters α_{j-1} and ϵ_j, and let the channel of noise in cell states between two adjacent writes be $\mathrm{BSC}(p_e)$. Let $F_{\mathrm{WOM}(\alpha_{j-1},\epsilon_j)} \subseteq [N]$ be the frozen set of the polar code constructed for $\mathrm{WOM}(\alpha_{j-1}, \epsilon_j)$, and let $F_{\mathrm{BSC}(p_e)} \subseteq [N]$ be the frozen set of the code constructed for $\mathrm{BSC}(p_e)$. When $N \to \infty$, let $x_j \triangleq |F_{\mathrm{WOM}(\alpha_{j-1},\epsilon_j)} \cap F_{\mathrm{BSC}(p_e)}|/|F_{\mathrm{BSC}(p_e)}| \leqslant 1$. For $j \in [t]$, the number of bits written in the j-th rewrite is $\mathcal{M}_j = |F_{\mathrm{WOM}(\alpha_{j-1},\epsilon_j)}| - |F_{\mathrm{WOM}(\alpha_{j-1},\epsilon_j)} \cap F_{\mathrm{BSC}(p_e)}| = N\alpha_{j-1}\mathrm{H}(\epsilon_j) - x_j|F_{\mathrm{BSC}(p_e)}| = N(\alpha_{j-1}\mathrm{H}(\epsilon_j) - x_j\mathrm{H}(p_e))$ and the number of additional cells we use to store the bits in $F_{\mathrm{BSC}(p)} - F_{\mathrm{WOM}(\alpha_{j-1},\epsilon_j)}$ is $N_{\mathrm{additional},j} = \frac{N\,\mathrm{H}(p_e)(1-x_j)}{1-\mathrm{H}(p_e)}$. Therefore, we get the instantaneous rate for the j-th write

$$R_j \triangleq \frac{\mathcal{M}_j}{N + \sum_{k=1}^{t} N_{\mathrm{additional},j}} = \frac{\alpha_{j-1}\mathrm{H}(\epsilon_j) - \mathrm{H}(p_e)x_j}{1 + \frac{\mathrm{H}(p_e)}{1-\mathrm{H}(p_e)}\sum_{k=1}^{t}(1-x_k)}.$$

Lemma 4. *[6, Lemma 5] Let $0 < p_e \leqslant \alpha_{j-1}\epsilon_j$. Then $x_j \geqslant \gamma_j$, where*

$$\gamma_j \triangleq \max\left\{ \frac{\alpha_{j-1}\mathrm{H}(\frac{p_e}{\alpha_{j-1}})}{\mathrm{H}(p_e)}, \frac{\alpha_{j-1}\mathrm{H}(\epsilon_j) + \mathrm{H}(p_e) - \mathrm{H}(\alpha_{j-1}\epsilon_j)}{\mathrm{H}(p_e)} \right\}.$$

Lemma 5. *Let $0 < p_e \leqslant \alpha_{j-1}\epsilon_j$. Then $R_j \in [R_j^-, R_j^+]$, where*

$$R_j^- = \frac{\alpha_{j-1}\mathrm{H}(\epsilon_j) - \mathrm{H}(p_e)}{1 + \frac{\mathrm{H}(p_e)}{1-\mathrm{H}(p_e)}\sum_{k=1}^{t}(1-\gamma_k)}, \tag{3}$$

$$R_j^+ = \alpha_{j-1}\mathrm{H}(\epsilon_j) - \mathrm{H}(p_e)\gamma_j. \tag{4}$$

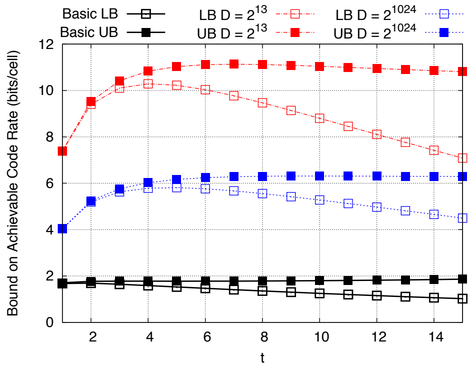

Fig. 1. The lower and upper bounds (marked by LB and UB) on the achievable code rates for different t and D. Here $\log_2 L = 2^{13}$, $C = 8$ and $p = 10^{-3}$.

The results above can be directly applied to the codes in Construction 2. For $i \in \{0, 1, \cdots, C-1\}$, let $0 < p^{*(C-i)} \leqslant \alpha_{i,j-1}\epsilon_{i,j}$, then $R_{i,j} \in [R^-_{i,j}, R^+_{i,j}]$ where $R^-_{i,j}$ and $R^+_{i,j}$ are computed with the right hand sides of Eq. (3) and (4) by replacing α_{j-1}, ϵ_j and p_e with $\alpha_{i,j-1}$, ϵ_j and $p^{*(C-i)}$.

Theorem 6. *For $i \in \{0, 1, \cdots, C-1\}$, $j \in [t]$, let $0 < p^{*(C-i)} \leqslant \alpha_{i,j-1}\epsilon_{i,j}$. Then $R \in \{R^-, R^+\}$ where*

$$R^- = \frac{t \cdot C}{\frac{1}{\min_{j \in [t]} R^-_{0,j}} + \frac{\log_2 D}{\log_2 L} \sum_{i=1}^{C-1} \frac{1}{\min_{j \in [t]} R^-_{i,j}}}.$$

and the upper bound R^+ can be computed by replacing $R^-_{0,j}$ and $R^-_{i,j}$ in the above equation with $R^+_{0,j}$ and $R^+_{i,j}$, respectively.

Figure 1 shows some numerical results for the bounds of our code, where for all i, j we let $\epsilon_{i,j} = 1/(2 + t - j)$. To show the benefit obtained by taking advantage of the partial rewriting constraints, we compare the bounds of our scheme to those of the basic scheme, which is simply a Ct-write polar EC-WOM code correcting errors from $\mathrm{BSC}(p)$. In each rewrite, the basic scheme stores each updated message using rewriting. The results suggest our code performs significantly better than the basic scheme (Note that the WOM codes considered in this paper are constant rate codes. Given such codes, the bounds in Figure 1 decreases when t becomes sufficiently large due to the drop in the instantaneous rates.)

IV. A More Generalized Coding Scheme

We now discuss a more generalized coding scheme. In this scheme, the trajectory codes not only use registers to store the changes in the messages, but can also store part of the errors found in previous registers. When the error probability of the channel is small, only a small number of additional cells are needed to store such error information. We focus on a specific construction in the following.

A. Code Construction

Let the error-free cell states of register r_i (immediately after it is written) be $\boldsymbol{c}_i^0 \in \{0, 1\}^{n_i}$. Let the cell states immediately before each of the next C updates of messages be $\boldsymbol{c}_i^1, \boldsymbol{c}_i^2, \cdots, \boldsymbol{c}_i^C$. According to the error model in Section III, the error vector $\boldsymbol{c}_i^k \oplus \boldsymbol{c}_i^{k+1}$ contains the errors introduced by $\mathrm{BSC}(p)$. When $n_i \to \infty$, the vector $\boldsymbol{c}_i^k \oplus \boldsymbol{c}_i^{k+1}$ can be compressed into $n_i \mathrm{H}(p)$ bits using lossless source coding. The encoder and the decoder for the j-th update in the new construction are defined below.

Construction 7. *For $i \in \{0, 1, \cdots, C-1\}$, let register r_i use a t-write EC-WOM code, correcting the errors from $\mathrm{BSC}(p)$. For $j = 0, 1, \cdots, Ct-1$, we have*

Encoder $\mathbf{E}_j(\boldsymbol{s}_j, M_j, \mathcal{G}(V, E)) = \boldsymbol{s}'_j$
If $l = 0$, rewrite r_0 with M_j. Otherwise, do;
(1) Recover message $\hat{M}_{j-1} = \mathbf{D}_{j-1}(\boldsymbol{s}_j, \mathcal{G}(V, E))$.
(2) Compute the label Δ_j s.t. $(\pi(\hat{M}_{j-1}) \xrightarrow{\Delta_j} \pi(M_j)) \in E$.
(3) Rewrite register r_j to store Δ_j and the compressed version of the error vectors $\boldsymbol{c}^0_{l-1} \oplus \boldsymbol{c}^1_{l-1}, \boldsymbol{c}^1_{l-2} \oplus \boldsymbol{c}^2_{l-2}, \cdots, \boldsymbol{c}^{l-1}_0 \oplus \boldsymbol{c}^l_0$.

Decoder $\mathbf{D}_j(\boldsymbol{s}'_j, \mathcal{G}(V, E)) = \hat{M}_j$
(1) For k from 0 to l, let the state of register r_{l-k} be $\boldsymbol{c}^{k+1}_{l-k}$. Using it and the error vectors obtained previously from decoding $r_{l-k+1}, r_{l-k+2}, \cdots, r_l$, we get $\boldsymbol{c}^{k+1}_{l-k} \oplus \sum_{x=0}^{k-1} (\boldsymbol{c}^x_{l-k} \oplus \boldsymbol{c}^{x+1}_{l-k}) = \boldsymbol{c}^0_{l-k} \oplus (\boldsymbol{c}^k_{l-k} \oplus \boldsymbol{c}^{k+1}_{l-k})$. (Note that when $k = 0$, the above equals $\boldsymbol{c}^1_l$.) Decode the right hand side of the above equation, and obtain the recorded error vectors about the first $(l-k)$ registers—$\boldsymbol{c}^{l-k}_0 \oplus \boldsymbol{c}^{l+1-k}_0, \boldsymbol{c}^{l-k-1}_1 \oplus \boldsymbol{c}^{l-k}_1, \cdots, \boldsymbol{c}^0_{l-k} \oplus \boldsymbol{c}^1_{l-k}$—the estimated message $\hat{\mathcal{M}}_{j-l}$ (when $k = l$) or the estimated edge label $\hat{\Delta}_{j-k}$ (when $k < l$).
(2) We now compute $\hat{M}_j$; we traverse the graph $\mathcal{G}(V, E)$ along the path marked by the labels $\hat{\Delta}_{j-l+1}, \hat{\Delta}_{j-l+2}, \cdots, \hat{\Delta}_j$, which leads to vertex $\hat{v}_j$. Output $\hat{M}_j = \pi^{-1}(\hat{v}_j)$.

Example 8. *Let $n = 10$, $t \geqslant 1$, $C = 3$, $L = 4$ and $D = 2$. Assume $n_0 = 3$, $n_1 = 3$, $n_2 = 4$, and that the WOM code of each register corrects 1 error. Assume that between two adjacent updates, an error occurs in each register. We assume the same rewriting graph as in Example 3, and let $(0, 3, 2)$ be the first three messages to be stored. We only illustrate the update for the message "2" due to space limitation. Assume the changes in the cell states during the updates are as in the table below. At time 2^-, errors occur in r_0 and r_1. To perform the update, we first decode r_1, and obtain the label "(1)" and the decompressed error vector $\boldsymbol{c}^0_0 \oplus \boldsymbol{c}^1_0 = (0, 0, 1)$ for r_0. Given the error vector and the current state $\boldsymbol{c}^2_0$, compute $\boldsymbol{c}^2_0 \oplus (\boldsymbol{c}^0_0 \oplus \boldsymbol{c}^1_0) = (0, 0, 0)$ where the middle bit still contains error. Decoding $\boldsymbol{c}^2_0 \oplus (\boldsymbol{c}^0_0 \oplus \boldsymbol{c}^1_0)$ gives the message "0". Given the new message "2" and the recovered label "(1)" and the message "0" in r_0, the label "(0)" is determined and stored by writing "(0)" in r_2, which completes the update.*

j	r_0	r_1	r_2
0^-	$(0, 0, 0)$	$(0, 0, 0)$	$(0, 0, 0, 0)$
0^+	$(0, 1, 0)$	$(0, 0, 0)$	$(0, 0, 0, 0)$
1^-	$(0, 1, \underline{1})$	$(0, 0, 0)$	$(0, 0, 0, 0)$
1^+	$(0, 1, \underline{1})$	$(1, 0, 0)$	$(0, 0, 0, 0)$
2^-	$(0, \underline{0}, \underline{1})$	$(1, \underline{1}, 0)$	$(0, 0, 0, 0)$
2^+	$(0, \underline{0}, \underline{1})$	$(1, \underline{1}, 0)$	$(1, 0, 1, 0)$

B. Analysis of the Correctness of the Code

The correctness of Construction 7 can be shown by induction. (We again assume the number of cells in each register goes to infinity.) Assume the first j messages have been stored successfully, and we elaborate on the j-th update with $l > 0$. To perform the update, we need to recover the message M_{j-1} so that the label of the edge from M_{j-1} to M_j can be computed and stored in r_l. We first decode r_{l-1} with state $\boldsymbol{c}_{l-1}^1$ which received the errors from $\mathrm{BSC}(p)$. Since each register tolerates errors from $\mathrm{BSC}(p)$, r_{l-1} can be decoded to obtain the edge label Δ_{j-1} (that specifies the edge connecting M_{j-2} to M_{j-1}) as well as the error vectors $\boldsymbol{c}_{l-2}^0 \oplus \boldsymbol{c}_{l-2}^1$, $\boldsymbol{c}_{l-3}^1 \oplus \boldsymbol{c}_{l-3}^2$, $\cdots$, $\boldsymbol{c}_0^{l-2} \oplus \boldsymbol{c}_0^{l-1}$ with each error vector being for one of the first $l-1$ registers. Next, we decode r_{l-2} with state $\boldsymbol{c}_{l-2}^2$. To do so, we first use the error vector obtained previously on r_{l-2} to correct part of the errors by computing $\boldsymbol{c}_{l-2}^2 \oplus (\boldsymbol{c}_{l-2}^0 \oplus \boldsymbol{c}_{l-2}^1)$. The remaining errors can be equivalently seen as coming from $\mathrm{BSC}(p)$, and are thus correctable. Decoding them gives the edge label Δ_{j-2} as well as the error vectors regarding the previous registers. We continue the joint decoding in the same fashion towards r_0. Thanks to the error vectors from the previous decoding, each register needs to correct errors from $\mathrm{BSC}(p)$ (instead of $\mathrm{BSC}(p^{*(C-i)})$ for $i = 0, 1, \cdots, C$–1). After r_0 is decoded, we obtain the message M_{j-l} and the labels $\Delta_{j-l+1}, \Delta_{j-l+2}, \cdots, \Delta_{l-1}$. By traversing $\mathcal{G}(V, E)$ along the path marked by the labels, we recover $\hat{M}_{j-1}$. The label Δ_j is then determined and written into r_l.

C. Code Analysis

We analyze the code performance of Construction 7. The analysis is different from Construction 2 mainly for two reasons. The EC-WOM code of each register for the codes of this section corrects the errors from $\mathrm{BSC}(p)$ while each WOM code tolerates different amount of noise in the previous construction. Moreover, since each register (besides r_0) stores both error vectors as well as an edge label, the value of n_i also depends on $n_0, n_1, \cdots, n_{i-1}$.

We first derive n_i for each r_i. As r_0 is used in the same way as the previous codes, and r_i stores i error vectors and one edge label in each write, we have $n_0 = \log_2 L / \min_{j\in[t]} R_{0,j}$ and $n_i = (\log_2 D + \mathrm{H}(p)\sum_{k=0}^{i-1} n_k)/\min_{j\in[t]} R_{i,j}$ for $i \in [C$–$1]$. Here the term $\mathrm{H}(p)\sum_{k=0}^{i-1} n_k$ is the length of the compressed error vectors $\boldsymbol{c}_{i-1}^0 \oplus \boldsymbol{c}_{i-1}^1$, $\boldsymbol{c}_{i-2}^1 \oplus \boldsymbol{c}_{i-2}^2, \cdots, \boldsymbol{c}_0^{i-1} \oplus \boldsymbol{c}_0^i$. In practice, each register can choose to use the WOM code with the same parameters to simplify the implementation. In such cases, $(n_1, n_2, \cdots, n_{C-1})$ form a geometric sequence.

Proposition 9. *For $i \in \{1, 2, \cdots, C-1\}$, let $\min_{j\in[t]} R_{i,j}$ be some constant A. Then we have $n_i = (n_0 \mathrm{H}(p) + \log_2 D)(A + \mathrm{H}(p))^{i-1}/A^i$.*

Therefore, the rate of the code in this section can be computed using Eq. (1). To derive the bounds for Construction 7, we apply the same techniques used in Section III. Assume each WOM code in is a polar EC-WOM code which corrects errors from $\mathrm{BSC}(p)$. By applying Lemma 5, we show the bounds to the instantaneous rates $R_{i,j}$ in the next lemma.

Lemma 10. *For $i \in \{0, 1, \cdots, C-1\}$ and $j \in [t]$, let $0 < p \leqslant \alpha_{i,j-1}\epsilon_{i,j}$. Then we have $R_{i,j} \in [R_{i,j}^-, R_{i,j}^+]$, where $R_{i,j}^- = [\alpha_{i,j-1}\mathrm{H}(\epsilon_{i,j}) - \mathrm{H}(p)] / \Big[1 + \frac{\mathrm{H}(p)}{1-\mathrm{H}(p)} \sum_{j=1}^t (1 - \gamma_{i,j})\Big]$ and $R_{i,j}^+ = \alpha_{i,j-1}\mathrm{H}(\epsilon_{i,j}) - \mathrm{H}(p)\gamma_{i,j}$.*

Theorem 11. *For all i and j, let $0 < p \leqslant \alpha_{i,j-1}\epsilon_{i,j}$. Then we have $R \in [R^-, R^+]$, where $R^- = Ct\log_2 L / \Big[\frac{\log_2 L}{\min_{j\in[t]} R_{0,j}^-} + \sum_{i\in[C-1]} \frac{\log_2 D + \mathrm{H}(p)\sum_{k=0}^{i-1} n_k}{\min_{j\in[t]} R_{i,j}^-}\Big]$, and R^+ can be computed by replacing $R^-(0, j)$ and $R^-(i, j)$ in R^- above with $R^+(0, j)$ and $R^+(i, j)$, respectively.*

Figure 2 shows the numerical results that compare the bounds of Construction 2 and Construction 7 on parameters that are common for flash memories (e.g. message length > 1000 bits). The bounds for the codes in this section are

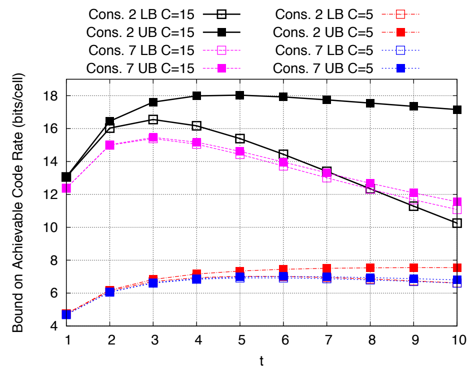

Fig. 2. The bounds to the achievable rates of the two constructions on different t and C. Here $\log_2 L = 2^{13}$, $p = 10^{-3}$, and $\epsilon_{i,j} = \frac{1}{2+t-j}$.

tighter than those of the previous construction. When t is sufficiently large, all bounds will decrease due to the decrease of the minimum instantaneous rates. However, the bounds of the codes in this section decrease more slowly. This is because in the first construction, the WOM code of r_i needs to tolerate the errors from $\mathrm{BSC}(p^{*(C-i)})$. Its error rates become much higher than what the codes in this section needs to tolerate (which is $\mathrm{BSC}(p)$) when C becomes large. Therefore, the minimum instantaneous rates of the WOM codes in the previous scheme decrease faster when t increases than those of the codes in this section do.

References

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

[2] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2012, pp. 521–526.

[3] E. En Gad, Y. Li, J. Kliewer, M. Langberg, A. Jiang, and J. Bruck, "Polar coding for noisy write-once memories," in *Proc. ISIT*, 2014.

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

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

[6] A. Jiang, Y. Li, E. En Gad, M. Langberg, and J. Bruck, "Joint rewriting and error correction in write-once memories," in *Proc. ISIT*, 2013, pp. 1067–1071.

[7] S. Korada and R. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. Inf. Theor.*, vol. 56, no. 4, pp. 1751–1768, 2010.

[8] A. Shpilka, "Capacity achieving multiwrite wom codes," *CoRR*, vol. abs/1209.1128, 2012.

[9] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. IT*, vol. 58, no. 9, pp. 5985–5999, 2012.

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

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