High Sum-Rate Three-Write and Nonbinary WOM Codes

Eitan Yaakobi, *Member, IEEE*, and Amir Shpilka

***Abstract*— Write-once memory (WOM) is a storage medium with memory elements, called cells, which can take on q levels. Each cell is initially in level 0 and can only increase its level. A t-write WOM code is a coding scheme, which allows one to store t messages to the WOM such that on consecutive writes every cell's level does not decrease. The sum-rate of the WOM code, which is the ratio between the total amount of information written in the t writes and number of memory cells, is bounded by $\log(t+1)$. Our main contribution in this paper is a construction of binary three-write WOM codes with sum-rate approaching 1.885 for sufficiently large number of cells, whereas the upper bound is 2. This improves upon a recent construction of sum-rate 1.809. A key ingredient in our construction is a recent capacity achieving construction of two-write WOM codes, which uses the so-called Wozencraft ensemble of linear codes. In our construction, we encode information in the first and second write in a way that leaves a large number (roughly half) of the cells nonprogrammed. This allows us to use the above two-write construction in order to invoke a third write to the memory. We also give specific constructions of nonbinary two-write WOM codes and multiple writes, which give better sum-rate than the currently best known ones. In the construction of these codes, we build upon previous nonbinary constructions and show how tools such symbols relabeling can help in achieving high sum-rates.**

***Index Terms*— Coding theory, write-once memories, flash memories, WOM-codes.**

I. Introduction

WRITE-ONCE memories (WOM) were first introduced in 1982 by Rivest and Shamir [22]. Their motivation came from storage medium like punch cards and optical disks. These media are comprised of storage elements, which we call *cells*. The most distinguishable property of these cells is their asymmetric programming attribute. In the binary version, it is only allowed to irreversibly program each cell from value zero to value one. If a cell can accommodate more than two levels, then on each programming operation, it is only possible to increase the cell's level. A WOM code is a coding scheme which allows one to store information and reuse the WOM more than once, while preserving the property that cells are only allowed to increase their levels on each write.

TABLE I
A WOM Code Example

Data bits	First write	Second write (if data changes)
00	000	111
10	100	011
01	010	101
11	001	110

The most famous example of a WOM code is the one given by Rivest and Shamir for storage of two bits twice using only three cells [22] (Table I). In their work, they also analyzed the bounds on the amount of information possible to store in a WOM and presented more codes. Since then, more constructions were given in 1980's and 1990's, see [6], [12], [21] as well as capacity analysis, see [9], [14], [25].

A renewed interest in WOM codes came along in the past five years as a result of the tremendous research work on coding for flash memories. Flash memory is another example of a WOM where its cells are charged with electrons and thus represent multiple levels [4]. Increasing a cell level is fast and easy; however, in order to decrease its level, its entire containing block of cells has to be erased. This does not only affect the writing speed of the flash memory but also significantly reduces its lifetime [4].

Assume t messages are stored to the memory, consisting of n cells. On the i-th write, $1 \leqslant i \leqslant t$, the message size is M_i. The *rate* on the i-th write is defined to be $\mathcal{R}_i = \frac{\log M_i}{n}$, and the *sum-rate* is $\mathcal{R}_{\text{sum}} = \sum_{i=1}^{t} \mathcal{R}_i$. The capacity region of a t-write WOM is the set of all achievable rate tuples. For the binary case, the capacity region was found in [9], [14], and [22]. It was also proved that the maximum achievable sum-rate for a binary WOM code with t writes is $\log(t+1)$ (all logarithms in this paper are taken base 2). These results were generalized in [9] for non-binary WOM and the maximum sum-rate for WOM with cells of q levels was shown to be $\log \binom{t+q-1}{t}$.

The main goal in designing a WOM code is to achieve high sum-rate, while decreasing its encoding and decoding complexities. In [28], motivated by the constructions in [6] and [26], a capacity achieving two-write WOM code construction was presented, and in [17], state of the art results for multiwrite codes were given. Recently, in [24], yet another two-write capacity achieving construction was given.

Manuscript received October 27, 2012; revised February 26, 2014; accepted June 24, 2014. Date of publication August 26, 2014; date of current version October 16, 2014. E. Yaakobi was supported in part by the ISEF Foundation, New York, NY, USA, and in part by the Lester Deutsch Fellowship. A. Shpilka was supported by the Israel Science Foundation under Grant 339/10. This paper was presented at the 2012 IEEE International Symposium on Information Theory [30].

E. Yaakobi is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: yaakobi@caltech.edu).

A. Shpilka was with the Centre Interfacultaire Bernoulli, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland. He is now with the Department of Computer Science, Technion-Israel Institute of Technology, Haifa 32000, Israel (e-mail: shpilka@cs.technion.ac.il).

Communicated by V. Guruswami, Associate Editor for Complexity and Cryptography.

Digital Object Identifier 10.1109/TIT.2014.2352213

This construction is more attractive than the one in [28] in the sense that it does not require a block-length as large as required in [28] in order to achieve rates close to the capacity, and furthermore, it has better encoding and decoding complexities.

For three-write WOM codes, the best sum-rate that can be found according to [28] was 1.66 while the upper bound is 2. The work in [24] showed another scheme which improved the sum-rate to be 1.809. Recently two constructions of capacity achieving WOM codes were presented [3], [23]. Burshtein and Sturagslki gave in [3] a construction of WOM codes that are based on polar codes [1]. While the last construction is attractive from its encoding and decoding complexities point of view, it doesn't guarantee success in the worst case, that is, for any sequence of writes. The construction [23] overcomes this constraint and works for the worst case. However, it suffers a large block length and hence is less attractive from the practical point of view.

The first contribution we give in the paper is a construction of three-write WOM codes. We follow up on a two-write capacity achieving construction from [24]. The main idea of this construction is to use a collection of binary codes that are "MDS on the average". Then, if on the first write a fraction of at most p cells in an n-cell block are programmed, then on the second write, it is possible to asymptotically program $(1-p)n$ more bits using one of the matrices in this collection. However, for that purpose one needs to encode also the index of the matrix that is used on the second write. In order to overcome this drawback, it was shown that if the same strategy is invoked for a relatively large number of blocks then one matrix will suffice to guarantee a successful encoding on the second write. In order to use this construction for a three-write WOM code, we simply treat the third write as a second write of a two-write WOM code from [24]. That is, if after the second write only p out of the cells are programmed, then it is possible to take advantage of these remaining non-programmed cells in order to program roughly $(1-p)n$ more information bits. These ideas enable us to give a construction of three-write WOM code with sum-rate approaching 1.885 for sufficiently large number of cells.

While the binary case received most of the attention in the literature, significantly less is known for non-binary WOM codes. This problem was already proposed by Rivest and Shamir in their pioneering work [22]. The information theory limits under constrained sources were analyzed in [8] and the capacity region was studied in [9]. However, first WOM code constructions, based on error-correcting codes, were given only recently by Huang et al. in [15]. These results were then improved in [11]. In [10], Gabrys and Dolecek gave a construction of two-write WOM code for $q=3$ and improved some of the bounds on the sum-rate in case the rates on each write are the same. In [19], Kurkoski gave a first analysis of a potential construction of two-write non-binary WOM code and Bhatia et al. extended this idea in [2] to show a specific WOM code construction for arbitrary number of writes and two cells. Yet another extension for two writes and multiple cells was reported by Kurkoski [18]. Another family of non-binary WOM, based on extending the work of Merkx [12], was recently reported by Haymaker and Kelley in [13].

The second part of our work is dedicated for efficiently constructing first two and then multiple-write non-binary WOM codes. Our approach is similar to the theme in constructing the binary three-write WOM codes. Namely, on each write we use ideas that do not (or slightly) reduce the rate and yet manage to guarantee a large number of cells with a reasonably low level. Our point of departure is a simple two-write construction from [11], which on the first write programs only the low levels in the cells and on the second write only the high levels. We show a first technique to improve upon this construction by relabeling symbols on the first write so there is a relatively large number of cells with low level. Then, during the second write, the cells with low level enable to accommodate the write of another information word. The next step to improve this result is to write biased data on the first write such that a large number of cells are in low level after the first write. Thus, it is possible to write even more information bits on the second write and the sum-rate is improved again. Lastly, we incorporate these ideas with another construction from [11] to get our best results. Lastly, we use similar and other ideas in order to present several constructions of reasonable block lengths for multiple-write WOM codes.

The rest of the paper is organized as follows. In Section II, we briefly review the definitions of WOM codes and state the results which we will use in our constructions. In Section III, we show our main result of a three-write WOM code construction with sum-rate approaching 1.885. In Section IV, we present constructions for two-write non-binary WOM codes and in Section V, we show constructions for arbitrary number of writes. All our constructions improve upon the currently best known sum-rates. Finally, Section VI concludes the paper.

II. Definitions and Basic Properties

In this work, the cells can have two or more (q) levels. Initially, all cells are in level zero. On each write, it is only possible to increase the levels of each cell. A vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0, \ldots, q-1\}^n$ will be called a *cell-state vector*. For two cell-state vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we say that $\boldsymbol{x} \leqslant \boldsymbol{y}$ if $x_i \leqslant y_i$ for all $1 \leqslant i \leqslant n$. For $j \leqslant i$, we use the notation $[j : i]$ to define the set of integers $\{j, \ldots, i\}$. If x represents a bit value then its complement is $\overline{x} = 1 - x$, and for a binary vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\overline{\boldsymbol{x}} = (\overline{x}_1, \ldots, \overline{x}_n)$. For any map $f : A \to B$, $Im(f)$ is the image of the map f.

Definition: An $[n, t; M_1, \ldots, M_t]_q$ t-write WOM code is a coding scheme comprising of n q-ary cells and is defined by t pairs of encoding and decoding maps $(\mathcal{E}_i, \mathcal{D}_i)$, for $1 \leqslant i \leqslant t$. The encoding map $\mathcal{E}_i$ is defined by

$$\mathcal{E}_i : [1 : M_i] \times Im(\mathcal{E}_{i-1}) \to \{0, \ldots, q-1\}^n,$$

where, by definition, $Im(\mathcal{E}_0) = \{(0, \ldots, 0)\}$, such that $\mathcal{E}_i(m, \boldsymbol{c}) \geqslant \boldsymbol{c}$ for all $(m, \boldsymbol{c}) \in [1 : M_i] \times Im(\mathcal{E}_{i-1})$. Similarly, the decoding map $\mathcal{D}_i$ is defined by

$$\mathcal{D}_i : Im(\mathcal{E}_i) \to [1 : M_i],$$

such that for all $(m, \boldsymbol{c}) \in [1 : M_i] \times Im(\mathcal{E}_{i-1})$, $\mathcal{D}_i(\mathcal{E}_i(m, \boldsymbol{c})) = m$. The rate on the i-th write is defined by $\mathcal{R}_i = \frac{\log M_i}{n}$, and the sum-rate is $\mathcal{R}_{sum} = \sum_{i=1}^{t} \mathcal{R}_i$.

We assume that the write number is known at every write as this does not affect the asymptotic of the achievable rates; for more details, see [28].

In [9] and [14], the capacity region of a binary t-write WOM was found to be

$$C_t = \Big\{(\mathcal{R}_1, \ldots, \mathcal{R}_t) | \mathcal{R}_1 \leqslant h(p_1), \mathcal{R}_2 \leqslant (1-p_1)h(p_2), \ldots, \mathcal{R}_{t-1} \leqslant \Big(\prod_{i=1}^{t-2}(1-p_i)\Big)h(p_{t-1}), \mathcal{R}_t \leqslant \prod_{i=1}^{t-1}(1-p_i), \text{ where } 0 \leqslant p_1, \ldots, p_{t-1} \leqslant 1/2\Big\},$$

and $\log(t+1)$ was proved to be the maximum sum-rate. In [9], it was shown that the maximum sum-rate for q levels is $\log \binom{q+t-1}{t}$. It was also shown that all rate-tuples in the capacity region are achievable. However, the problem of finding efficient code construction still remains a challenge.

The capacity region of a two-write WOM is given by

$$C_2 = \big\{(\mathcal{R}_1, \mathcal{R}_2) | \mathcal{R}_1 \leqslant h(p), \mathcal{R}_2 \leqslant (1-p), 0 \leqslant p \leqslant 1/2\big\}.$$

The best sum-rate reported in the literature is 1.4928 [28] and a code construction which is capacity achieving was given. Recently, the work in [24] shows another capacity achieving construction for which the code length and encoding and decoding complexities are significantly better. The capacity achieving constructions in [28] and in [24] use similar ideas. If the WOM has n cells, then on the first write approximately $h(p)n$ bits are stored in the memory in a way that at most pn cells are programmed. Then, on the second write, it is shown how to store approximately $(1-p)n$ more bits on the remaining $(1-p)n$ cells. However, while the construction in [28] restricts on the first write which of the pn cells can be programmed, in [24] any of the pn cells can be programmed.[1] In order to accomplish this property, the construction from [24] uses a set of "average" MDS codes which are derived from a *Wozencraft ensemble* [16], [20]. This collection of matrices guarantees that if on the first write at most p out of the n cells in a block are programmed, then on the second write there exists a matrix that can be used to encode roughly $(1-p)n$ more information bits. But, since the index of this matrix has to be encoded as well, it was shown that for a large number of blocks the same matrix can be used and thus the degradation in the rate to encode the matrix index is negligible. In order to simply the description of the construction in the paper, we will state the property we use from [24] for an individual block and then the repetition of a large number of blocks will give us the results and rate stated in the paper.

Theorem 1 [24]: For any $0 < p < 1$ and $\epsilon > 0$, there exists a length-n WOM code $\mathcal{C}_p$ with rates $(h(p)-\epsilon, 1-p-\epsilon)$ such that on the first write any vector with weight at most pn can be written.

A code whose existence is guaranteed by Theorem 1 will be denoted in the paper by $\mathcal{C}_p$, where $0 < p < 1$. We furthermore assume that the encoding map of the second write of $\mathcal{C}_p$ is denoted by $\mathcal{E}_p$ and the decoding map of the second write is denoted by $\mathcal{D}_p$. The useful property from Theorem 1 is used in showing the next lemma, which can be used to construct multiple-write WOM codes.

Lemma 2: For any $\epsilon > 0$ and integer n there exists $N = poly(1/\epsilon) \cdot 2^{O(n)}$ such that if there exists a t-write WOM code with n cells and on the first $t-1$ writes at most pn cells are programmed, then there is a t-write WOM code of length N that achieves the same rate on the first $t-1$ writes and on the last write achieves rate $(1-p-\epsilon)$.

Sketch: We concatenate $M = \text{poly}(1/\epsilon) \cdot 2^{O(n)}$ copies of the basic code, and set $N = M \cdot n$. On the first $t-1$ writes we encode using M independent copies of the given coding scheme. Then, the last write is performed as the second write of the construction in [24] and thus it is possible to store $(1-p-\epsilon)N$ more bits. The complexity of encoding and decoding is polynomial (or even quasi-linear) in the block-length N. ■

Remark 1: In the forthcoming constructions, it may be easier to present the memory input $\boldsymbol{w}$ as a string in $M = \{0, \ldots, a-1\}^m$. If $\boldsymbol{w}$ can be any word in M then it represents $m \log a$ bits. However, in many cases we will choose only words $\boldsymbol{w}$ in M such that the number of times each symbol $0 \leqslant \ell \leqslant a-1$ appears in $\boldsymbol{w}$ is fixed to be $p_\ell m$, where m is large enough and $\sum_{\ell=0}^{a-1} p_\ell = 1$. Then, we will assume that the message $\boldsymbol{w}$ represents $m \cdot \mathcal{H}(p_0, \ldots, p_{a-1})$ bits, where $\mathcal{H}$ is the entropy function. Even though this is not exactly accurate as there is a $O(\log m)$ factor that has to be subtracted, we will use this approximation in case m is a linear function of the number of cells n. Note that the degradation in the sum-rate is negligible as the block length goes to infinity. Furthermore, we assume that an efficient mapping between all such words $\boldsymbol{w}$ and the number of bits they represent is given, as this part is out of the scope of the paper. For more details of how to construct such mappings we refer the reader for example to the enumerative encoding scheme by Cover [7].

Lastly, we note here that in many places we drop all notation of floors and ceilings in order to ease the analysis of the codes. However, the loss in the rate due to these roundings is negligible and does not affect the asymptotic sum-rate results.

III. Three-Write WOM Codes

In this section, we present our construction of three-write WOM codes. As we use similar ideas, let us first briefly review the three-write WOM code construction in [24] that achieves sum-rate 1.809.

The construction in [24] uses $n = 3m$ cells partitioned into m three-cell blocks. In every three-cell block, the Rivest Shamir construction, shown in Table I, is used such that on the first two writes, the input to every block is two bits. Alternatively, we can think on the input as a quaternary symbol. In general, if any two quaternary symbols are written to a three-cell block, then in the worst case, all three cells are programmed. However, if, for example, on the first write, the symbol zero is stored, then it is guaranteed that after the second write at most one cell is programmed. Furthermore, if a non-zero symbol is stored on both the first and second write then exactly two out of the three cells are programmed.

[1] This is true in an asymptotic sense as Lemma 2 demonstrates, but for sake of intuition it is best to think about it this way.

The guiding principles in the construction in [24] are to program on the first two writes the least number of cells possible in order to leave a large number of cells for the third write. More specifically, two of the main ideas from [24] are to write a biased quaternary vector and relabel its symbols when necessary such that on the third write a large number of three-cell blocks is left with zero or one cell programmed in them. According to Lemma 2, if after the first two writes, at most pn cells are programmed, then on the third write it is possible to write approximately $(1-p)n$ more bits (at the cost of having block-length which is exponentially longer, yet with efficient (in the new block-length) encoding and decoding algorithms).

The construction we present is similar but we use two-cell blocks instead of three, which we simply call *blocks*. We follow similar guidelines from [24] to decrease the worst case maximum number of programmed cells after the first two writes. We use $n = 2m+1$ cells, partitioned into m blocks and one extra cell. We denote the m blocks by $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m)$, $1 \leqslant i \leqslant m$, $\boldsymbol{c}_i = (c_{i,1}, c_{i,2})$, and the extra cell by c_{2m+1}.

We describe in details the encoding and decoding maps for each of the three writes.

First Write: Write a message $\boldsymbol{w}_1 \in \{0,1,2\}^m$ with probabilities $p_0 = 0.5$, $p_1 = p_2 = 0.25$. Namely, $\boldsymbol{w}_1$ has $m/2$ zeros, $m/4$ ones, and $m/4$ twos. For $1 \leqslant i \leqslant m$, the i-th ternary symbol $w_{1,i}$ is stored in the i-th block $\boldsymbol{c}_i$ according to the map $\boldsymbol{c}_i = \psi(w_{1,i})$, where

$$\psi(0) = (0,0),\ \psi(1) = (0,1),\ \psi(2) = (1,0).$$

The decoding of the message $\boldsymbol{w}_1$ is straightforward from this definition. According to Remark 1 and since the number of cells can be arbitrarily large, we conclude that the number of bits stored on the first write approaches[2]

$$m \cdot \mathcal{H}(0.5, 0.25, 0.25) = 1.5m.$$

Second Write: The second write is performed in two steps. On the first step we store roughly m bits and on the second step approximately $m/4$ more bits. Let $\boldsymbol{w}_2 \in \{0,1\}^m$ be a balanced binary vector (i.e., it has the same number of zeros and ones) and let S be the set of blocks that were not programmed on the first write, i.e., $S = \{i \in [1:m] \mid \boldsymbol{c}_i = (0,0)\}$. Let

$$I_0 = \{i \in [1:m] \mid w_{2,i} = 0\},\ I_1 = \{i \in [1:m] \mid w_{2,i} = 1\}.$$

If $|S \cap I_0| \geqslant \frac{|S|}{2}$ we let $\boldsymbol{w}'_2 = \boldsymbol{w}_2$, and otherwise $\boldsymbol{w}'_2 = \overline{\boldsymbol{w}}_2$ and we program the $(2m+1)$-th cell to be 1. The aim of this choice of the vector $\boldsymbol{w}'_2$ is to guarantee that at least half of the blocks in S store bit-value zero of $\boldsymbol{w}'_2$.

For $1 \leqslant i \leqslant m$, $w'_{2,i}$ is stored in $\boldsymbol{c}_i$ according to its parity, and as follows: $\boldsymbol{c}'_i = \phi(\boldsymbol{c}_i, w'_{2,i})$, where

1) $\phi((0,0),0) = (0,0)$ or $(1,1)$ and $\phi((0,0),1) = (0,1)$ or $(1,0)$. It will be explained in the second step which option to choose in each case.
2) $\phi((0,1),0) = (1,1)$, $\phi((0,1),1) = (0,1)$.
3) $\phi((1,0),0) = (1,1)$, $\phi((1,0),1) = (1,0)$.

[2]This is only true in an asymptotic sense, and in reality, if the block-length is n then we lost $\epsilon \approx \log(n)/n$ in the rate. Our final block length will need to be exponential in the error parameter ϵ.

Now we explain the second step of the second write. Note that if after the first write, the state of the i-th block is $\boldsymbol{c}_i = (0,0)$ then according to the definition of ϕ there are two options to program the i-th block. But, these options depend on the value of $w'_{2,i}$ and it is not necessarily clear to know in advance in how many of these blocks the value of $w'_{2,i}$ is 0 or 1. Therefore, it is not clear how to take advantage and store more information this way (e.g., by treating $(0,0)$ and $(0,1)$ as 0 and $(1,1)$ and $(1,0)$ as 1). However, in spite of this difficulty, we next explain how to modify this idea and store roughly $m/4$ more bits. Note also that if $w'_{2,i} = 1$ then the number of programmed cells in the two options is one, while for $w'_{2,i} = 0$ this is not case. Hence we do not seek to program all of these blocks this way and leave some of the blocks where $w'_{2,i} = 0$ to be programmed with $(0,0)$ so the number of available cells on the third write is still relatively large.

By abuse of notation, we say that the i-th block, $1 \leqslant i \leqslant m$, stores the information $[a_i, b_i] \in \{0,1,2\} \times \{0,1\}$ or is called an $[a_i, b_i]$-block if $a_i = w_{1,i}$ and $b_i = w'_{2,i}$. According to the choice of the vectors $\boldsymbol{w}_1, \boldsymbol{w}'_2$, at least 1/4 of the blocks store the information $[0,0]$ and hence we can say that for some $0 \leqslant x \leqslant 1/4$,

1) $(1/4 + x)m$ blocks store the information $[0,0]$.
2) $(1/4 - x)m$ blocks store the information $[0,1]$.
3) $(1/2 - (1/4 + x))m = (1/4 - x)m$ blocks store the information $[1,0]$ or $[2,0]$.
4) $(1/2 - (1/4 - x))m = (1/4 + x)m$ blocks store the information $[1,1]$ or $[2,1]$.

We shall use $m/4$ among the $m/2$ blocks in S and store one more bit in each one of them, while guaranteeing that the number of available cells for the third write will be at least m. This requires the following steps:

1) We mark $m/4$ out of the $m/2$ blocks in S as follows: xm blocks which store the information $[0,0]$ and their set of indices is denoted by S_0, and $(1/4 - x)m$ blocks which store the information $[0,1]$ and their set of indices is denoted by S_1. The other $3m/4$ blocks which do not belong to $S_0 \cup S_1$ are programmed according to the map ϕ, while we choose to program here a $[0,0]$-block with $(0,0)$, that is we choose the option $\phi((0,0),0) = (0,0)$ for these blocks. Each of the $m/4$ blocks in $S_0 \cup S_1$ has two options to be programmed, while preserving its parity value. A $[0,0]$-block can be programmed as $(0,0)$ or $(1,1)$ and a $[0,1]$-block as $(0,1)$ or $(1,0)$.
2) Next, we generate a length-m binary vectors from the m blocks, a bit from each block. Each of the $3m/4$ blocks that do not belong to $S_0 \cup S_1$ is mapped to a bit, representing a cell value, according to its most-significant bit (MSB)

$$(0,0), (0,1) \to 0,\ (1,1), (1,0) \to 1.$$

The $m/4$ blocks in $S_0 \cup S_1$ correspond to $m/4$ available binary cells in an m-cell binary vector. Then, we use a length-m two-write WOM code, denoted by $\mathcal{C}_{3/4}$ which its existence is guaranteed according to Theorem 1. Thus, we will write $m/4$ more bits according to the

second write of $\mathcal{C}_{3/4}$. (The remaining $3m/4$ bits of the binary two-write WOM code are the MSBs that we just defined.)

3) Finally, we program the $m/4$ blocks in $S_0 \cup S_1$. If a block stores the information $[0, 0]$ then it is programmed to be $(0, 0)$ if its corresponding bit by the WOM code $\mathcal{C}_{3/4}$ is 0 and we program it to be $(1, 1)$ if its corresponding bit by $\mathcal{C}_{3/4}$ is 1. Similarly, a $[0, 1]$-block is programmed to be $(0, 1)$ if its corresponding bit by $\mathcal{C}_{3/4}$ is 0 and it is programmed to be $(1, 0)$ if its corresponding bit by $\mathcal{C}_{3/4}$ is 1.

To summarize this discussion, we give the exact encoding map construction for the second write. We assume that the cell-state vector after the first write is $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m, c_{2m+1})$ and the input to the encoder consists of two vectors: a balanced vector $\boldsymbol{w}_2 \in \{0, 1\}^m$ and a vector $\boldsymbol{s} \in \{0, 1\}^{m/4-\epsilon}$. We use a binary two-write WOM code $\mathcal{C}_{3/4}$ of length m, given by the construction in [24] and stated in Theorem 1, with encoding map $\mathcal{E}_{3/4}$ and decoding map $\mathcal{D}_{3/4}$ for the second write. Yet another property we take from the two-write construction in [24] is the following. Assume k bits are to be stored on the second write and there are more than k cells which were not programmed on the first write. Then, it is possible to specify the encoder with a set S of k cells out of the non-programmed cells such that on the second write, only cells from S are allowed to be programmed. (In fact, this is the model for memories with defects.) Hence, the encoding map $\mathcal{E}_{3/4}(\boldsymbol{c}, \boldsymbol{s}, S) = \boldsymbol{c}'$ receives as an input a binary cell-state vector $\boldsymbol{c}$ of weight at most $3m/4$, a message $\boldsymbol{s}$ of $m/4-\epsilon$ bits, and a set S of $m/4$ cells that can be programmed. It outputs a vector $\boldsymbol{c}' \geqslant \boldsymbol{c}$, such that $\mathcal{D}_{3/4}(\boldsymbol{c}') = \boldsymbol{s}$.

The steps to calculate the output $(\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_m, c'_{2m+1})$ of the encoding map are summarized as follows. The comments in parenthesis are just side properties on the values we calculate.

1) $S = \{i \in [1 : m] \mid \boldsymbol{c}_i = (0, 0)\}$, $(|S| = m/2)$.
2) $I_0 = \{i \in [1 : m] \mid w_{2,i} = 0\}$, $I_1 = \{i \in [1 : m] \mid w_{2,i} = 1\}$, $(|I_0| = |I_1| = m/2)$.
3) If $|S \cap I_0| \geqslant \frac{|S|}{2}$ then $\boldsymbol{w}'_2 = \boldsymbol{w}_2$ and $c'_{2m+1} = 0$; otherwise $\boldsymbol{w}'_2 = \overline{\boldsymbol{w}}_2$ and $c'_{2m+1} = 1$.
4) $S'_0 = \{i \in S \mid w'_{2,i} = 0\}$, $S_1 = \{i \in S \mid w'_{2,i} = 1\}$, $(|S'_0| + |S_1| = m/2, |S_1| \leqslant m/4 \leqslant |S'_0|)$.
5) Choose $S_0 \subseteq S'_0$ such that $|S_0| = m/4 - |S_1|$.
6) For all $i \in [1 : m] \backslash (S_0 \cup S_1)$, $\boldsymbol{c}'_i = \phi(\boldsymbol{c}_i, w'_{2,i})$.
7) Let $\boldsymbol{c} = (c_1, \ldots, c_m)$ be $c_i = c_{i,1}$ for $i \in [1 : m] \backslash (S_0 \cup S_1)$ and otherwise $c_i = 0$.
8) Let $\boldsymbol{c}' = \mathcal{E}_{3/4}(\boldsymbol{c}, \boldsymbol{s}, S_0 \cup S_1)$.
9) For $i \in S_0$, $\boldsymbol{c}'_i = (c'_i, c'_i)$ and for $i \in S_1$, $\boldsymbol{c}'_i = (c'_i, 1 - c'_i)$.

To complete the second write, we show how to decode the messages $\boldsymbol{w}_2$ and $\boldsymbol{s}$. Assume the memory state is $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m, c_{2m+1})$ and we decode the messages $\widehat{\boldsymbol{w}}_2$ and $\widehat{\boldsymbol{s}}$. The message $\widehat{\boldsymbol{s}}$ is decoded according to

$$\widehat{\boldsymbol{s}} = \mathcal{D}_{3/4}(c_{1,1}, c_{2,1}, \ldots, c_{m,1}),$$

and the message $\widehat{\boldsymbol{w}}_2$ is decoded according to

$$\widehat{\boldsymbol{w}}_2 = (c_{1,1} \oplus c_{1,2}, c_{2,1} \oplus c_{2,2}, \ldots, c_{m,1} \oplus c_{m,2}) \oplus c_{2m+1} \cdot \mathbf{1},$$

where $\mathbf{1}$ denotes the all-ones vector, and $\oplus$ denotes the XOR operation between two bits.

Third Write: We first calculate the minimum number of available cells for the third write. With the notation of the second write, note the following:

1) blocks storing the information $[0, 0]$: there are $(1/4 + x)m$ such blocks. In $m/4$ blocks, the two cells are not programmed and in the other xm blocks, the two cells may have been programmed.
2) blocks storing the information $[0, 1]$: there are $(1/4 - x)m$ such blocks and in all of them exactly one cell is programmed.
3) blocks storing the information $[1, 0]$ or $[2, 0]$: there are $(1/4 - x)m$ such blocks and in all of them the two cells are programmed.
4) blocks storing the information $[1, 1]$ or $[2, 1]$: there are $(1/4 + x)m$ such blocks and in all of them exactly one cell is programmed.

Therefore, the number of cells which are not programmed after the first two writes is at least

$$m/4 \cdot 2 + (1/4 - x)m \cdot 1 + (1/4 + x)m \cdot 1 = m,$$

and according to Lemma 2 it is possible to store approximately m more bits.[3] That is, now we choose a length-$2m$ two-write WOM code $\mathcal{C}_{1/2}$ with encoding map $\mathcal{E}_{1/2}$ and decoding map $\mathcal{D}_{1/2}$ for the second write. The information written on this write is a word $\boldsymbol{s}'$ of approximately m bits which will be encoded by $\mathcal{E}_{1/2}(\boldsymbol{c}, \boldsymbol{s}')$ and decoded by $\mathcal{E}_{1/2}(\boldsymbol{c})$, where $\boldsymbol{c}$ is the cell-state vector of the first $2m$ cells during encoding and decoding.

To conclude, the sum-rate of the construction approaches

$$\frac{3m/2 + (m + m/4) + m}{2m + 1} = 1.875 - \frac{1.875}{2m + 1}.$$

As the value of m is sufficiently large, we conclude that the sum-rate approaches 1.875.

A small improvement is achieved in case that the probability p_0 on the first write is not 0.5. Assume that $\boldsymbol{w}_1$ is distributed according to $p_0 = p$, $p_1 = p_2 = (1 - p)/2$. Thus, the rate on the first write approaches $(h(p) + (1 - p))/2$. The construction is similar with the following modification on the number of blocks of each type on the second write:

1) $p/2 + x$ of the blocks store the information $[0, 0]$.
2) $p/2 - x$ of the blocks store the information $[0, 1]$.
3) $1/2 - (p/2 + x) = (1 - p)/2 - x$ of the blocks store the information $[1, 0]$ or $[2, 0]$.
4) $1/2 - (p/2 - x) = (1 - p)/2 + x$ of the blocks store the information $[1, 1]$ or $[2, 1]$.

As before, roughly m bits are stored on the first step of the second write. By marking xm $[0, 0]$-blocks and $(p/2 - x)m$ $[0, 1]$-blocks, approximately $xm + (p/2 - x)m = pm/2$ more bits are stored using another two-write WOM code $\mathcal{C}_{1-p/2}$. Thus, the rate on the second write approaches $(1 + p/2)/2$. The remaining number of cells for the third write is

[3] More accurately, we should move to a code which is a concatenation of our construction with itself many (i.e. $exp(mn)$) times and in the new code we will manage to write on the remaining m bits in each block. For simplicity of exposition we keep the current description and just note that we need block-length that is exponentially long in the error parameter ϵ.

given by:

1) At least $pm/2$ blocks store the information $[0, 0]$ and their two cells are not programmed. They contribute at least pm cells.
2) $(p/2 - x)m$ blocks store the information $[0, 1]$ and exactly one of their cells is programmed. They contribute $(p/2 - x)m$ cells.
3) $((1 - p)/2 + x)m$ blocks store the information $[1, 1]$ or $[2, 1]$ and just one of their cells is programmed. They contribute $((1 - p)/2 + x)m$ cells.

Hence, there are $pm + (p/2 - x)m + ((1 - p)/2 + x)m = (1/2 + p)m$ available cells for the third write. Thus, for large number of cell, the sum-rate approaches

$$\frac{(h(p) + 1 - p) + (1 + p/2) + (p + 1/2)}{2}.$$

This term is maximized for $p = \frac{\sqrt{2}}{1+\sqrt{2}}$ and its value is 1.885. We summarize this in the following theorem.

Theorem 3: For any $\epsilon > 0$, there exists a three-write WOM code of sum-rate at least $1.885 - \epsilon$.

To conclude our discussion, note that the construction in [23] gives WOM codes which achieve the capacity, that is, in our case WOM codes with sum-rate approaching 2. As in our construction, the main drawback of [23] is its significant block length, which is in the order of $(t/\epsilon)^{O(t/\epsilon)}$. Nevertheless, our construction has the advantage that it also provides ideas that can be used for constructing codes of reasonable block-lengths.

IV. Two-Write WOM Codes Over Large Alphabets

In this section, we give new constructions of two-write WOM codes over larger alphabets. We shall consider alphabets of size q, namely, $\{0, \ldots, q-1\}$, where on each write it is only possible to increase the level of each cell. The motivation in constructing these codes is inspired by similar ideas from the construction of binary three-write WOM codes in Section III coupled with two constructions from [11].

What we show next is an approach for constructing two-write WOM codes with sum-rate of $\log(q^2/3) + O(\log(q)/q)$. The term $O(\log(q)/q)$ represents our improvement over existing constructions and note that the theoretical upper bound in this case is $\log \binom{q+1}{2}$ [9]. This is done in several steps. First, we start with a simple construction. Then, we show improvements for this construction using an idea of [24]. Finally, we embed these ideas together with a construction in [11] in order to achieve the currently best known sum-rates. We then show that our approach besides being theoretically better also yields concrete constructions which beat the current state of the art constructions for alphabets of size 8 and 16 starting at block lengths which are not too large. We note that the recent work [23] gave constructions that are better for very large block-length, but the methods of [23] inherently demand a large block-length and so the codes that we obtain here are more likely to be practical than those obtained in [23].

Let us first start with a simple construction, reported also in [11]. We use it as a building step for our construction.

A Simple Construction: Assume q is even. On the first write we store an arbitrary word $\boldsymbol{w}_1 \in \{0, \ldots, q/2\}^n$ and on the second write a word $\boldsymbol{w}_2 \in \{0, q/2 + 1, \ldots, q - 1\}^n$. For $1 \leqslant i \leqslant n$, if $w_{2,i} \neq 0$ then we write $w_{2,i}$ in the i-th cell, and otherwise ($w_{2,i} = 0$) we change the content of the i-th cell to $q/2$. Since we wrote $q/2$ only when $\boldsymbol{w}_2$ is zero, we will be able to recover $\boldsymbol{w}_2$ easily. The sum-rate of this construction is

$$\log(q/2 + 1) + \log(q/2) = \log(q(q + 2)/4).$$

A. First Idea, Renaming the Symbols

Note that if $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ were chosen uniformly at random, from their respective domains, then we would have $\Theta(1/q)$ coordinates in which it holds that $\boldsymbol{w}_1$ is smaller than, say, $q/2 - q/\log(q)$, and that $\boldsymbol{w}_2$ is zero. A much better sum-rate would be achieved if we could store more information on these memory cells. For example, we could hope to encode there roughly $O(n/q)$ symbols in an alphabet of size $O(q/\log(q))$. If we could do that, then the rate on the second write will be $\log(q/2) + O((1/q)\log(q))$, as we wanted.

There are several problems of course. The first is that $\boldsymbol{w}_1$ does not necessarily have many "small" coordinates, e.g., it could be the constant $q/2$ word. Similarly, it can be the case that $\boldsymbol{w}_2$ has no zeros. An even bigger concern is how to make sure that there are indeed $O(1/q)$ coordinates in which $\boldsymbol{w}_1$ is small and $\boldsymbol{w}_2$ is zero.

The idea of [24] for overcoming these issues is to simply rename the symbols so that we will have many coordinates with the required properties. This is done by adding a few more coordinates to the code (when n is large this has a negligible effect on the sum-rate) in which we write the new representations of the symbols. We illustrate these ideas in the following example.

Example 1: In this example, we construct two-write WOM codes for $q = 8$. We let the length of the code be $N = n + 3$.

First Write: We get a word $\boldsymbol{w}_1 \in \{0, 1, 2, 3, 4\}^n$. Let $\alpha \leqslant \beta \in \{0, \ldots, 4\}$ be the two most common values appearing in $\boldsymbol{w}_1$. Define a new word $\boldsymbol{w}'_1$ as follows. Whenever $w_{1,i} = \alpha$ we set $w'_{1,i} = 0$ and when $w_{1,i} = 0$ we write $w'_{1,i} = \alpha$. In the same way we replace the value 1 with β. The rest of the coordinates are unchanged. We now write $\boldsymbol{w}'_1$ to the first n memory cells as is. In order to recover $\boldsymbol{w}_1$ we write α in the $(n + 1)$-th cell and β in the $(n + 2)$-th cell. It is clear how we can recover $\boldsymbol{w}_1$ by reading the memory cells.

Second Write: Let $\boldsymbol{w}_2 \in \{0, 5, 6, 7\}^n$ and $\boldsymbol{w}_3 \in \{1, 2, 3\}^{n/10}$ be the input words to be written. Let I' be the set of "small" coordinates of $\boldsymbol{w}'_1$, i.e., $I' = \{i \mid w'_{1,i} \leqslant 1\}$. Note that $|I'| \geqslant (2/5) \cdot n$. Let $\gamma \in \{0, 5, 6, 7\}$ be the most common symbol appearing in $(w_2)_{I'}$. That is, we only consider the coordinates I' and among them we check which value was the most popular in $\boldsymbol{w}_2$. (We break ties arbitrarily.) Let $I'' = \{i \mid i \in I' \text{ and } w_{2,i} = \gamma\}$, and note that

$$|I''| \geqslant (1/4) \cdot |I'| \geqslant n/10.$$

Let I be the first $n/10$ coordinates of I''. We now write γ on the N-th memory cell and define a new word $\boldsymbol{w}'_2$ as follows. Whenever $\boldsymbol{w}_2$ had zero we change it to γ and whenever it

had γ we change it to zero. We now write the rest of the memory cells as follows.

1) If $w'_{2,i} \neq 0$ then we write its value to the i-th cell.
2) If $w'_{2,i} = 0$ and $i \notin I$ then we write the value 4 in the i-th cell.
3) If $w'_{2,i} = 0$ and $i \in I$ and it is the j-th element in I (according to the order $0 < 1 < 2 < \cdots < n/10$) then we write the value $w_{3,j}$ in the i-th cell (alternatively, we can think of $\boldsymbol{w}_3$ as $\boldsymbol{w}_3 \in \{1,2,3\}^{|I|}$ and write its j-th coordinate). Notice that for $i \in I$, $w'_{1,i} \leqslant 1 \leqslant w_{3,i}$ and so this is a "legal" write.

In order to decode the word $\boldsymbol{w}_2$, one needs to read all cell levels, while treating levels 1, 2, 3, and 4 to be zero and converting the γ symbol. The word $\boldsymbol{w}_3$ is decoded according to the first $n/10$ cells of level less than 4.

Finally, we estimate the sum-rate of the construction. In the first write we wrote an arbitrary word in $\{0,1,2,3,4\}^n$ so we transmitted $\log(5^n) = \log(5) \cdot n$ bits of information. In the second write we transmitted a pair $(w_2, w_3) \in \{0,5,6,7\}^n \times \{1,2,3\}^{n/10}$ so overall we sent in the second write $2n + (\log(3)/10)n$ bits. Thus, the sum-rate, given by

$$\begin{aligned}&(\log 5 + 2 + \log 3/10) \cdot (n/N)\\ &\quad = (\log 5 + 2 + \log 3/10) \cdot \left(1 - \frac{3}{n+3}\right),\end{aligned}$$

approaches ≈ 4.4804 for n large enough. In particular, for $n > 31675$ this already improves upon the current state of the art, given in the work of Gabrys et al. [11], which has sum-rate 4.4784. Note that the upper bound on the sum-rate in this case is $\log \binom{9}{2} = \log 36 \approx 5.1699$.

The construction in Example 1 can be extended for arbitrary q and will be called *Construction 1*. We state the construction and prove its sum-rate result (in a slightly informal manner) in the next theorem.

Theorem 4: Construction 1 achieves sum-rate $2\log(q/2) + O(\log(q)/q)$.

Proof: Let us assume that q is even. The case where q is odd will be very similar. Let $0 < \ell < q/2$ be some fixed integer, which will be determined later. Assume there are $N = n + 1$ cells.

First Write: We get a balanced word $\boldsymbol{w}_1 \in \{0, \ldots, q/2\}^n$ (we can also change the representation of some of the symbols, but for simplicity we choose $\boldsymbol{w}_1$ to be balanced since it does not incur any rate loss). We write the word $\boldsymbol{w}_1$ to the memory such that the i-th cell, $1 \leqslant i \leqslant n$ is programmed to level $(w_1)_i$. The decoding is clear.

Second Write: We get two words:

1) $\boldsymbol{w}_2 \in \{0, q/2+1, \ldots, q-1\}^n$,
2) $\boldsymbol{w}_3 \in \{\ell - 1, \ldots, q/2 - 1\}^{\frac{\ell n/(q/2+1)}{q/2}}$.

Let $I' = \{i \mid (w_1)_i \leqslant \ell - 1\}$, and note that

$$|I'| \geqslant \ell n/(q/2+1).$$

We let $\gamma \in \{0, q/2+1, \ldots, q-1\}$ be the most frequent symbol of $\boldsymbol{w}_2$ in the set I' and we generate a new word $\boldsymbol{w}'_2$, where the symbol γ is flipped with 0. This change is marked in the last cell. Now, we let $I'' = \{i \mid i \in I' \text{ and } (w_2)_i = \gamma\}$, so

$$|I''| \geqslant \frac{|I'|}{q/2} \geqslant \frac{\ell n/(q/2+1)}{q/2},$$

and finally we choose I to be a subset of the first $\frac{\ell n/(q/2+1)}{q/2}$ cells in I''. The words $\boldsymbol{w}'_2$ and $\boldsymbol{w}_3$ are stored in the memory as was demonstrated in Example 1. If $w'_{2,i} \neq 0$ then its value is written to the i-th cell. If $w'_{2,i} = 0$ and $i \notin I$ then the value $q/2$ is written to the i-th cell. Finally, if $w'_{2,i} = 0$ and $i \in I$ and it is the j-th element in I, then the value $w_{3,j}$ is written to the i-th cell.

In order to decode the word $\boldsymbol{w}_2$, the value of its i-th symbol is determined from the i-th cell, c_i. If $c_i > q/2$ then $w_{2,i} = c_i$ and otherwise $w_{2,i} = 0$. The word $\boldsymbol{w}_3$ is determined from the first $\frac{\ell n/(q/2+1)}{q/2}$ cells of value less than $q/2$ and these cells determine the $\frac{\ell n/(q/2+1)}{q/2}$ symbols of $\boldsymbol{w}_3$.

Finally, we get that the sum-rate approaches

$$\begin{aligned}&\log\left(\frac{q}{2}+1\right) + \log\left(\frac{q}{2}\right) + \frac{\ell}{\frac{q}{2}(\frac{q}{2}+1)} \cdot \log\left(\frac{q}{2} - \ell + 1\right)\\ &\quad = \log\left(\frac{q}{2}(\frac{q}{2}+1)\right) + \frac{\ell \log\left(\frac{q}{2} - \ell + 1\right)}{\frac{q}{2}(\frac{q}{2}+1)}.\end{aligned}$$

Now, if we choose $\ell = q/2 - \frac{q}{\lfloor \log q \rfloor}$, we get sum-rate

$$= \log\left(\frac{q}{2}(\frac{q}{2}+1)\right) + \frac{\left(q/2 - \frac{q}{\lfloor \log q \rfloor}\right) \log\left(\frac{q}{\lfloor \log q \rfloor} + 1\right)}{\frac{q}{2}(\frac{q}{2}+1)},$$

which for q large enough is in the order of $2\log(q/2) + O(\log(q)/q)$. ∎

We finally note in this part that it is possible to choose a different alphabet size for the words on the first write or remap more symbols on the second write. These two options along with setting other values for ℓ can give other rate pairs. However, we chose these parameters and remapped only a single symbol in order to maximize the sum-rate of this construction.

B. Second Idea, Writing Balanced Vectors on the First Write

Next, we show how to achieve another improvement to the result of Construction 1. Instead of renaming symbols on the first write, the main idea and modification from the last subsection is to force having a large number of symbols with low levels on the first write. Even though this approach will incur a degradation in the rate of the first write, it will enable to increase the rate of the second write which in total improves the sum-rate. We demonstrate it in the next example, which is built upon Example 1.

Example 2: We improve the construction of the two-write code for $q = 8$ in Example 1 as follows. Let $N = n + 1$. Let $2/5 \leqslant p \leqslant 1$ be some parameter. On the first write, we store only words $\boldsymbol{w}_1 \in \{0,1,2,3,4\}^n$ that have at least pn coordinates that are either zero or one (for simplicity, assume that pn is an integer). First, we do not need to write α and β. This enables us to take $N = n+1$. We repeat the construction in Example 1 as before except for the following changes. The word $\boldsymbol{w}_2 \in \{0,5,6,7\}^n$ remains the same and the word $\boldsymbol{w}_3$

satisfies $\boldsymbol{w}_3 \in \{1,2,3\}^{pn/4}$. The sets I', I'' are defined to be $I' = \{i \mid w_{1,i} \leqslant 1\}$ and $I'' = \{i \mid i \in I' \text{ and } w_{2,i} = \gamma\}$, where $\gamma \in \{0,5,6,7\}$ is the most common symbol appearing in $(w_2)_{I'}$. Thus, we have that $|I''| \geqslant pn/4$ so we can let I be the first $pn/4$ coordinates of I''. This explains the change in the domain of $\boldsymbol{w}_3$. The rest of the construction remains the same.

The sum-rate analysis is as follows. On the first write, we store asymptotically

$$(h(p) + p + (1-p)\log 3)n$$

many bits. On the second write, we stored

$$(2 + (p/4)\log 3)n$$

bits. Thus, overall the asymptotic sum-rate is

$$(h(p) + p + (1-p)\log 3 + 2 + (p/4)\log 3)(n/(n+1)).$$

Maximizing over p we get that for

$$p = 2/(2 + 3^{3/4}) \approx 0.4672$$

the sum-rate is approximately 4.493, which improves upon the sum-rate result of 4.4804 from Example 1.

To conclude this part, we show how to apply our ideas in order to construct two-write WOM codes for $q = 16$.

Example 3: In this example we demonstrate how to apply Theorem IV-A and its improvement to the case where $q = 16$. As the the scheme for this case is similar to the case $q = 8$, we only sketch the improvement to Theorem 4.

Let $5/9 \leqslant p \leqslant 1$ be a parameter to be determined later and $N = n + 1$. On the first write, we store only words $w_1 \in \{0,\ldots,8\}^n$ that have at least pn coordinates with values in $\{0,1,2,3,4\}$. It follows that in the second write we can write an additional $w_3 \in \{4,5,6,7\}^{np/8}$ on top of a word $\boldsymbol{w}_2 \in \{0,9,\ldots,15\}^n$. Thus we get an asymptotic sum-rate of

$$\frac{n \cdot (h(p) + p\log 5 + (1-p)\log 4 + \log 8 + (p/8)\log 4)}{n+1},$$

and for $p = 0.597829711$ the sum-rate is larger than 6.314. Looking at concrete numbers we see that for $n = 5000$ and $p = 0.598$ our construction already gives a WOM code of sum-rate larger than 6.31, while the previously best known sum-rate is 6.3083 and the upper bound on the sum-rate is 7.0875.

C. Improvement Using the Construction of [11]

By combining our techniques with Construction B of [11] we can achieve better sum-rates. We call this method Construction 2 and we show that the sum-rate is roughly $\log(q^2/3) + O(\log(q)/q)$.

First, let us remind Construction B from [11], which achieves sum-rate $\log(q^2/3)$. According to Construction B, the cell levels are partitioned into three non-overlapping parts: $\{0,\ldots,q/3-1\}, \{q/3,\ldots,2q/3-1\}, \{2q/3,\ldots,q-1\}$ (for simplicity, we assume that q is a multiple of 3). On each write, two messages are written: the first one is a binary word and the second is a word over an alphabet of size $q/3$. Let $\boldsymbol{w}_{1,1} \in \{0,1\}^n$, $\boldsymbol{w}_{1,2} \in \{0,\ldots,q/3-1\}^n$ be the messages on the first write and $\boldsymbol{w}_{2,1} \in \{0,1\}^n$, $\boldsymbol{w}_{2,2} \in \{0,\ldots,q/3-1\}^n$ be the messages on the second write. While there is no constraint on the messages $\boldsymbol{w}_{1,2}$ and $\boldsymbol{w}_{2,2}$, the messages $\boldsymbol{w}_{1,1}$ and $\boldsymbol{w}_{1,2}$ belong to some two-write binary WOM code. On the first write, the i-th cell is programmed to level $q/3(w_{1,1})_i + (w_{1,2})_i$ and on the second write it is programmed to level $q/3 + q/3(w_{2,1})_i + (w_{2,2})_i$. Since the words $\boldsymbol{w}_{1,1}$ and $\boldsymbol{w}_{1,2}$ belong to a two-write WOM code, it is easy to verify that on the second write, no cell can decrease its level. It is also clear how to decode on each write and for more details we refer the reader to [11]. If a capacity achieving two-write WOM code is used in this construction then the sum-rate approaches $\log 3 + 2\log(q/3) = \log(q^2/3)$.

We now show the sum-rate result of Construction 2.

Theorem 5: Construction 2 achieves sum-rate $\log(q^2/3) + O(\log(q)/q)$.

Proof: As done in Construction B from [11], the cell levels are partitioned into the same three groups while we also assume here for simplicity that q is a multiple of 3. Let $\mathcal{C}$ be a capacity achieving two-write binary WOM code of length n. We construct a code with the same length which we describe its first and second writes as follows:

First Write: We get two words, $\boldsymbol{w}_{1,1} \in \{0,1\}^n$ and $\boldsymbol{w}_{1,2} \in \{0, q/3-1\}^n$ such that $\boldsymbol{w}_{1,1}$ belongs as a first write codeword in $\mathcal{C}$. Since $\mathcal{C}$ is a capacity achieving code the weight of the binary word $\boldsymbol{w}_{1,1}$ on the first write is at most $n/3$. Let I' be the set $I' = \{i \mid (w_{1,1})_i = 0\}$ and let $0 < \ell < q/3$ be some integer whose value will be determined later. Let $\gamma_1, \gamma_2, \ldots, \gamma_\ell$ be the ℓ most frequent symbols of the word $\boldsymbol{w}_{1,2}$ in the set I' and we define a new word $\boldsymbol{w}'_{1,2}$ where the symbols $\gamma_1, \gamma_2, \ldots, \gamma_\ell$ are flipped with the symbols $0, 1, \ldots, \ell-1$ (here and later in the construction we assume that these changes are marked in some extra cells, however this does not affect the sum-rate of the code). Finally, the i-th cell is programmed to level $c_i = q/3(w_{1,1})_i + (w'_{1,2})_i$. The decoding is done in a very similar way.

Second Write: We get three words on this write, $\boldsymbol{w}_{2,1} \in \{0,1\}^n$ and $\boldsymbol{w}_{2,2} \in \{0,\ldots,q/3-1\}^n$ and $\boldsymbol{w}_3 \in \{\ell-1,\ldots,q/3-1\}^{3\ell n/(2q^2)}$. The word $\boldsymbol{w}_{2,1}$ belongs as a second write codeword in $\mathcal{C}$ while $\boldsymbol{w}_{1,1}$ was written on the first write. Let I'' be the set of all cells which their level on the first write is at most $\ell-1$, that is $I'' = \{i \mid c_i \leqslant \ell-1\}$ and note that

$$|I''| \geqslant \frac{\ell n/3}{q/3} = \ell n/q.$$

Let $\gamma \in \{0,\ldots,q/3-1\}$ be the most frequent symbol of $\boldsymbol{w}_{2,2}$ over the cells in I'' and as before we generate a word $\boldsymbol{w}'_{2,2}$ where γ is flipped with 0. Next, we let

$$I_0 = \{i \mid i \in I'' \text{ and } (w_{2,2})_i = \gamma \text{ and } (w_{2,1})_i = 0\},$$
$$I_1 = \{i \mid i \in I'' \text{ and } (w_{2,2})_i = \gamma \text{ and } (w_{2,1})_i = 1\}.$$

Assume that $|I_0| \geqslant |I_1|$, so

$$|I_0| \geqslant (\ell n/q)/(2q/3) = 3\ell n/(2q^2).$$

We marked this case on an additional cell as well and it will be clear how to modify the construction for the case $|I_0| < |I_1|$.

TABLE II
SUM-RATE RESULTS FOR $q = 8$ AND $q = 16$

q	Simple construction	Renaming symbols	First improvement	Combining with [11]	Upper bound
8	4.3219	4.4804	4.493	4.7368/4.829	5.1699
16	6.1699	6.3084	6.314	6.466/6.5585	7.0875

We let I be a subset of the first $3\ell n/(2q^2)$ cells in I_0. Now, we are ready to write the word $\boldsymbol{w}_3$ as follows:

1) If $(w_{2,1})_i = 1$ then the i-th cell is programmed to be $2q/3 + (\boldsymbol{w}'_{2,2})_i$.
2) If $(w_{2,1})_i = 0$ and $i \notin I$ then the i-th cell is programmed as $q/3 + (\boldsymbol{w}'_{2,2})_i$.
3) If $(w_{2,1})_i = 0$ and $i \in I$ and it is the j-th element in I then the i-th cell is programmed as $(w_3)_j$.

For decoding, the word $\boldsymbol{w}_{2,1}$ is decoded by $(w_{2,1})_i = 1$ if $c_i \geqslant 2q/3$ and otherwise $(w_{2,1})_i = 0$. The word $\boldsymbol{w}_{2,1}$ is decoded by $(w_{2,1})_i = (c_i \bmod q/3)$ if $c_i \geqslant q/3$ and otherwise $(w_{2,1})_i = 0$. The word $\boldsymbol{w}_3$ is determined according to the first $3\ell n/(2q^2)$ cells of value less than $q/3$ and they determine the $3\ell n/(2q^2)$ symbols of $\boldsymbol{w}_3$.

Finally, for large block lengths, we get that the sum-rate approaches

$$\log 3 + 2\log(q/3) + \frac{3\ell}{2q^2} \cdot \log\left(\frac{q}{3} - \ell + 1\right)$$
$$= \log(q^2/3) + \frac{3\ell}{2q^2} \cdot \log\left(\frac{q}{3} - \ell + 1\right)$$

Now, if we choose $\ell = q/3 - \frac{q}{\lfloor \log q \rfloor}$, we get sum-rate

$$\log(q^2/3) + \frac{3(q/3 - \frac{q}{\lfloor \log q \rfloor})}{2q^2} \cdot \log\left(\frac{q}{\lfloor \log q \rfloor} + 1\right),$$

which for q large enough is in the order of $\log(q^2/3) + O(\log(q)/q)$. ∎

To conclude this section, we summarize the sum-rate results for two writes over $q = 8$ and $q = 16$, using all the techniques we discussed and mentioned, in Table II. Note that the construction we took from [11] uses a two-write WOM code. The first number in the fifth column corresponds to the case where one takes the best known two-write WOM code from [28] of sum-rate 1.4928. The second number corresponds to the case where the WOM code achieves the maximum sum-rate $\log 3$. Note that the first number corresponds to codes with a reasonable block-length, whereas the second one requires exponentially longer block-length.

V. MULTIPLE WRITES WOM CODES OVER LARGE ALPHABETS

In this section we consider the case of WOM codes with $t > 2$ over an alphabet of size q. For ease of notation, a t-write WOM codes which uses n cells of q levels and achieves sum-rate r is denoted as an (n, t, q, r)-*code*. Furthermore, if the WOM code has sum-rate which is greater than r then we denote it as an $(n, t, q, > r)$-*code*.

We begin with an easy observation. If $t_1 + t_2 = t$ and $q_1 + q_2 = q + 1$ and there exist an (n, t_1, q_1, r_1)-code and an (n, t_2, q_2, r_2)-code then we can combine them to get an $(n, t, q, r_1 + r_2)$-code. Indeed, in the first t_1 writes we use the first code and write using symbols from $\{0, 1, \ldots, q_1 - 1\}$. In the last t_2 writes we use the second code and write using the symbols $\{q_1 - 1, \ldots, q - 1\}$.

As in Section IV, if we had some information on the distribution of symbols after the first t_1 writes then, in principal, it would have allowed us to encode more bits in the last t_2 writes. We demonstrate this idea by designing codes over the alphabet $q = 8$ for the cases $t = 3, 4$.

We first demonstrate our ideas by giving an $(n, 2, 5, >3.28)$-code for the first 2 writes and then combine it with a simple $(n, 1, 4, 2)$-code to get an $(n, 3, 8, >5.28)$-code. We then analyze the weight distribution after the first two writes and obtain an $(n, 3, 8, >5.36)$-code.

An $(n, 2, 5, >3.16)$-code is constructed by simply writing an arbitrary word in $\{0, 1, 2\}$ in the first write and an arbitrary word in $\{2, 3, 4\}$ in the second write. An improvement can be achieved if we insist that the word in the first write contained at least $1/3$ fraction of zeros and then, perhaps by changing the meaning of some symbols in the word of the second write as was done in Section IV, in the second write we get that roughly $1/9$ of the coordinates had the value 0 in both writes. Therefore, we can encode another binary word of length $n/9$ into these locations, using the symbols $\{0, 1\}$. Thus, the sum-rate is $\log(3) + \log(3) + 1/9 > 3.28$. Now, if we insist that in the third write the word is again equidistributed in $\{0, 5, 6, 7\}$,[4] then we get (again, we may need to change a value in the word to be written in that round), that there is a fraction of $1/12$ of the coordinates that were in $\{0, 1, 2\}$ after the first two writes and that got the value 0 in the third write. We can therefore encode another bit on those coordinates using the values $\{2, 3\}$. Thus, the sum-rate is

$$\log(3) + (\log(3) + 1/9) + (\log 4 + 1/12) > 5.36.$$

We now give another example consisting of an $(n, 2, 6, >3.75)$-code and a simple $(n, 1, 3, 2)$-code, that yields an $(n, 3, 8, >5.33)$-code. Again, by analyzing the weight distribution after the first two writes, we manage to push the sum-rate to be greater than 5.44.

In the first write, we store an equidistributed word in $\{0, 1, 2, 3\}$. In the second write, we write a word w_2 in $\{0, 4, 5\}$ (where 0 means leave the current value unchanged). Again, by possible switching the meaning of the alphabet, we can assume that there is at least a $1/6$ fraction of the coordinates that had value in $\{0, 1\}$ after the first write and 0 in the second write. On those coordinates we can encode another bit by using the values $\{1, 2\}$ (on the other coordinates that had value 2 we write 3 instead). This gives sum-rate $\log 4 + (\log 3 + 1/6) > 3.75$. If we combined it with a third write word in $\{0, 6, 7\}$ then we get sum-rate >5.33. However, as before, we can assume that $1/9$ of the elements had value 0 in both the second and third writes, and so now they store value in $\{0, 1, 2, 3\}$ and therefore, by pushing other values to 5 if necessary, we can store another $n/9$ bits using the values $\{3, 4\}$.

[4] A word $\boldsymbol{w}$ is equidistributed in a certain set $A \subseteq \{0, \ldots, q - 1\}$ if every symbol in A appears the same number of times in $\boldsymbol{w}$.

This gives sum-rate

$$\log 4 + (\log 3 + 1/6) + (\log 3 + 1/9) > 5.44.$$

Finally, we remark that by combining our approach of a $(n, 2, 5, >3.28)$-code with the $(n, 2, 2, >2.98)$-code of [11], we get an $(n, 4, 8, >6.26)$-code.

VI. Conclusion

In this paper, we studied binary three-write and non-binary WOM codes. The upper bound on the sum-rate of binary three-write WOM code is 2 and the recently best known construction in [24] had sum-rate 1.809. We used similar techniques from the construction in [24] in order to achieve sum-rate approaching 1.885. We then studied the problem of non-binary WOM codes by using similar ideas of the three-write WOM codes construction that reduce the number of cells with high level. Accordingly, we gave constructions of two-write WOM codes which improved upon the previously best known sum-rate while attaining a reasonable block-length. We also gave an arbitrary scheme to construct multiple writes WOM codes and showed specific examples for three and four writes using 8 levels.

Acknowledgement

The authors sincerely thank the anonymous reviewers for valuable comments and suggestions in improving the presentation of the paper.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[2] A. Bhatia, A. R. Iyengar, and P. H. Siegel, "Multilevel 2-cell t-write codes," in *Proc. IEEE Inf. Theory Workshop*, Lausanne, Switzerland, Sep. 2012, pp. 247–251.

[3] D. Burshtein and A. Strugatski, "Polar write once memory codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 1972–1976.

[4] P. Cappelletti, C. Golla, P. Olivio, and E. Zanoni, *Flash Memories*. Boston, MA, USA: Kluwer, 1999.

[5] Y. Cassuto and E. Yaakobi, "Short q-ary WOM codes with hot/cold write differentiation," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 1391–1395.

[6] G. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories," *IEEE Trans. Inf. Theory*, vol. 32, no. 5, pp. 697–700, Sep. 1986.

[7] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.

[8] A. Fiat and A. Shamir, "Generalized 'write-once' memories," *IEEE Trans. Inf. Theory*, vol. 30, no. 3, pp. 470–480, Sep. 1984.

[9] F. W. Fu and A. J. H. Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 308–313, Jan. 1999.

[10] R. Gabrys and L. Dolecek, "Characterizing capacity achieving write once memory codes for multilevel flash memories," in *Proc. IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, Jul./Aug. 2011, pp. 2517–2521.

[11] R. Gabrys, E. Yaakobi, L. Dolecek, P. H. Siegel, A. Vardy, and J. K. Wolf, "Non-binary WOM-codes for multilevel flash memories," in *Proc. IEEE Inf. Theory Workshop*, Paraty, Brazil, Oct. 2011, pp. 40–44.

[12] P. Godlewski, "WOM-codes construits à partir des codes de Hamming," *Discrete Math.*, vol. 65, no. 3, pp. 237–243, Jul. 1987.

[13] K. Haymaker and C. A. Kelley. (May 2012). "Geometric WOM codes and coding strategies for multilevel flash memories." [Online]. Available: http://arXiv:1206.5762v1

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

[15] Q. Huang, S. Lin, and K. A. S. Abdel-Ghaffar, "Error-correcting codes for flash coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 9, pp. 6097–6108, Sep. 2011.

[16] J. Justesen, "Class of constructive asymptotically good algebraic codes," *IEEE Trans. Inf. Theory*, vol. 18, no. 5, pp. 652–656, Sep. 1972.

[17] S. Kayser, E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple-write WOM-codes," in *Proc. 48th Annu. Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, Sep. 2010, pp. 1062–1068.

[18] B. M. Kurkoski, "Lattice-based WOM codebooks that allow two writes," in *Proc. Int. Symp. Inf. Theory Appl.*, Honolulu, HI, USA, Oct. 2012, pp. 101–105.

[19] B. M. Kurkoski, "Notes on a lattice-based WOM construction," in *Proc. 34th Symp. Inf. Theory Appl.*, Iwate, Japan, Nov./Dec. 2011, pp. 520–524.

[20] J. L. Massey, "Threshold decoding," Res. Lab. Electron., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. 410, 1963.

[21] F. Merkx, "Womcodes constructed with projective geometries," *Traitement Signal*, vol. 1, no. 2, pp. 227–231, 1984.

[22] R. L. Rivest and A. Shamir, "How to reuse a 'write-once≓ memory'," *Inf. Control*, vol. 55, nos. 1–3, pp. 1–19, Dec. 1982.

[23] A. Shpilka, "Capacity-achieving multiwrite WOM codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1481–1487, Mar. 2014.

[24] A. Shpilka, "New constructions of WOM codes using the Wozencraft ensemble," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4520–4529, Jul. 2013.

[25] J. K. Wolf, A. D. Wyner, J. Ziv, and J. Körner, "Coding for a write-once memory," *AT&T Bell Labs. Tech. J.*, vol. 63, no. 6, pp. 1089–1112, 1984.

[26] Y. Wu, "Low complexity codes for writing a write-once memory twice," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, pp. 1928–1932.

[27] Y. Wu and A. Jiang, "Position modulation code for rewriting write-once memories," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3692–3697, Jun. 2011.

[28] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," in *Proc. IEEE Inf. Theory Workshop*, Dublin, Ireland, Sep./Aug. 2010, pp. 1–5.

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

[30] E. Yaakobi and A. Shpilka, "High sum-rate three-write and non-binary WOM codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 1386–1390.

Eitan Yaakobi (S'07-M'12) received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion - Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011.

He is currently a postdoctoral researcher in electrical engineering at the California Institute of Technology, Pasadena and he is also affiliated with the Center for Magnetic Recording Research at the University of California, San Diego. His research interests include information and coding theory with applications to non-volatile memories, associative memories, data storage and retrieval, and voting theory. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.

Amir Shpilka received the B.A. and Ph.D. degrees in mathematics from the Hebrew University, Jeruslam, Israel, in 1996 and 2001, respectively. He is currently a faculty member at the department of computer science at the Technion-Israel Institute of Technology, Haifa, Israel. His main research area is computational complexity but he is also interested in broader aspects of theoretical computer science and coding theory.