On The Generalization of Error-Correcting WOM Codes

Anxiao (Andrew) Jiang

Computer Science Dept., Texas A&M University, College Station, TX 77843, USA

Email: ajiang@cs.tamu.edu

Abstract—WOM (Write Once Memory) codes are codes for efficiently storing and updating data in a memory whose state transition is irreversible. Storage media that can be classified as WOM includes flash memories, optical disks and punch cards. Error-correcting WOM codes can correct errors besides its regular data updating capability. They are increasingly important for electronic memories using MLCs (multi-level cells), where the stored data are prone to errors. In this paper, we study error-correcting WOM codes that generalize the classic models. In particular, we study codes for *jointly* storing and updating multiple variables – instead of one variable – in WOMs with multi-level cells. The error-correcting codes we study here are also a natural extension of the recently proposed *floating codes* [7].

We analyze the performance of the generalized error-correcting WOM codes and present several bounds. The number of valid states for a code is an important measure of its complexity. We present three optimal codes for storing two binary variables in n q-ary cells, where $n = 1, 2, 3$, respectively. We prove that among all the codes with the minimum number of valid states, the three codes maximize the total number of times the variables can be updated.

I. INTRODUCTION

The Write Once Memory (WOM) was first introduced by Rivest and Shamir [10] to model those memories where each basic storage element can transit from a 0-state to a 1-state but not *vice versa*. Early examples of WOM include punch cards and optical disks. In recent years, flash memories, which use floating gates as storage cells, have emerged as an important family of memories that can be modelled as WOMs. In flash memories, every memory cell has a threshold voltage that is one of q possible values: $0, 1, \cdots, q - 1$. The threshold voltage is the *state* of the cell. Moving the cell from a lower state to a higher state can be realized efficiently using either the hot-electron injection mechanism or the Fowler-Nordheim tunneling mechanism. However, moving the cell from a higher state to a lower state is much more expensive, because it requires erasing and re-writing all the data in a memory block, which typically consists of about 128 kilobytes. The erasure and rewriting of a block are not only very slow, but also degrade the cells' quality and shorten the memory's lifetime. Currently, a flash memory's lifetime is bounded by around 10^5 program-erase cycles. For this reason, the operation of lowering a cell's state should be delayed as much as possible. When we consider the time period between two block erasure operations, a cell's state can only move upward in its q states. When $q > 2$, the cell is called a multi-level cell (MLC).

The data in memories often need to be updated, especially for applications such as file systems, programs, etc. Since a WOM's state transition is irreversible, the number of updates it allows is limited. Many researchers have studied WOM codes, where a single variable is stored in a WOM, and the WOM code aims at maximizing the number of times the variable can be rewritten (i.e., updated) [2], [3], [6], [8], [11]. Multiple families of WOM codes, including linear codes [1], [10], tabular codes [10], codes based on projective geometries [9], etc., have been invented. WOM codes that can correct errors have also been explored [4], [12].

The capability of error correction is especially important for electronic memories using multi-level cells. Using MLC is a fundamental approach for increasing the data density. In flash memories, q-ary cells where $q = 4$ up to 256 have been implemented. It, however, brings reliability issues. Various reasons can make the state of a cell be read incorrectly, especially for adjacent states.

We model the problem we study as follows. The memory consists of n cells, where each cell has q states: $0, 1, \cdots, q-1$. A cell can change from state i to state j if and only if $i < j$. ($0 \leq i, j \leq q - 1$.) k variables are stored in the memory, where each variable takes its value from an alphabet of size l: $\{0, 1, \cdots, l - 1\}$. By default, initially, all the cells are in state 0, and all the variables have the value 0. Every write changes the value of exactly one variable. We use t to denote the maximum number of writes allowed by the memory in the worst case. Specifically, for any sequence of writes, the first t of them are guaranteed to be implementable.

We use $(c_1, c_2, \cdots, c_n)$ – called the *cell state vector* – to denote the states of the n cells, where $c_i \in [0, q - 1]$ is the state of the i-th cell. The value of $\sum_{i=1}^{n} c_i$ is called the *weight* of the cell state vector. For any two cell state vectors $A = (c_1, c_2, \cdots, c_n)$ and $B = (c'_1, c'_2, \cdots, c'_n)$, their L_1 *distance* is defined to be $d_L(A, B) = \sum_{i=1}^{n} |c_i - c'_i|$. We use $(v_1, v_2, \cdots, v_k)$ – called the *variable vector* – to denote the values of the k variables, where $v_i \in [0, l-1]$ is the value of the i-th variable.

Our error model has three parameters: Δ^+, Δ^-, and E. Here, Δ^+ (resp., Δ^-) is the maximum magnitude of a single-cell error in the upward direction (resp., in the downward direction), and E is the maximum total magnitude of the errors. (Naturally, $E \geq \Delta^+ \geq 0$ and $E \geq \Delta^- \geq 0$.) Specifically, let's use $(e_1, e_2, \cdots, e_n)$ – called the *error vector* – to denote the n additive errors in the n cells. For $i = 1, 2, \cdots, n$, it makes the state of the i-th cell, c_i, to be mistakenly read as

$c_i + e_i \in [0, q-1]$. The errors satisfy the following constraints: $-\Delta^- \le e_i \le \Delta^+$ for all i, and $\sum_{i=1}^{n} |e_i| \le E$. It is a general error model for memories using MLCs.

An error-correcting WOM code maps every cell state vector (whether it contains errors or not) to a variable vector. That is the *decoding*. We assume that errors happen only in the *read* phase – which are errors caused by faults or noise in the reading circuit – and therefore, the real states of the cells are always correct. So for the writing circuit, the cells' states are always error-free. (Correcting errors in the real states of the cells using rewriting is another fascinating topic, which we will study later.) Given the current (error-free) cell state vector and a write request, the code changes the cell state vector to a new one corresponding to the new variable vector. That is the *encoding*. The error-correcting WOM code should be able to correct any error with parameters Δ^+, Δ^- and E. Given these constraints, the error-correcting WOM code that maximizes t – the total number of writes – is called *optimal*.

Our objective is to look for optimal error-correcting WOM codes. The model we study generalizes the traditional WOM codes, because here we consider the *joint* storage, update and error correction of multiple variables instead of just one variable. Having multiple variables (e.g., words) stored in a memory is common in nearly all practices. By considering the joint coding of multiple variables, we can often achieve substantially better performance than separate coding for each individual variable. This motivation has led to the study of *floating codes* – recently proposed in [7], – which generalize traditional WOM codes by using joint coding of multiple variables for data storage and update. The error-correcting codes we study here are a natural extension of the floating codes.

An example of error-correcting WOM codes is given in Fig. 1(a). The vector inside each circle is a (valid) cell state vector, and the vector beside the circle, which is in the bold font, is the corresponding variable vector. The two arrows leaving a circle indicate the next two cell state vectors given the $2 = k(l-1)$ possible write requests. For example, if the first write changes the second variable v_2 to 1, and the second write changes the first variable v_1 to 1, then the variable vector changes as $(0,0) \to (0,1) \to (1,1)$, and the cell state vector changes as $(0,0) \to (0,3) \to (3,3)$. The code allows $t = 2$ writes, and can correct any error with parameters $\Delta^+ = \Delta^- = E = 1$. For example, if the true cell state vector is $(c_1 = 6, c_2 = 0)$ and the error vector is $(e_1 = -1, e_2 = 0)$, then the cell state vector will be incorrectly read as $(c_1 = 5, 0)$. Since there is only one valid cell state vector within L_1 distance 1 from $(5,0)$, which is $(c_1 = 6, c_2 = 0)$, the cell state vector and the variable vector will be correctly decoded as $(6,0)$ and $(0,0)$, respectively.

In this paper, we study both code construction and the performance analysis of general error-correcting WOM codes. We present several bounds for performance analysis. The number of valid cell state vectors in a WOM code is an important measure of the code's complexity. We present three codes for storing two binary variables in n q-ary cells, where

$n = 1, 2, 3$, respectively. The three codes have $t = \lfloor \frac{q-1}{6} \rfloor$, $\lfloor \frac{4q-6}{7} \rfloor$ and $q - 2 - \lfloor \frac{q}{7} \rfloor$, respectively, which are provably optimal among all the codes with the minimum number of valid cell state vectors. The details are as follows.

II. BOUNDS AND SIMPLE CODE CONSTRUCTIONS

Two example codes are shown in Fig. 1(a), (b). In fact, they show two basic ways to derive error-correcting WOM codes from ordinary WOM codes. Both example codes are derived from the ordinary WOM code in Fig. 1(c). The first code (shown in Fig. 1(a)) is an example of the following approach: given an ordinary WOM code, map the cell state i to the cell state $i \cdot (\Delta^+ + \Delta^- + 1)$. In this way, any two valid states for a cell are sufficiently far away, thus allowing error correction. The second code (shown in Fig. 1(b)) is an example of the following approach: given an ordinary WOM code, map every cell to $2E + 1$ cells and use the repetition code, which can correct any E errors. Since both approaches change either q or n, the appropriate ordinary WOM code need to be selected. We will show later that when n and q are sufficiently large, these two approaches – especially the first one – can actually build codes quite close to optimal.
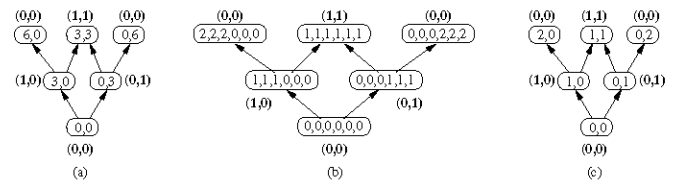

Fig. 1. (a) An error-correcting WOM code with $k = 2, l = 2, n = 2, q = 7, t = 2, \Delta^+ = \Delta^- = E = 1$. (b) An error-correcting WOM code with $k = 2, l = 2, n = 6, q = 3, t = 2, \Delta^+ = \Delta^- = E = 1$. (c) A non-error-correcting WOM code with $k = 2, l = 2, n = 2, q = 3, t = 2$. (For non-error-correcting WOM codes, $\Delta^+ = \Delta^- = E = 0$.)

We first present in Theorem 1 a necessary and a sufficient condition for a WOM code to be an error-correcting code. Both conditions are tight.

Note that a *valid* cell state vector is a cell state vector that can be truly reached by the memory after a sequence of writes, when there is no error. For example, all the cell state vectors shown in Fig. 1 are valid cell state vectors. Given a WOM code, we use d_{min} to denote the *minimum L_1 distance* between two distinct valid cell state vectors.

Theorem 1: For a WOM code to be an error-correcting WOM code, the following condition is sufficient:

$$d_{min} \ge 2E + 1;$$

the following condition is necessary:

$$d_{min} \ge \Delta^+ + \Delta^- + 1.$$

Proof: The sufficient condition can be very easily proved with the standard ball packing argument. Now let's consider the necessary condition. The proof is by contradiction. Assume that $d_{min} \le \Delta^+ + \Delta^-$. We need to show that for any two valid cell state vectors $A = \{a_1, a_2, \cdots, a_n\}$, $B = \{b_1, b_2, \cdots, b_n\}$ whose L_1 distance is d_{min}, there are two error patterns that

can lead A and B to the same cell state vector, thus creating a case where decoding becomes impossible.

For $i = 1, 2, \cdots, n$, define S_i to be such a set of integers: $S_i = \{s| \min\{a_i, b_i\} \leq s \leq \min\{a_i, b_i\} + \Delta^+, \max\{a_i, b_i\} - \Delta^- \leq s \leq \max\{a_i, b_i\}\}$. Since $\max\{a_i, b_i\} - \min\{a_i, b_i\} = |a_i - b_i| \leq \sum_{j=1}^{n} |a_j - b_j| = d_{min} \leq \Delta^+ + \Delta^-$, we get $S_i \neq \emptyset$.

Define P, Q to be such two sets of integers: $P = \{i | 1 \leq i \leq n, a_i \leq b_i\}$, and $Q = \{i | 1 \leq i \leq n, a_i > b_i\}$.

Choose $w_1, w_2, \cdots, w_n$ to be non-negative integers that maximize the value of $\sum_{i=1}^{n} w_i$ subject to the following two constraints: (1) if $i \in P$, then $a_i + w_i \in S_i$; otherwise (namely, if $i \in Q$), $a_i - w_i \in S_i$; (2) $\sum_{i=1}^{n} w_i \leq E$. It is not difficult to see that those integers $w_1, w_2, \cdots, w_n$ do exist.

Define two error vectors $\varepsilon_A = \{e_1^A, e_2^A, \cdots, e_n^A\}$, $\varepsilon_B = \{e_1^B, e_2^B, \cdots, e_n^B\}$ as follows: if $i \in P$, then $e_i^A = w_i$ and $e_i^B = -(|a_i - b_i| - w_i)$; if $i \in Q$, then $e_i^A = -w_i$ and $e_i^B = |a_i - b_i| - w_i$. Let's call a *valid error vector* to be an error vector that satisfies those constraints imposed by the parameters Δ^+, Δ^-, E. It is not hard to see that ε_A is a valid error vector. Now we show that so is ε_B. By the definition of S_i, we can see that $-\Delta^- \leq e_i^B \leq \Delta^+$. To show that $\sum_{i=1}^{n} |e_i^B| \leq E$, consider three cases: *Case 1:* $\sum_{i=1}^{n} w_i = E$. In this case, $\sum_{i=1}^{n} |e_i^B| = d_L(A, B) - \sum_{i=1}^{n} |e_i^A| = d_{min} - \sum_{i=1}^{n} w_i \leq \Delta^+ + \Delta^- - E \leq 2E - E = E$. *Case 2: there exists j such that "$j \in P$ and $w_j = \Delta^+$" or "$j \in Q$ and $w_j = \Delta^-$".* In this case, $\sum_{i=1}^{n} |e_i^B| = d_L(A, B) - \sum_{i=1}^{n} |e_i^A| = d_{min} - \sum_{i=1}^{n} w_i \leq d_{min} - w_j \leq \Delta^+ + \Delta^- - \min\{\Delta^+, \Delta^-\} \leq E$. *Case 3: it is neither of the previous two cases.* In this case, $\sum_{i=1}^{n} w_i < E$. Since $w_1, w_2, \cdots, w_n$ maximize $\sum_{i=1}^{n} w_i$ by their definition, we get $e_i^B = 0$ for all i, so $\sum_{i=1}^{n} |e_i^B| \leq E$.

So both ε_A and ε_B are valid error vectors; what's more, $A + \varepsilon_A = B + \varepsilon_B$. So when the received (i.e., read) cell state vector is $A + \varepsilon_A$, we cannot tell if the true cell state vector is A or B. Thus decoding fails, finishing the proof based on contradiction. So the necessary condition is proved. ∎

The following theorem shows the asymptotic performance of error-correcting WOM codes when $n \to \infty$ and $q \to \infty$.

Theorem 2: Let $a = \Delta^+ + \Delta^- + 1$. Let k, l be fixed, and let $n \to \infty$, $q \to \infty$. Then, for any optimal error-correcting WOM code,

$$\frac{qn}{a} - o(qn) \leq t \leq \frac{(q-1)n}{a}.$$

Proof: First, we show the lower bound to t. It has been shown in [10] that when $q = 2$ and $n \to \infty$, there is an ordinary WOM code – named the *tabular code* – that achieves $t = n - o(n)$. Although the tabular code is for a single variable, we can see the k variables here as one super variable from an alphabet of size l^k. In this way, the tabular code can be applied to the k variables and still achieve $t = n - o(n)$. Now when $q \to \infty$, we first apply the tabular code to the cell states 0 and 1; then, apply it to cell states 1 and 2; then to cell states 2 and 3; so on $\cdots$ We obtain an ordinary WOM code with $t = qn - o(qn)$ in this way.

Now we use an approach of deriving error-correcting WOM codes from ordinary WOM codes that we presented earlier.

The approach is to map the cell states $0, 1, 2, 3 \cdots$ to the cell states $0, a, 2a, 3a \cdots$. By applying this approach, we get an error-correcting WOM codes with $t = \frac{qn}{a} - o(qn)$. So for the optimal error-correcting WOM code, $t \geq \frac{qn}{a} - o(qn)$.

We now show the upper bound. By Theorem 1, $d_{min} \geq a$. So every write increases the *weight* of the cell state vector by at least a. The weight of the cell state vector can never exceed $(q-1)n$. So the upper bound $t \leq \frac{(q-1)n}{a}$ holds. ∎

Theorem 2 shows that asymptotically, $t = \frac{(q-1)n}{\Delta^+ + \Delta^- + 1} - o(qn)$. In practice, the most common type of errors have $\Delta^+ = \Delta^- = 1$. That is, only two adjacent cell levels may become indistinguishable to the reading circuit. In such a case, we get $t = \frac{(q-1)n}{3} - o(qn)$. It means that even if there are many errors, the errors reduce the value of t by only a factor of 3.

It has essentially been shown in the proof of Theorem 2 that when n, q are sufficiently large, the first approach introduced at the beginning of this section – which is for deriving error-correcting WOM codes from ordinary WOM codes – constructs codes very close to optimal. It is simple to see that the second approach introduced there can construct asymptotically optimal codes. This can be regarded as a situation where the separation between source coding and channel coding holds.

III. THREE OPTIMAL CODES WITH COMPLEXITY CONSTRAINTS

Two basic approaches for code construction have been shown in Section II. However, they usually produce codes that are not optimal when n or q is small. In fact, optimal codes often exhibit irregular internal structures. For such codes, the basic method for decoding is to use a lookup table, which maps cell state vectors to variable vectors. The complexity of this decoding method is proportional to the total number of valid cell state vectors, which we shall call the *cardinality* of the code. It is, therefore, useful to study codes with minimum cardinalities.

In this section, we study error-correcting WOM codes for two binary variables. That is, $k = l = 2$. Also, we let $E = \Delta^+ = \Delta^- = 1$. Namely, the code corrects any single error of magnitude 1. It is of special interest to study *binary variables* because in electronic memories, the 16 bits of a word are often stored in 16 parallel blocks at the same address. Consequently, writing a word becomes writing a bit in each block. For $i = 0, 1, 2, 3 \cdots$, we define the *cell state vectors of the i-th generation* to be the valid cell state vectors that the memory can reach after exactly i writes. (For example, in Figure 1 (a), the cell state vectors in the 1st generation are $(3, 0), (0, 3)$, and those in the 2nd generation are $(6, 0), (3, 3), (0, 6)$.) When $k = 2$ and $l = 2$, it is simple to see that when $i > 0$, the cell state vectors in the i-th generation correspond to exactly two variable vectors: if i is odd, the two variable vectors are $(1, 0)$ and $(0, 1)$; if i is even, they are $(0, 0)$ and $(1, 1)$. So when $i > 0$, the i-th generation contains at least two cell state vectors. Therefore, the *minimum cardinality* of a code is $2t + 1$.

We present three codes with the minimum cardinality $2t + 1$, respectively for $n = 1, 2, 3$ and arbitrary q. We show that they are optimal among all the codes with cardinality $2t + 1$, in the

sense that they maximize t, the number of writes. What's more, all the three codes have periodic internal structures, which significantly reduces the decoding complexity using the lookup method. In fact, the size of the lookup table is only 4, 9 and 15, respectively, regardless of how large q is. In the following, we first present the code for $n = 3$, then the codes for $n = 2$ and 1.

A. Optimal Code for $n = 3$

The code for $n = 3$ is shown in Fig. 2. Its internal structure has a periodic pattern, where every period consists of six consecutive generations. To see it, observe the 2nd generation of cell state vectors and the 8th generation. The 2nd generation consists of cell state vectors $(c_1 = 2, c_2 = 3, c_3 = 1)$ and $(c_1 = 3, c_2 = 2, c_3 = 2)$, while the 8th generation consists of $(c'_1 = 10, c'_2 = 8, c'_3 = 9)$ and $(c'_1 = 9, c'_2 = 9, c'_3 = 10)$. The following simple mapping shows the relationship between these two generations: $c'_1 = c_2 + 7$, $c'_2 = c_3 + 7$, $c'_3 = c_1 + 7$. More generally, the above mapping holds for the i-th generation and the $(i+6)$-th generation, for $i = 2, 3, \cdots, 7$. So a period consists of six generations.

We keep building the code using such a pattern. The code has the following property: for $i = 1, 2, \cdots, 6$ and $j = 0, 1, 2, 3 \cdots$, the above simple mapping holds for the $(6j+i+1)$-th generation and the $(6(j+1)+i+1)$-th generation. The code can have infinitely many generations. When q is finite, we truncate the code to the maximum generation subject to the constraints that every cell's state does not exceed $q - 1$.

It is simple to prove that the code can correct any single error of magnitude 1. (Note that here $E = \Delta^+ = \Delta^- = 1$.) For a code with the structure as shown in Fig. 2, it is sufficient to verify that for any two cell state vectors either in the same generation or connected by an arrow, their L_1 distance is at least 3. Then by theorem 1, the code can correct single errors.
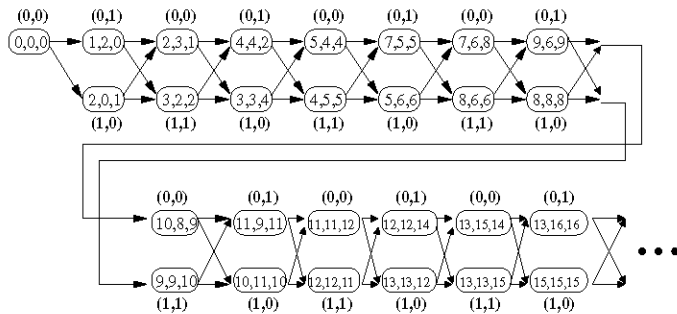

Fig. 2. An error-correcting WOM code with $k = 2, l = 2, n = 3, \Delta^+ = \Delta^- = E = 1$ and arbitrary q. It has $t = q - 2 - \lfloor \frac{q}{7} \rfloor$.

In the following, we prove that among all the codes with the minimum cardinality $2t + 1$, the code in Fig. 2 has the maximum value of t.

Lemma 1: For a code with cardinality $2t + 1$, let w_1, w_2 denote the weights of the two cell state vectors in the 2nd generation. Then, $\max\{w_1, w_2\} \geq 7$.

Proof: Since the code can correct any single error, for any two valid cell state vectors, their L_1 distance must be at

least 3. So every write increases the weight of the cell state vector by at least 3. Therefore, $w_1 \geq 6$, $w_2 \geq 6$.

Assume that $w_1 = w_2 = 6$. Then for the two cell state vectors in the 1st generation, their weights are both 3. Since the L_1 distance between those two vectors is at least 3, a simple enumeration shows that without loss of generality, those two vectors are one of the following seven pairs: $(1, 1, 1)$ and $(3, 0, 0)$, $(2, 1, 0)$ and $(0, 2, 1)$, $(2, 1, 0)$ and $(1, 0, 2)$, $(2, 1, 0)$ and $(0, 1, 2)$, $(2, 1, 0)$ and $(0, 3, 0)$, $(2, 1, 0)$ and $(0, 0, 3)$, $(3, 0, 0)$ and $(0, 3, 0)$. (Any missing case can be obtained by combining one of the seven cases with a permutation of the cells.) If those two vectors are $(1, 1, 1)$ and $(3, 0, 0)$, since every vector in the 1st generation can reach every vector in the 2nd generation through a write operation, we find that for both vectors in the 2nd generation, the three cells' states have to be at least $\max\{1, 3\} = 3$, $\max\{1, 0\} = 1$, $\max\{1, 0\} = 1$, respectively; since their weights are both 6, their L_1 distance is less than 3, which is a contradiction. The other six cases can be analyzed similarly. So the assumption cannot be true. So we get $\max\{w_1, w_2\} \geq 7$. ∎

Lemma 2: For a code with cardinality $2t + 1$, let w_1, w_2 denote the weights of the two cell state vectors in the i-th generation, where $i = 1, 2, 3 \cdots$. Then, $\max\{w_1, w_2\} \geq \frac{7(i-1)}{2} + 3$ if i is odd, and $\max\{w_1, w_2\} \geq \frac{7i}{2}$ if i is even.

Proof: By induction. When $i = 1$, $\max\{w_1, w_2\} \geq 3$; when $i = 2$, by lemma 1, $\max\{w_1, w_2\} \geq 7$. In both cases, the lemma holds. This is the base case of induction.

Assume that the lemma holds for any $i \leq 2m$, where m is a positive integer. Let a be the cell state vector in the $2m$-th generation whose weight is at least $7m$. Consider the two generations following the initial cell state vector $(0, 0, 0)$ (namely, the 1st and the 2nd generations), and compare them with the two generations following a (namely, the $(2m+1)$-th and the $(2m+2)$-th generations). We see that the analysis in the proof of lemma 1 also holds for the latter case, because all the constraints on L_1 distance must also be satisfied for the latter case. So when $i = 2m + 1$, $\max\{w_1, w_2\} \geq 7m + 3 = \frac{7(i-1)}{2} + 3$; when $i = 2m + 2$, $\max\{w_1, w_2\} \geq 7m + 7 = \frac{7i}{2}$. That completes the induction. ∎

Lemma 3: For a code with cardinality $2t + 1$, $t \leq q - 2 - \lfloor \frac{q}{7} \rfloor$.

Proof: The weight of a cell status vector cannot exceed $3(q - 1)$. By Lemma 2, we see that if $3(q-1) \mod 7$ is less than 3, then $t \leq 2 \cdot \lfloor \frac{3(q-1)}{7} \rfloor$; otherwise, $t \leq 2 \cdot \lfloor \frac{3(q-1)}{7} \rfloor + 1$.

We consider seven cases: (1) $q = 7i$; (2) $q = 7i + 1$; $\cdots$; (7) $q = 7i + 6$. Here $i \geq 1$ in cases (1), (2), and $i \geq 0$ in the latter five cases. As an example, consider case (1): $q = 7i$. In this case, $3(q - 1) = 3(7i - 1) = 21i - 3 = (21i - 7) + 4$. So $t \leq (3i - 1) \cdot 2 + 1 = 6i - 1$. However, if $t = 6i - 1$, then the weights of the two cell state vectors in the t-th generation are at least $7(3i - 1) + 3 = 3(q - 1) - 1$, so the L_1 distance of those two vectors is less than 3, which is a contradiction. So $t \leq 6i - 2 = q - 2 - \lfloor \frac{q}{7} \rfloor$. So the lemma holds. The other six cases can be analyzed in a similar way. For simplicity, we omit the details. ∎

It is not hard to verify that the code presented in Fig. 2 has

1394

$t = q - 2 - \lfloor \frac{q}{7} \rfloor$. It achieves the upper bound to t in lemma 3. So we get:

Theorem 3: The code presented in Fig. 2 has $t = q - 2 - \lfloor \frac{q}{7} \rfloor$. It is optimal among all the codes with the minimum cardinality $2t + 1$, in the sense that it maximizes t.

B. Optimal Codes for $n = 2$ and 1

The codes for $n = 2$ and 1 are shown in Fig. 3 (a) and (b), respectively. The code for $n = 1$ is very simple and clearly optimal. The code for $n = 2$ has a periodic internal structure, where every period consists of four generations. To see that, observe the 1st generation of cell state vectors and the 5th generation. The 1st generation consists of vectors $(c_1 = 3, c_2 = 0)$ and $(c_1 = 1, c_2 = 2)$, while the 5th generation consists of $(c'_1 = 7, c'_2 = 10)$ and $(c'_1 = 9, c'_2 = 8)$. The following simple mapping shows the relationship between these two generations: $c'_1 = c_2 + 7$, $c'_2 = c_1 + 7$. More generally, for $i = 1, 2, 3, 4$ and $j = 0, 1, 2, 3 \cdots$, the above simple mapping holds for the $(4j + i)$-th generation and the $(4(j + 1) + i)$-th generation.
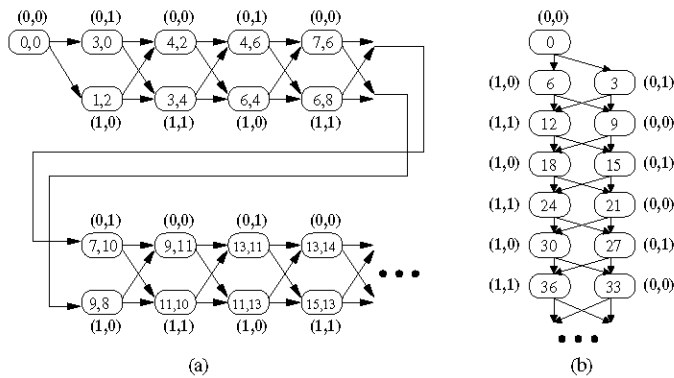

Fig. 3. Two error-correcting WOM codes with $k = 2, l = 2, \Delta^+ = \Delta^- = E = 1$ and arbitrary q. (a) $n = 2$ and $t = \lfloor \frac{4q-6}{7} \rfloor$; (b) $n = 1$ and $t = \lfloor \frac{q-1}{6} \rfloor$.

These two codes can be analyzes in a similar way as the code for $n = 3$. For simplicity, we skip the details. We present the final conclusion:

Theorem 4: The two codes presented in Fig. 3 has $t = \lfloor \frac{4q-6}{7} \rfloor$ (when $n = 2$) or $t = \lfloor \frac{q-1}{6} \rfloor$ (when $n = 1$). They are optimal among all the codes with the minimum cardinality $2t + 1$, in the sense that they maximize t.

IV. EXTENDED ANALYSIS ON GENERAL ERROR-CORRECTING WOM CODES

In section II, we have presented a bound to t when n and q are sufficiently large. In this section, we present a bound for general error-correcting WOM codes. When n is small, it can be (much) better than the following simple bound: $t \leq (q-1)n/(\Delta^+ + \Delta^- + 1)$. For example, when $k = l = 3, n = 2, \Delta^+ = \Delta^{-1} = E = 1$, Theorem 5 gives $t \leq 3\lceil \frac{q-1}{7} \rceil \approx \frac{q-1}{2.333}$, while the simple bound only gives $t \leq \frac{q-1}{1.5}$.

For any two cell state vectors $C = (c_1, c_2, \cdots, c_n)$ and $C' = (c'_1, c'_2, \cdots, c'_n)$, we say that C is above C' if and only if for $i = 1, 2, \cdots, n$, $c_i \geq c'_i$.

Theorem 5: Let $a = \Delta^+ + \Delta^- + 1$. Let $b = \lfloor \frac{a-1}{2} \rfloor$. Let $c = \sum_{i=0}^{k} \binom{k}{i}(l-1)^i \cdot \lfloor \frac{k-i}{2} + 1 \rfloor$. Let d denote the smallest positive integer such that $\binom{n+d}{n} \geq c\binom{n+b}{n} + b(c-1)$. Then, if $d - b > 0$, every error-correcting WOM code has $t \leq k \cdot \lceil \frac{(q-1)n}{d-b} \rceil$.

Proof: Starting with any valid cell state vector, k consecutive writes can make the variable vector reach or go through any of the l^k possible values (including the current variable vector). There are $\binom{k}{i}(l-1)^i$ variable vectors at Hamming distance i from the current variable vector, and every such variable vector can be reached after $i, i+2, i+4 \cdots$ writes, with the corresponding cell state vector's weight monotonically increasing. So k consecutive writes can make the cell state vector reach or go through c or more distinct values (including the current value). Let's pick c such cell state vectors, and denote them by $s_1, s_2, \cdots, s_c$. In particular, let s_1 denote the current (the starting) cell state vector.

For $i = 1, 2, \cdots, c$, let B_i denote the ball of radius b (measured by the L_1 distance) centered at s_i. By theorem 1, when $i \neq j$, the two balls B_i and B_j are disjoint. There are $\binom{n+b}{n}$ elements in B_i that are *above* s_i; and if $i > 1$, there are at least b elements in B_i that are above s_1 and whose weights are less than the weight of s_i by $1, 2, \cdots, b$, respectively. All the $c\binom{n+b}{n} + b(c-1)$ elements (which are all cell state vectors) discussed above are above s_1. It is not hard to see that among those $c\binom{n+b}{n} + b(c-1)$ elements, one of them has a weight that is greater than the weight of s_1 by at least d. So there exists a sequence of at most k writes that can raise the weight of the cell state vector by at least $d - b$. By partitioning t writes into such sequences, where each sequence contains at most k writes, we get the conclusion. ∎

REFERENCES

[1] G. D. Cohen, P. Godlewski and F. Merkx, " Linear binary code for write-once memories," *IEEE Transactions on Information Theory*, vol. IT-32, pp. 697-700, September 1986.

[2] A. Fiat and A. Shamir, "Generalized 'write-once' memories," *IEEE Transactions on Information Theory*, vol. IT-30, pp. 470-480, May 1984.

[3] F. Fu and A. J. Han Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 308-313, 1999.

[4] F. Fu and R. W. Yeung, "On the capacity and error-correcting codes of write-efficient memories," *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2299-2314, 2000.

[5] S. Gregori, A. Cabrini, O. Khouri and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proceedings of The IEEE*, vol. 91, no. 4, April 2003.

[6] C. Heegard, "On the capacity of permanent memory," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 34-42, January 1985.

[7] A. Jiang, V. Bohossian and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," *Proc. IEEE International Symposium on Information Theory (ISIT)*, Nice, France, June 2007.

[8] A. V. Kuznetsov and A. J. H. Vinck, "On the general defective channel with informed encoder and capacities of some constrained memories," *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1866-1871, Nov. 1994.

[9] F. Merkx, "WOMcodes constructed with projective geometries," *Traitement du Signal*, vol. 1, no. 2-2, pp. 227-231, 1984.

[10] R. L. Rivest and A. Shamir, "How to reuse a 'write-once' memory," *Information and Control*, vol. 55, pp. 1-19, 1982.

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

[12] G. Zemor and G. D. Cohen, "Error-correcting WOM-codes," *IEEE Trans. Information Theory*, vol. 37, no. 3, pp. 730-734, 1991.