Trade-offs between Instantaneous and Total Capacity in Multi-Cell Flash Memories

Eyal En Gad
Electrical Engineering
California Institute of Technology
Pasadena, CA 91125, U.S.A.
eengad@caltech.edu

Anxiao (Andrew) Jiang
Computer Science and Engineering
Texas A&M University
College Station, TX 77843, U.S.A.
ajiang@cse.tamu.edu

Jehoshua Bruck
Electrical Engineering
California Institute of Technology
Pasadena, CA 91125, U.S.A.
bruck@caltech.edu

***Abstract*—The limited endurance of flash memories is a major design concern for enterprise storage systems. We propose a method to increase it by using relative (as opposed to fixed) cell levels and by representing the information with Write Asymmetric Memory (WAM) codes. Overall, our new method enables faster writes, improved reliability as well as improved endurance by allowing multiple writes between block erasures. We study the capacity of the new WAM codes with relative levels, where the information is represented by multiset permutations induced by the charge levels, and show that it achieves the capacity of any other WAM codes with the same number of writes. Specifically, we prove that it has the potential to double the total capacity of the memory. Since capacity can be achieved only with cells that have a large number of levels, we propose a new architecture that consists of multi-cells - each an aggregation of a number of floating gate transistors.**

I. INTRODUCTION

Flash memory is the most widely-used type of non-volatile electronic memory [1]. The amount of charge stored in a flash memory cell can be quantized into $q \geqslant 2$ discrete values in order to represent up to $\log_2 q$ bits. (The cell is called a single-level cell (SLC) if $q = 2$, and called a multi-level cell (MLC) if $q > 2$). We call the q states of a cell its levels: level 0, level 1, . . ., level $q - 1$. The charge is quantized into the discrete levels by an appropriate set of threshold levels. The level of a cell can be increased by injecting charge into the cell, and decreased by removing charge from the cell. Flash memories have the prominent property that although it is relatively easy to increase a cell's level, it is very costly to decrease it. This follows from the fact that flash-memory cells are organized as blocks, where every block has about $10^5 \sim 10^6$ cells. To decrease any cell's level, the whole block needs to be erased (which means to remove the charge from all the cells of the block) and then be reprogrammed. Block erasures not only are slow and energy consuming, but also significantly reduce the longevity of flash memories, because every block can endure only $10^4 \sim 10^5$ erasures with guaranteed quality [1]. Therefore, it is highly desirable to minimize the number of block erasures.

We can store $\log_2 q$ bits on a flash cell with q levels. That way, each time we want to update the data on the memory, we would have to erase the whole block. We call this representation method "the trivial scheme". We could also use a bit more sophisticated update schemes. For example, we could store only 1 bit in each cell, according to the *parity* of the level of the cell. If the cell is in level 3, for example, it stores the value 1. Using this scheme, we can update the data $q - 1$ times before a block erasure will be required. We call this scheme "the parity scheme". Update schemes like the parity scheme can be especially useful for enterprise applications of flash memory, where the endurance of the memory becomes a major design concern. Update schemes are also known as *write once memory (WOM)* codes for $q = 2$ [6], and *write asymmetric memory (WAM)* codes for $q > 2$ [3]. In this paper we focus on the WAM model. The capacity of WAM codes was studied at [4], but no capacity achieving constructions are known.

A main trade-off in the design of WAM codes is between the number of times we can update the memory and the amount of data the memory can store at a time. We call the amount of data stored at a time the *instantaneous* rate of the code. In general, the higher the instantaneous rate, the lower the number of times we can update the memory. In order to settle this trade-off, we focus on optimizing the *product* of these two values. In other words, we optimize the total amount of data the memory stores between two erasures. We call that number the *total* rate of the code. Once the total rate is optimized, we optimize the instantaneous rate under that constraint. Back to the previous examples, we remember that the trivial scheme has an instantaneous rate of $\log_2 q$ bits in each cell. Its total rate is the same, since we can only write the data once between block erasures. The parity scheme, however, has an instantaneous rate of only 1 bit per cell. Its total rate is $q - 1$ bits per cell, since it allows for $q - 1$ writes between block erasures. So the parity scheme is better according to our standards.

In MLC flash memory, the process of writing a specific level on a cell is designed to cautiously approach the target level from below so as to avoid undesired block erasures in case of overshoots. Consequently, these attempts require many programming cycles, and they work only up to a moderate number of levels per cell. In order to avoid that problem, it was suggested to represent the data by a set of n cells, according to the permutation induced by the relative charge levels of the individual cells [5]. When we inject charge into a cell, we only need to make its charge level higher than that of the previous cell in the permutation, and therefore there is no risk of overshooting. Another advantage of representing data by relative levels is that the threshold levels are no longer needed. This mitigates the effects of retention in the cells (slow charge leakage). That method was called rank modulation.

In this paper we extend this idea and suggest to use permutations of a given multiset. That is, we still use the relative levels of the cells instead of the absolutes level, but allow multiple cells to be in the same relative level. We use multisets with the same multiplicity for all the elements. That is, the number of cells in each relative level is equal. Using multiset permutations, we still benefit from all of the advantages of rank modulation. In addition, we gain more flexibility, and we show in this work that this flexibility could result in better performance.

While the values of the cells don't need to be quantized using thresholds, we still use discrete levels for the analysis. This is to allow easy and fair analysis, and because there should still be a certain charge difference between the cells in order to limit errors. When we use a discrete model, the problem of designing update schemes with relative levels become a special case of the WAM problem. Namely, we are interested in a class of WAM codes, where the data is represented only by the multiset permutation induced by the levels of the cells. We call this class of codes: rank modulation WAM (RMWAM) codes. We define the capacity of the model as the tightest upper bound on the amount of information that can be stored on the memory over multiple writing cycles, and study the capacity of rank modulation WAM model. We show that when q is large, it can achieve the capacity of the more general WAM model for the same number of writes. Specifically, it is possible to store almost 2 bits per cell at a given time, while reusing the memory close to q times. That is twice the amount of information that is stored with the "parity scheme" from the example above. One caveat for that results is that in practical flash memory devices, q is a moderately small number. In order to tackle this obstacle, we propose a method to achieve high values of q with the existing cell technology. The main idea is to combine several floating gate transistors into a virtual cell, which we call a multi-cell.

The rest of the paper is organized as following: In section II we present the notations and definitions. In section III we study the cost of updating the memory under the RMWAM model. In section IV we state and discuss the main result of the paper, the capacity of the model. Section V describes the proof of the capacity theorem, and finally, in section VI we present the proposed structure of multi-cell flash memory.

II. Definitions and notations

A write asymmetric memory (WAM) is a q-ary information storage medium consists of n cells. The q states of a cell are also called *levels*: from level 0 to level $q-1$. A cell can change from level i to level j if and only if $i < j$. The initial state of all the cells is 0. We want to reuse the WAM for T successive cycles. We only consider the following case: The encoder knows and the decoder does not know the previous state of the memory. The encoder and decoder can use arbitrary codes for every cycle, and there are no decoding errors (zero-error case). For the vectors

$$x^n = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_q^n$$

$$y^n = (y_1, y_2, \ldots, y_n) \in \mathbb{Z}_q^n$$

we denote $x^n \Rightarrow y^n$ if and only if $x_i \leqslant y_i$, $i = 1, 2, \ldots, n$.

Definition 1. *An $(n, T, M_1, \cdots, M_T)$ WAM code consists of T pairs of encoding and decoding functions$\{(f_t, g_t)\}_{t=1}^{T}$, where the message index sets $I_t = \{1, \cdots, M_t\}$, the encoding functions $f_t : I_t \times \mathbb{Z}_q^n \to \mathbb{Z}_q^n$, and the decoding function $g_t : \mathbb{Z}_q^n \to I_t$. These encoding and decoding functions satisfy: For any $m_1 \in I_1, m_2 \in I_2, \cdots, m_T \in I_T$, denote $y_0^n = (0, \cdots, 0) = \underline{0} \in \mathbb{Z}_q^n$ and $y_t^n = f_t(m_t, y_{t-1}^n), t = 1, \cdots, T$. Then, $y_{t-1}^n \Rightarrow y_t^n$ and $g_t(y_t^n) = m_t, t = 1, \cdots, T$.*

Denote

$$R_t = (1/n) \log_2 M_t, \qquad t = 1, \cdots, T.$$

We will use the binary logarithm in the rest of the paper. The T-tuple $(R_1, \cdots, R_T)$ is called the rate vector of this code. The closure of the set of all rate-vectors $\mathcal{A}_T$ is called the capacity region of the WAM. The maximum total number of information bits stored in one storage cell of the WAM during the T updating cycles is

$$C_T = \max \left\{ \sum_{t=1}^{T} R_t | (R_1, R_2, \cdots, R_T) \in \mathcal{A}_T \right\}.$$

Fu and Vinck [4] showed that $C_T = \log \binom{T+q-1}{q-1}$.

Since we want to use only the relative values of the cells, we use permutations of a multiset. We use a multiset of l elements (not including repetitions), where the multiplicity of each element is z. The cardinality of the multiset is thus $n = lz$. We denote the set of all permutations of a multiset of l elements with multiplicities z as $S_{l,z}$. Let $c = (c_1, c_2, \ldots, c_n)$, with $c_i \in \{0, 1, \ldots, q-1\}$ be the state of an array of n flash cells, each having q discrete levels. We further assume that the variables induce a multiset permutation $\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)] \in S_{l,z}$. The multiset permutation σ is uniquely defined by the constraint $c_i > c_j$ for all (i, j) s.t. $\sigma(i) > \sigma(j)$, where $l - 1 \geqslant \sigma(i) > \sigma(j) \geqslant 0$. We define σ^{-1} such that $\sigma^{-1}(i)$ is the set of all cells with relative level i.

Definition 2. *A (l, z, T, M)-Rank-Modulation Write-Asymmetric-Memory (RMWAM) code is a WAM code for which:*

1) *$(M_t, f_t, g_t, I_t) = (M, f, g, I)$ for all $t = 1, \cdots, T$.*
2) *$g : S_{l,z} \to I$ is based only according to the multiset permutation induced by the cell levels.*

Since $M_t = M$ for $t = 1, \cdots, T$, it follows that $R_t = R = (1/lz) \log M$ for $t = 1, \cdots, T$. Therefore we call R

the instantaneous rate of the code, and RT the total rate (also known as sum rate). In addition, we define $C = C_T/T$, and call C the instantaneous capacity of the RMWAM model, and C_T its total capacity.

III. Cost of Update

To change the multiset permutation from σ to σ', we program the cells based on their relative levels in σ', so that every cell's level increases as little as possible. Let $c' = (c'_1, c'_2, \ldots, c'_n)$ denote the new cell's levels to be set. First, for each $i \in \sigma'^{-1}(0)$, we set $c'_i = c_i$. Then, for $j = 1, 2, \cdots, l-1$, and for $i \in \sigma'^{-1}(j)$, we set

$$c'_i = \max\{c_i, \max_{k \in \sigma'^{-1}(j-1)} c'_k + 1\}.$$

Given two cell states c and c', let $\text{cost}(c \to c')$ denote the *cost* of changing the cell state from c to c'. We define the cost as the difference between the levels of the highest cell, before and after the update operation. Namely, if $\sigma(j) = \sigma'(i) = l-1$, $\text{cost}(c \to c') = c'_i - c_j$.

In order to calculate the cost, we need to simulate the update operation. We now present an equivalent definition of the cost, that can be calculated directly from the current multiset permutation and the multiset permutation to be written. The Lemma is a generalization of Theorem 1 from [2], and it further assumes that $c_i = \sigma(i)$ for $i = 1, \cdots, n$.

Lemma 1.

$$\textit{cost}(\sigma \to \sigma') = \max_{i=0,\cdots,l-1} (\sigma(i) - \sigma'(i))$$

In other words, the cost is the asymmetric infinity metric.

Proof:

Assume by induction on $\sigma'(i)$, that

$$c'_i = \max\{\sigma(i), \sigma'(i) + \max_{j \text{ s.t. } \sigma'(j)<\sigma'(i)} [\sigma(j) - \sigma'(j)]\}$$

In the base case, $\sigma'(i) = 0$, so there is no j s.t. $\sigma'(j) < \sigma'(i)$. Therefore, $c'_i = \sigma(i) = c_i$, as described in the programming process. For $\sigma'(i) > 0$,

$$c'_i = \max\{c_i, \max_{k \in \sigma'^{-1}(\sigma'(i)-1)} c'_k + 1\}$$

$$\begin{aligned}
&\max_{k \in \sigma'^{-1}(\sigma'(i)-1)} c'_k + 1 \\
&= \max_{k \in \sigma'^{-1}(\sigma'(i)-1)} \{\sigma(k), \\
&\qquad \sigma'(k) + \max_{j \text{ s.t. } \sigma'(j)<\sigma'(k)} [\sigma(j) - \sigma'(j)]\} + 1 \\
&= \sigma'(i) + \max_{k \in \sigma'^{-1}(\sigma'(i)-1)} \{\sigma(k) - \sigma'(k), \\
&\qquad \max_{j \text{ s.t. } \sigma'(j)<\sigma'(k)} [\sigma(j) - \sigma'(j)]\} \\
&= \sigma'(i) + \max_{j \text{ s.t. } \sigma'(j)<\sigma'(i)} [\sigma(j) - \sigma'(j)]
\end{aligned}$$

And the induction is proven. Now let $\sigma'(i) = l-1$:

$$\begin{aligned}
\text{cost}(c \to c') &= c'_i - c_j \\
&= \max\{\sigma(i), \\
&\qquad \sigma'(i) + \max_{j \text{ s.t. } \sigma'(j)<\sigma'(i)} [\sigma(j) - \sigma'(j)]\} - (l-1) \\
&= (l-1) - (l-1) + \max_{j \text{ s.t. } \sigma'(j)<\sigma'(i)} [\sigma(j) - \sigma'(j)] \\
&= \max_{i=0,\cdots,l-1} (\sigma(i) - \sigma'(i))
\end{aligned}$$

■

We now show an example of an update operation, and calculate the cost according to the two equivalent definitions:

Example 1. *Let* $(l,z) = (3,1)$, *and* $c = (0,1,2)$. *So* $\sigma = [0,1,2]$. *Now let* $\sigma' = [1,2,0]$. *We increase the levels of the cells to represent* σ'. *First we set* $c'_3 = c_3 = 2$. *Then we set* $c'_1 = \max\{c_1, c'_3 + 1\} = \max\{0,3\} = 3$. *Finally we set* $c'_2 = 4$. *The cost of the update is* $c'_2 - c_3 = 4 - 2 = 2$. *We can also calculate it directly from the multiset permutations:* $\sigma = [0,1,2]$, *and* $\sigma' = [1,2,0]$. *So* $\sigma - \sigma' = [-1,-1,2]$, *and the maximum is* 2, *so this is the cost.*

Finally, for a fixed $\sigma \in S_{l,z}$, set

$$B_{l,z,r}(\sigma) = \{\sigma' \in S_{l,z} | \text{cost}(\sigma \to \sigma') \leqslant r\}.$$

We note that $|B_{l,z,r}(\sigma)|$ is independent of σ.

IV. Capacity

In the following we present an expression for the capacity of the RMWAM model.

Theorem 1. C_T *is maximal when* $T = q - l + 1$, *and in this case:*

$$C = \frac{l-1}{l} \times \frac{\log\binom{2z}{z}}{z}$$

The proof of Theorem 1 will be given in section V.

As a corollary of the theorem, we look at three different cases:

1) The case of $(l,z) = (2,\infty)$. Here $C = \lim_{z\to\infty}(1/2) \times \frac{\log\binom{2z}{z}}{z} = 1$. Therefore we can store up to 1 bit in each cell in each updating cycle. In fact, in this case, a trivial code that assign a different message index to each multiset permutation achieves the capacity.
2) The case of $(l,z) = (\infty,1)$. Here $C = \lim_{l\to\infty} \frac{l-1}{l} \times \frac{\log\binom{2}{1}}{1} = 1$. Therefore in this case we can also store up to 1 bit in each cell in each updating cycle. However, here it is not easy to design a code that archives the capacity, and that problem is still open.
3) For $(l,z) = (\infty,\infty)$. Here $C = \lim_{l,z\to\infty} \frac{l-1}{l} \times \frac{\log\binom{2z}{z}}{z} = 1 \times 2 = 2$. So we can store up to 2 bits per cell in each updating cycle in this case. So $C_T = 2(q-l+1)$, and in the case of $q \gg l$, $C_T \to 2q$. We notice that in that case the total capacity of the WAM model is the same [4]. That is since for WAM, $C_T = \log\binom{T+q-1}{q-1} = \log\binom{2q-l-1}{q-1} \to 2q$, $q/l \to \infty$. So in this case, the total capacity of the RMWAM model is the same as that of the WAM model.

V. Proof of theorem 1

A. Converse Part

If we want to guarantee that the cost of each update operation is no more than r, we must set $M \leqslant |B_{l,z,r}(\sigma)|$. Otherwise, if we would like to write the message m, we cannot guarantee that there is a multiset permutation in $B_{l,z,r}(\sigma)$ that represents m. We let $K_r = \frac{1}{lm}\log|B_{l,z,r}(\sigma)|$. By setting $R \leqslant K_r$, we cannot guarantee to write more than $(q-l+1)/r$ times, so RT is at most $(q-l+1)K_r/r$. In Lemma 2 we calculate $B_{l,z,r}(\sigma)$ in order to achieve an explicit bound on the rate. The lemma is a generalization of Theorem 2 from [2] for general z.

Lemma 2.

$$|B_{l,z,r}(\sigma)| = \frac{[(r+1)z]!}{(z!)^{r+1}}\binom{(r+1)z}{z}^{l-(r+1)}$$

Proof: We use induction on l. When $l = r+1$, the statement is trivial, and $|B_{l,z,r}(\sigma)| = (lz)!/(z!)^l$. Now we assume that the statement is true for $l \leqslant l_0$, and consider $l = l_0+1$ and $l > r+1$. Let $\sigma' \in B_{l,m,r}(\sigma)$. According to Lemma 1, for any $i \in \sigma'^{-1}(0)$, $\sigma(i) \in \{0,1,\cdots,r\}$. Therefore, there are $((r+1)z)!/(z!(rz)!)$ choices to form $\sigma'^{-1}(0)$. Let $\hat\sigma \in S_{l-1,z}$ be obtained from σ by setting $\hat\sigma(i) = \sigma(i)-1$ for all i s.t. $\sigma(i) \geqslant r+1$ and fill the lower levels of $\hat\sigma$ arbitrarily with the remaining cells. In addition, let $\hat\sigma'(i) = \sigma'(i)-1$ for all $i \notin \sigma'^{-1}(0)$. Given a certain choice of $\sigma'^{-1}(0)$, there is a one-to-one mapping between updating the levels of the remaining $z(l-1)$ cells from σ to σ' and updating the levels of those $z(l-1)$ cells from $\hat\sigma$ into $\hat{\sigma'}$. Therefore we have $\text{cost}(\hat\sigma \to \hat{\sigma'}) = \text{cost}(\sigma \to \sigma') \leqslant r$. So we get $|B_{l,z,r}(\sigma)| = ((r+1)z)!/(z!(rz)!)|B_{l-1,z,r}(\sigma)| = [(r+1)z]!/(z!)^{r+1}\binom{(r+1)z}{z}^{l-(r+1)}$. ■

Since $RT \leqslant (q-l+1)K_r/r$, we are now interested in the behavior of K_r/r. In Lemma 3 we show that K_r/r is decreasing in r, and thus $RT \leqslant (q-l+1)K_1$.

Lemma 3. *K_r/r is decreasing in r for $r \geqslant 1$.*

Proof: It is enough to show that for $r \geqslant 2$,

$$\frac{\log|B_{l,z,r-1}(u)|}{r-1} - \frac{\log|B_{l,z,r}(u)|}{r} \geqslant 0$$

$$\begin{aligned}
&r\log|B_{l,m,r-1}(u)| - (r-1)\log|B_{l,m,r}(u)|\\
&= r\log\left(\frac{(zr)!}{(z!)^r}\binom{zr}{z}^{l-r}\right)\\
&\quad -(r-1)\log\left(\frac{(z(r+1))!}{(z!)^(r+1)}\binom{z(r+1)}{z}^{l-(r+1)}\right)\\
&= r\log(rz)! - r^2\log z! + rl\log\binom{rz}{z} - r^2\log\binom{rz}{z}\\
&\quad -(r-1)\log((r+1)z)! + (r^2-1)\log z!\\
&\quad -l(r-1)\log\binom{(r+1)z}{z} + (r^2-1)\log\binom{(r+1)z}{z}\\
&= r^2\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}} - \log\binom{(r+1)z}{z} - r\log\frac{((r+1)z)!}{(rz)!}\\
&\quad +\log((r+1)z)! - lr\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}} + l\log\binom{(r+1)z}{z}\\
&\quad -\log z! + r\log z! - r\log z!\\
&= (l-r)\left(\log\binom{(r+1)z}{z} - r\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}}\right) + \log\frac{(rz)!}{(z!)^r}\\
&> (l-r)\left(\log\binom{(r+1)z}{z} - r\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}}\right)\\
&\geqslant \log\binom{(r+1)z}{z} - r\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}}
\end{aligned}$$

In the case of $z=1$ we use the inequality $(1+(1/x))^x < e$:

$$\log(r+1) - \log(1+1/r)^r \geqslant \log 3 - \log e > 0.$$

In the case of $z \geqslant 2$ we use the inequality $\binom{l}{\lambda l} \geqslant \frac{1}{\sqrt{8l\lambda(1-\lambda)}}2^{lH(\lambda)}$:

$$\begin{aligned}
&\log\binom{(r+1)z}{z} - r\log\frac{\binom{(r+1)z}{z}}{\binom{rz}{z}}\\
&\geqslant \log\binom{(r+1)z}{z} - \sum_{i=1}^{z} r\log\frac{zr+i}{zr-z+i}\\
&\geqslant z\log(r+1) + zr\log\frac{r+1}{r} - rz\log\frac{r}{r-1}\\
&\quad -(1/2)\log\left(8z\frac{r}{r+1}\right)\\
&= z\log(r+1) - zr\log\frac{r^2}{r^2-1} - (1/2)\log\left(8z\frac{r}{r+1}\right)\\
&= z\log(r+1) - z\frac{r}{r^2-1}\log\left(1+\frac{1}{r^2-1}\right)^{r^2-1}\\
&\quad -(1/2)\log\left(8z\frac{r}{r+1}\right)\\
&> z - (1/2)\log\left(8z\frac{r}{r+1}\right) > (1/2)\log\frac{2^{2z}}{8z} \geqslant 0
\end{aligned}$$

■

When $r=1$, $|B_{l,z,1}(u)| = \binom{2z}{z}^{l-1}$. Therefore, here:

$$\begin{aligned}
R &\leqslant (1/lz)\log|B_{l,z,1}(\sigma)| = (1/lz)\log\binom{2z}{z}^{l-1}\\
&= \frac{l-1}{l}\times\frac{\log\binom{2z}{z}}{z}
\end{aligned}$$

B. Direct part

We now show there exists a code whose rate approaches the upper bound.

Proof: Let $M = |B_{l,z,1}(\sigma)|/(lz)^{1+\epsilon}$, where ϵ is a positive constant. In the following we show that there exists a $(l, z, q-$

$l+1, M)$-RMWAM code. We first calculate R:

$$\begin{aligned} R &= (1/lz)\log M \\ &= (1/lz)\log|B_{l,z,1}(\sigma)| - (1/lz)(1+\epsilon)\log(lz) \\ &\to \frac{l-1}{l} \times \frac{\log\binom{2z}{z}}{z}, lz \to \infty \end{aligned}$$

So the instantaneous rate of such a code is asymptotically optimal. If we show that the cost is always 1, it follows that the total rate is also asymptotically optimal.

Suppose $\{F_m\}_{m=1}^M$ is a partition of $S_{l,z}$, i.e., $F_m \cap F'_m = \emptyset$, $m \neq m'$; and $\cup_{m=1}^M F_m = S_{l,z}$. We now show that there exists a partition of $S_{l,z}$, such that for any $\sigma \in S_{l,z}$ and any $m \in M$, there exists a vector $\sigma' \in F_m$, such that $\mathrm{cost}(\sigma \to \sigma') = 1$. We use a random coding method. With every $\sigma \in S_{l,z}$, we connect a random index r_σ which is uniformly distributed over the data set $I = \{1, \cdots, M\}$, and all these random indices are independent. Define

$$F_m = \{\sigma \in S_{l,z} | r_\sigma = m\}, \qquad m = 1, 2, \cdots, M.$$

Then $\{F_m\}_{m=1}^M$ forms a random partition of $S_{l,z}$. Fix $m \in M$ and $\sigma \in S_{l,z}$, then

$$\begin{aligned} &Pr\{F_m \cap B_{l,z,1}(\sigma) = \emptyset\} = Pr\{\forall \sigma \in B_{l,z,1}(\sigma), r_b \neq m\} \\ &\quad = [1 - 1/M]^{|B_{l,z,1}(\sigma)|} \leqslant \exp\{-|B_{l,z,1}(\sigma)|/M\} \\ &\quad = \exp\{-(lz)^{1+\epsilon}\} \end{aligned}$$

Therefore,

$$\begin{aligned} &Pr\{\exists m \in M \text{ and } \sigma \in S_{l,z}, \text{ s.t. } F_m \cap B_{l,z,1}(\sigma) = \emptyset\} \\ &\quad \leqslant M|S_{l,z}| \exp\{-(lz)^{1+\epsilon}\} \\ &\quad \leqslant 2^{2lz}(lz)! \exp\{-(lz)^{1+\epsilon}\} \\ &\quad \leqslant \exp\{lz(2 + \ln(lz) - (lz)^\epsilon)\} \to 0 \quad, lz \to \infty \end{aligned}$$

This implies that when $n = lz$ is sufficiently large, there exists a partition of $S_{l,z}$ such that the cost of each update is 1. ■

VI. Multi-cell Flash Memory

NAND flash memory is the most widely used type for general storage purpose. In NAND flash, several floating gate transistors are connected in series, where we read or write only one of them at a time. We propose to replace each transistor with a multi-cell of m transistors connected in parallel, and to connect their control gates, as shown in Figure 1. That way, their current sums together in read operations, and the read precision increases by m times, allowing to store mq levels in a single multi-cell. In write operations, we write the same value to all the transistors, such that the sum of their charge levels gives us the desired total level. We suspect that the error rate would be similar to that of a traditional flash cell.

If we store data on n transistors that form n/m multi-cells of mq levels without a WAM code, we would get a rate of $R = RT = (n/m)\log_2(mq)$. This is less than the $n\log_2 q$ we would get using traditional cells. However, if we use RMWAM codes, we could get a total capacity approaching $2nq$ both with multi-cells and with traditional cells. In order to approach a total capacity of $2nq$ with RMWAM, the number of updates the code can take must be much greater than the number of relative levels we use. By using multi-cells, we increase T at the expense of the R, and thus approach C_T faster.

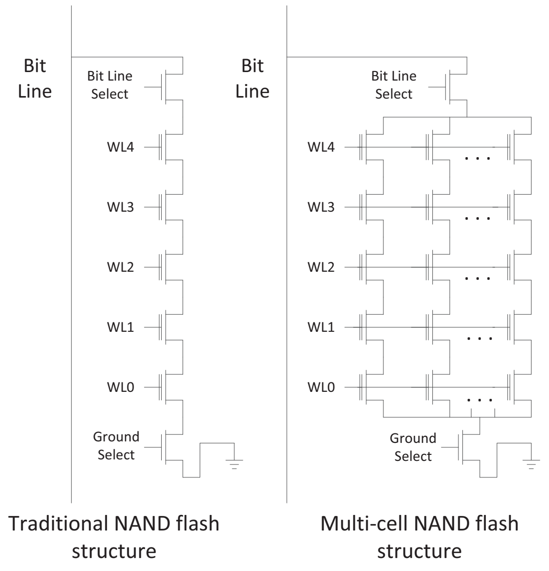

Figure 1. The proposed architectural change in NAND flash

VII. Conclusions

In this paper, we studied the capacity of rank-modulation write asymmetric memory codes. A class of WAM codes, RMWAM codes can allow faster update and better protection against errors in flash memories, since they don't require discrete threshold levels. We showed that the capacity of RMWAM codes approaches the capacity of WAM codes. In addition, we presented a new flash cell structure (multi-cell) that can increase the number of levels in the cells.

VIII. Acknowledgments

This work was partially supported by an NSF grant ECCS-0801795 and a BSF grant 2010075. The author would like to acknowledge that Qing Li from Texas A&M University derived Lemmas 1 and 2 independently.

References

[1] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash Memories*. Kluwer Academic Publishers, 1999.

[2] E. En Gad, A. Jiang, and J. Bruck, "Compressed encoding for rank modulation," in *Proceedings of the 2011 IEEE Int. Symp. on Inform. Theory, ISIT2011, St. Petersburg, Russia*, Aug. 2011, pp. 884–888.

[3] A. Fiat and A. Shamir, "Generalized "write-once" memories," *IEEE Trans. on Inform. Theory*, vol. IT-30, no. 3, pp. 470–480, May 1984.

[4] F.-W. 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 Trans. on Inform. Theory*, vol. 45, no. 1, pp. 308–313, Jan. 1999.

[5] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. on Inform. Theory*, vol. 55, no. 6, pp. 2659–2673, Jun. 2009.

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