Construction of Random Input-Output Codes with Moderate Block Lengths

Ravi Motwani
Non-Volatile Systems Group
Intel Corporation
Santa Clara, California, USA
Email: *ravi.h.motwani@intel.com*

Eitan Yaakobi
Department of Computer Science
Technion-Israel Institute of Technology
Haifa 32000, Israel
Email: *yaakobi@cs.technion.ac.il*

***Abstract*—*Random I/O (RIO) Codes*, recently introduced by Sharon and Alrod, is a coding scheme to improve the random input/output performance of flash memories. Multi-level flash memories require, on the average, more than a single read threshold in order to read a single logical *page*. This number is important to be optimized since it sets the read latency of flash memories. An (n, M, t) RIO code assumes that t pages are stored in n cells with $t + 1$ levels. The first page is read by applying a read threshold between levels t and $t + 1$. Similarly, the second page is read by applying a read threshold between levels $t - 1$ and t, and so on. The read binary vectors for consecutive pages satisfy the property that the set of positions read with value 1 can only increase. Therefore, Sharon and Alrod showed also that the design of RIO codes is equivalent to the design of *WOM codes*. The latter family of codes attracted a lot of attention in recent years in order to improve the lifetime of flash memories by allowing to write multiple messages to the memory without the need for a physical erase.**

In this paper we notice two important distinctions between RIO codes and WOM codes. While in WOM codes the messages are received one after the other and thus are not known all in advance, in RIO codes the information of all logical pages can be known in advance when programming the cells. Even though this knowledge does not improve the capacity of RIO codes, it allows the design of efficient high-rate codes with a moderate block length, which are hard to be found for WOM codes. We also study another family of RIO codes, called here *partial RIO Codes*, that allow to find even more efficient codes in the tradeoff of reading more than a single threshold to read a page.

I. Introduction

Flash memories are the prevalent type of non-volatile memory (NVM) in use today. They are employed in a wide range of applications such as mobile, embedded, and enterprise, and their dominance in these applications continues to grow at a staggering pace. Compared to other storage memories, one of the outstanding advantages of flash memories is their significantly high random read performance, which places them as an ideal solution for high throughput applications.

Flash memories are comprised of block of cells. These cells can have binary values, i.e. they store a single bit, or can have multiple levels and thus can store multiple bits in a cell. For example, in MLC flash two bits are stored in a cell and TLC flash supports three bits in a cell.The storage of more than a single bit in a cell is accomplished by supporting multiple charge levels in the cell, which are then distinguished by different read thresholds. Assume the flash memory cells have 2^m different charge levels and thus can store m bits per cell. If all these bits belong to the same logical information unit (also called *page*), then, in order to read these bits, $2^m - 1$ read thresholds had to be applied which will significantly slow down the reading speed of the memory. The solution to this obstacle is to distribute bits sharing the same group of cells among different logical pages such that when a page is read only a single bit is read from each cell. Then, the number of read thresholds depends on the bit which is read from the cells and the average number of read thresholds determines the memory read speed. For example, consider the three bits stored in a TLC flash memory cell as shown in Figure 1. Then, a group of multiple TLC flash cells stores three logical pages, called page 0, page 1, and page 2. In order to read page 0, one read threshold is required; to read page 1, two read thresholds are required; and finally to read page 2, four read thresholds are required. Hence, on the average 2.33 reads are required to read one page of data. Usually, each read threshold operation consumes 50 μsec and therefore a large number of average reads reflects directly the read performance of flash memories. This may prove to be a crucial bottleneck to the usage of multi-level flash memories in SSDs.

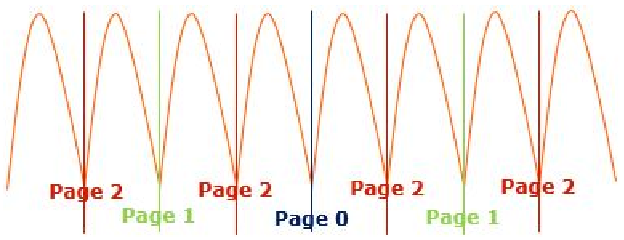

Figure 1. The cell voltage distribution of an 8-level Flash memory.

The solution to this problem was recently proposed in [9], where *Random I/O (RIO) Codes* were proposed in order to permit reading one page of data from multi-level flash memories using a single read threshold. They showed that a one-to-one correspondence exists between RIO codes and the well studied *WOM codes* [7]. Namely, a WOM code which permits to write k bits into n cells in t consecutive writes without erasing also gives a RIO code which permits to write t pages of k bits per page into n cells with $t + 1$ levels, while ensuring that each page is read with a single read threshold operation. For example, the well-known WOM code which stores 2 bits twice into 3 binary cells without erasure permits to write 2 pages with 2 bits each into 3 ternary flash memory cells; See Example 1 in Section II.

This paper takes advantage of an important property which can hold for RIO codes but not for WOM codes. In WOM codes, the messages are stored sequentially and are not all known in advance while encoding. Therefore, on each write, the encoder sets deterministically the cell state values as a function of the current memory state and the received message. However, in RIO codes, all the messages can be known in advance and thus the encoding of a particular page can be a function of the following pages as well. A code which leverages information of all messages will be called a *parallel RIO code* since the information of all t messages is known in advance and the encoding can be done in parallel. From the information theory point of view, parallel RIO codes, RIO codes, and WOM codes all have the same upper bound on their sum-rate, which is the total number of bits that is possible to write to the memory, normalized by the number of cells. However, the design of parallel RIO codes can be significantly simpler due to this property and thus it is possible to find codes with parameters for which WOM codes do not exist.

RIO codes can have another important property that can help in their design. Assume it is allowed to read more than a single read threshold, for example two thresholds, then we show that this model is equivalent to the case where both the encoder and decoder of a WOM code are informed with the previous memory state. In general, there can be four possible models depending whether the encoder and decoder are or aren't informed with the previous state of the memory; see [12] for a rigorous study of all four models. The traditional setting of WOM codes assumes the common and practical model, where the the encoder is informed while the decoder is not. However, this setup of RIO codes provides a practical example where both the encoder and decoder are informed. This knowledge allows to construct even more codes with parameters which did not exist in the codes we mentioned so far.

The rest of this paper is organized as follows. In Section II, we formally define WOM codes, RIO codes, and parallel RIO codes and state the connection established between them in [9]. In Section III, we show parameters for which it is possible to construct parallel RIO codes but not WOM codes or RIO codes and state a condition for the existence of parallel RIO codes. This condition leads in Section IV to an algorithm to construct parallel RIO codes. In Section V, we define partial RIO codes and study more code constructions for this family of RIO codes. Some proofs are omitted due to the lack of space.

II. Definitions and Basic Properties

In this section, we formally define the codes we study in the paper. We assume that the cells can have two or more (q) levels and we denote by $[q]$ the set $\{0, 1, \ldots, q-1\}$. Initially, all cells are in level zero and it is only possible to increase the level of each cell. For n cells, 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 $1 \leqslant i \leqslant n$. We will first review the definition of WOM codes. In general, there are two cases of WOM codes, known as *fixed-rate WOM codes*, where the number of messages on each write is the same, or *unrestricted-rate WOM codes*, where this constraint is not invoked [14]. In this work, we care about the implementation point of view of these codes and thus consider only the fixed-rate case for binary cells. However, we note that all of these results can be extended for the unrestricted-rate case as well.

Definition. *An $[n, M, t]$ WOM code is a coding scheme comprising of n binary 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*

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

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

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

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

In [5], [6], the capacity region of a binary t-write WOM was found and $\log(t+1)$ was proved to be the maximum sum-rate. It was also shown that all rate-tuples in the capacity region are achievable. Since then several code constructions were given with the intention of reaching high sum-rate, e.g. [**?**], [2], [3], [8]. Recently, some capacity achieving constructions were given both for two writes [1], [11], [13], [14] and multiple writes [1], [10] as well as for error-correction [4]. However, they all suffer either a large block length and encoding/decoding complexities or do not work in the worst case. Thus, the problem of finding efficient WOM code constructions with moderate block length still remains an important challenge.

Before we formally define RIO codes let us introduce the operation of reading a single threshold from a group of cells. Let $\boldsymbol{c} \in [q]^n$ be a cell-state vector and r be a threshold level, $1 \leqslant r \leqslant q-1$, between levels $r-1$ and r. The operation of reading the r-th threshold from the cell-state vector $\boldsymbol{c}$ is denoted by $RT_r(\boldsymbol{c})$ and returns a binary vector such that $RT_r(\boldsymbol{c})_i = 1$ if and only if $c_i \geqslant r$. The next claim is an immediate property of the definition of this operation.

Proposition 1 *For all $1 \leqslant r < s \leqslant q-1$, $RT_r(\boldsymbol{c}) \geqslant RT_s(\boldsymbol{c})$.*

The idea behind RIO codes is to use $(t+1)$-ary cells in order to store t messages. Assume $\boldsymbol{c}$ is the cell-state vector. The information of the first page is stored in the vector which is generated from the t-th threshold, i.e. $RT_t(\boldsymbol{c})$; the information of the second page is stored in the vector which is generated from the $(t-1)$-th threshold, i.e. $RT_{t-1}(\boldsymbol{c})$; and so on, the t-th page is generated from the vector which is generated from the first threshold, i.e. $RT_1(\boldsymbol{c})$. In order to program the t messages, we start with the first message. Since its information is carried by the vector $RT_t(\boldsymbol{c})$ we only need to use levels 0 or t. Then, when the second page is programmed, only the cells in level zero can be programmed and since its information is carried by the vector $RT_{t-1}(\boldsymbol{c})$, we only need to use levels 0 or $t-1$ over the non-programmed cells. This process repeats itself until we reach the t-th message and then only levels 0 and 1 are used over the non-programmed cells at that stage. Note that according to this process, in order to program the i-th page $2 \leqslant i \leqslant t$, the encoder only needs to read one threshold and thus receive the binary vector $RT_{t+2-i}(\boldsymbol{c})$. One property which follows from Proposition 1 is that $RT_t(\boldsymbol{c}) \leqslant RT_{t-1}(\boldsymbol{c}) \leqslant \cdots \leqslant RT_1(\boldsymbol{c})$ and thus the binary vectors representing the t messages satisfy a similar constraint to the cell-state vectors on consecutive writes of a WOM code. This connection between WOM codes and RIO codes was established by Sharon and Alrod in [9].

As opposed to WOM codes where on each write only the current message is known, in RIO codes it may happen that all t messages are available to the encoder in advance at the time of programming. This knowledge can simplify the construction of such codes. We will distinguish between these two families of RIO codes and show how this knowledge can provide the construction of RIO codes that do exist if this information is not known.

Definition. *An (n, M, t) RIO code is a coding scheme comprising of n $(t+1)$-ary cells, t messages, and t encoding and decoding maps $(\mathcal{E}_i, \mathcal{D}_i)$, for $1 \leqslant i \leqslant t$, defined as follows:*

$$\mathcal{E}_i : [M] \times [2]^n \to [t+1]^n,$$

such that for all $(m, \boldsymbol{c}) \in [M] \times [t+1]^n$ if $c_j > 0$ then $\mathcal{E}_i(m, RT_{t+2-i}(\boldsymbol{c}))_j = c_j$. Similarly, the decoding map $\mathcal{D}_i$ is defined by

$$\mathcal{D}_i : [2]^n \to [M],$$

such that for all $(m, \boldsymbol{c}) \in [M] \times [t+1]^n$, $\mathcal{D}_i(RT_{t+1-i}(\mathcal{E}_i(m, RT_{t+2-i}(\boldsymbol{c})))) = m$.

An (n, M, t) parallel RIO code is a RIO code with an encoding map

$$\mathcal{E} : [M]^t \to [t+1]^n,$$

and t decoding maps $\mathcal{D}_i : [2]^n \to [M]$, $1 \leqslant i \leqslant t$, such that for all $\boldsymbol{m} = (m_1, \ldots, m_t) \in [M]^t$, $\mathcal{D}_i(RT_{t+1-i}(\mathcal{E}(\boldsymbol{m}))) = m_i$.

TABLE I. A [3,4,2] WOM CODE

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

TABLE II. A (3,4,2) RIO CODE

Second set of information bits	First set of information bits			
	00	01	10	11
00	000	112	121	211
01	110	002	120	210
10	101	102	020	201
11	011	012	021	200

The following theorem was proved in [9] but is stated here for the completeness of the results.

Theorem 2. [9] *There exists an* $[n, M, t]$ *WOM code if and only if there exists an* (n, M, t) *RIO code.*

Next, we show an example for the construction of RIO codes from WOM codes.

Example 1. In this example we first review the famous $[3, 4, 2]$ WOM codes given by Rivest and Shamir [7]. This code is described in Table I. The construction of RIO code with the same parameters $(3, 4, 2)$ is described in Table II. For example, assume that the information bits of the first page are $(1, 1)$ then the first encoder programs the cells to state $(2, 0, 0)$. Then, if the information bits of the second page are $(1, 0)$, then the second encoder programs the cells to state $(2, 0, 1)$. Note that the binary vector that the first decoder extracts from the cells is $(1, 0, 0)$ since the threshold is between levels 1 and 2 and thus the two bits are decoded to $(1, 1)$. In the same manner, the binary vector that the second decoder extracts from the cells is $(1, 0, 1)$ since the the threshold is between levels 0 and 1.

Similarly to WOM codes, an upper bound on the sum-rate of the two families of RIO codes is $\log(t + 1)$.

Theorem 3. $\log(t + 1)$ *is an upper bound on the sum-rate of RIO codes and parallel RIO codes with t messages.*

The existence of RIO codes or WOM codes necessarily guarantees the existence of parallel RIO codes with the same parameters. However, the converse does not necessarily hold. In general, in the design of a WOM code or RIO code, when encoding the i-th message the output to be written into the cells does not depend on the following messages to come. However, for parallel RIO codes, it is possible to program a different cell-state vector for a given message based on the following messages that will be written to the memory. The information of all the messages in advance cannot improve the capacity results, however it can simplify the constructions of codes with high sum-rate and moderate block lengths.

III. WOM CODES AND RIO CODES

We start this section by showing an example of parameters where WOM codes and RIO codes do not exist, however parallel RIO codes do exist. Assume one wishes to construct a $(4, 7, 2)$ RIO code. Such a code does not exist, which will be proved next.

Lemma 4. *There does not exist a* $(4, 7, 2)$ *RIO code.*

Proof: Since RIO codes and WOM codes are equivalent, we will simply show that a $[4, 7, 2]$ WOM code does not exist. Assume in the contrary that such a code exist. Let $\mathcal{E}_1$ be its encoding map. Let us denote by $a = \max_{m \in [7]}\{w_H(\mathcal{E}_1(m))\}$, that is, a is the maximum Hamming weight of all possible cell-state vectors after the first write. We claim that $a \geqslant 2$. Otherwise, since the number of length-4 binary vectors of weight at most 1 is 5 and thus it is not possible to encode 7 different messages. If $\boldsymbol{c}$ is an achievable cell-state vector of weight two (or more) then on the next write it will be possible to encode at most 4 messages since the number of zeros in $\boldsymbol{c}$ is at most 2. Therefore, we conclude that there does not exist a $[4, 7, 2]$ WOM code or $(4, 7, 2)$ RIO code. ■

TABLE III. A (4,7,2) PARALLEL RIO CODE

	0	1	2	3	4	5	6
0	0000	1112	1121	1211	2111	1122	2211
1	0001	0002	1120	1210	2110	2120	2210
2	0010	1102	0020	1201	2101	2202	2201
3	0100	1012	1021	0200	2011	1022	2012
4	1000	0112	0121	0211	2000	0122	0222
5	0011	0012	0021	1200	2100	0022	2200
6	1010	0102	1020	0201	2010	2020	0202

Next we seek to show the existence of a $(4, 7, 2)$ parallel RIO code. The construction of such a code is described by in Table III which describes the encoding and decoding maps. The columns correspond to the symbol value of the first page and the rows correspond to the symbol value of the second page. We note again that the encoder knows at the time of encoding the two messages. Thus, for example the binary vectors which can be extracted from the cells in case the first page stores the symbol 5 are $0011, 1010$, or 1101, and if the symbol 6 is stored then these binary vectors can be $1100, 1001, 0111$, or 0101. In any event the different set of binary vectors corresponding to different symbols are mutually disjoint. We will show that this construction is optimal by proving that a $(4, 8, 2)$ parallel RIO code does not exist.

Lemma 5. *There does not exist a* $(4, 8, 2)$ *parallel RIO code.*

Proof: Assume in the contrary that such a code exists. For every message $m \in [8]$ let A_m be the set of all binary states it can achieve after the first write. There exists at least three sets which do not have any vectors of weight one. Since there are six vectors of weight two, there exist at least one set which have at most two vectors of weight two. However, the set of achievable vectors in that case is at most seven which gets a contradiction to writing 8 more different messages on the second write. ■

We conclude with the following condition on the existence of parallel RIO codes.

Theorem 6. *An* $(n, M, 2)$ *parallel RIO exists if and only if it is possible to divide the* 2^n *binary vectors into two sets of non-empty subsets* $\mathcal{A} = \{A_1, \ldots, A_M\}$ *and* $\mathcal{B} = \{B_1, \ldots, B_M\}$ *which satisfy the following conditions:*

1) *For all* $1 \leqslant i < j \leqslant M$, $A_i \cap A_j = \emptyset$ *and* $B_i \cap B_j = \emptyset$,
2) *For all* $1 \leqslant i, j \leqslant M$, *there exists* $\boldsymbol{a}_i \in A_i$ *and* $\boldsymbol{b}_j \in B_j$ *such that* $\boldsymbol{a}_i \leqslant \boldsymbol{b}_j$.

IV. CONSTRUCTION OF PARALLEL RIO CODES

In this section, an algorithm to construct an (n, M, t) parallel RIO code is outlined. The upper bound on M is $2^{\frac{n \log_2(t+1)}{t}}$. Consider reading message i by applying a threshold between levels $t - i$ and $t + 1 - i$, $1 \leqslant i \leqslant t$. Let $\boldsymbol{c} \in [t + 1]^n$ be the cell-state vector and $RT_{t+1-i}(\boldsymbol{c}) \in [2]^n$ is the message i read vector. Let $\mathcal{I}(\boldsymbol{r})$ denote the function which converts binary n-tuple $\boldsymbol{r}$ to its integer representation. Let $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})} \subseteq [t + 1]^n$ be the subset of cell-state vectors $\boldsymbol{c}$ such that $RT_{t+1-i}(\boldsymbol{c}) = \boldsymbol{r}$. The sum of the cardinality of the sets $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$, over all $\boldsymbol{r} \in [2]^n$ is $(t+1)^n,\ 1 \leqslant i \leqslant t$.

A hypercube of dimension t with length M per dimension consists of M^t entries with each entry $\boldsymbol{c} \in [t+1]^n$. The parallel RIO code is represented by this hypercube with each entry of the hypercube representing the cell-state vector for that t-message vector. For $1 \leqslant i \leqslant t, m \in [M]$, a hyperplane $\mathcal{H}^i_m$ of this hypercube with M^{t-1} entries corresponds to message i taking the value $m \in M$. The M^{t-1} entries of $\mathcal{H}^i_m$ will have all other messages $j, 1 \leqslant j \leqslant t, j \neq i$ span all possible values in $[M]$. The idea of parallel RIO code is to be able to decode the i-th message by applying a threshold between levels $t-i$ and $t+1-i$ for any entry in the hyperplane $\mathcal{H}^i_m$. The parallel RIO code constraint can be mathematically expressed as $RT_{t+1-i}(\mathcal{H}^i_m) \cap RT_{t+1-i}(\mathcal{H}^i_{m'}) = \emptyset,\ \forall m, m' \in M,\ m \neq m',\ 1 \leqslant i \leqslant t$.
This constraint is equivalent to the condition that elements from $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$ can be used to populate a hyperplane $\mathcal{H}^i_m$ for only one possible $m \in M$ and no other hyperplane $\mathcal{H}^i_{m'}, m' \neq m$ for all $1 \leqslant i \leqslant t$. The cardinality of $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$ plays an important role in the code construction. It can be shown that for $1 \leqslant i \leqslant t$, $\boldsymbol{r} \in [2]^n$, $|\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}| = i^j(t+1-i)^{n-j}$, where $j = w_H(\boldsymbol{r})$.

Define $\mathcal{B}_i = \{\mathcal{A}^i_j, 0 \leqslant j < 2^n\}, 1 \leqslant i \leqslant t$. For each i, a set of M disjoint subsets $\mathcal{C}^i_k \subset \mathcal{B}_i, 1 \leqslant k \leqslant M, \mathcal{C}^i_k = \bigcup_{m=1}^{j_k} \mathcal{A}^i_{k_m}$ is defined as permissible if $\sum_{m=1}^{j_k} |\mathcal{A}^i_{k_m}| \geqslant M^{t-1}$ and $\sum_{k=1}^{M}((\sum_{m=1}^{j_k} |\mathcal{A}^i_{k_m}|) - M^{t-1}) \leqslant (t+1)^n - M^t$.

For example, for the (4,7,2) parallel RIO code of Table III, the sets $\mathcal{C}^1_1 = \mathcal{A}^1_0, \mathcal{C}^1_2 = \mathcal{A}^1_1, \mathcal{C}^1_3 = \mathcal{A}^1_2, \mathcal{C}^1_4 = \mathcal{A}^1_4, \mathcal{C}^1_5 = \mathcal{A}^1_8, \mathcal{C}^1_6 = \mathcal{A}^1_3 \cup \mathcal{A}^1_{10} \cup \mathcal{A}^1_{13}, \mathcal{C}^1_7 = \mathcal{A}^1_5 \cup \mathcal{A}^1_7 \cup \mathcal{A}^1_9 \cup \mathcal{A}^1_{12}$, and $\mathcal{C}^2_1 = \mathcal{A}^2_0 \cup \mathcal{A}^2_{15}, \mathcal{C}^2_2 = \mathcal{A}^2_1 \cup \mathcal{A}^2_{14}, \mathcal{C}^2_3 = \mathcal{A}^2_2 \cup \mathcal{A}^2_{13}, \mathcal{C}^2_4 = \mathcal{A}^2_4 \cup \mathcal{A}^2_{11}, \mathcal{C}^2_5 = \mathcal{A}^2_7 \cup \mathcal{A}^2_8, \mathcal{C}^2_6 = \mathcal{A}^2_3 \cup \mathcal{A}^2_{12}, \mathcal{C}^2_7 = \mathcal{A}^2_5 \cup \mathcal{A}^2_{10}$ are permissible.

A *super-permissible* set is a collection of exactly one permissible set for each $i, 1 \leqslant i \leqslant t$.

Lemma 7. *For every (n, M, t) parallel RIO code, there exists a super-permissible set configuration.*

With this background, we propose an algorithm to construct (n, M, t) parallel RIO codes which attempts to populate the hypercube to construct the code. We then simulated our algorithm to obtain parallel RIO codes for $(n, M, 2), n \leqslant 6$. The maximum values of M for which codes could be obtained are listed in Table IV.

Lemma 8. *If an (n, M, t) parallel RIO code exists, Algorithm 1 will be successful in constructing it if all the sets $\bigcap_k \mathcal{C}^{i_k}_{l_k}$ corresponding to the existing parallel RIO codes have a single unique element.*

Proof: Since every (n, M, t) parallel RIO code has a super-permissible set configuration and Algorithm 1 spans all the super-permissible set configurations, it includes the super-permissible set configuration of the existing parallel RIO code. Since the entries in $\bigcap_k \mathcal{C}^{i_k}_{l_k}$ are unique, there is a unique way to populate these and hence algorithm 1 will always be successful in generating that parallel RIO code. ∎

Algorithm 1: (n, M, t) Parallel RIO code construction

```
input : the code parameters $(n, M, t)$, $t$-dimensional
        hypercube of length $M$ per dimension with
        all-zero entries, sets $\mathcal{B}_i, 1 \leqslant i \leqslant t$.
output: populated $t$-dimensional hypercube (if parallel
        RIO code exists).
1 Generate all possible permissible sets $\mathcal{C}^i_k, 1 \leqslant k \leqslant M$, for
  each $1 \leqslant i \leqslant t$.
2 Enlist all possible unique super-permissible set
  configurations using the ensemble of permissible sets
  generated in step 1, denote the total number of such
  configurations by $S$;
3 $j = 1$;
4 Populate hyperplanes $\mathcal{H}^i_m$ for each $m \in M$, for
  $1 \leqslant i \leqslant t$, by selecting elements from $\mathcal{C}^i_m$ from the
  $j-th$ super-permissible set configuration. The hypercube
  element at the intersection of $\bigcap_k \mathcal{C}^{i_k}_{l_k}$ for different
  messages $i_k$ can be assgined if $\bigcap_k \mathcal{C}^{i_k}_{l_k} \neq \emptyset$. If $\bigcap_k \mathcal{C}^{i_k}_{l_k}$ has
  a single element, that entry is populated. If $\bigcap_k \mathcal{C}^{i_k}_{l_k}$ has
  more than one element, any entry can be used. For eg. of
  the (4,7,2) parallel RIO code, the element at the
  intersection of the hyperplanes $\mathcal{H}^1_6$ and $\mathcal{H}^2_2$ is populated
  with the element 2120 which belongs to the intersection
  of $\mathcal{C}^1_6$ and $\mathcal{C}^2_2$.
5 If $\bigcap_k \mathcal{C}^{i_k}_{l_k} = \emptyset$, a parallel RIO code does not exist for this
  super-permissible set, increment $j$ by 1, go to step 4.
6 If all hyperplanes are populated, then the populated
  hypercube which is the parallel RIO code is constructed,
  EXIT.
7 If $j > S$, the algorithm could not construct a parallel
  RIO code, EXIT.
```

TABLE IV. $(n, M, 2)$ PARALLEL RIO CODES FOR $n \leqslant 6$

n	M	sum-rate
4	7	1.4037
5	11	1.3838
6	19	1.4160

V. Partial RIO Codes

In Lemma 4, we proved that a $(4, 8, 2)$ parallel RIO code does not exist. Note that the existence of such a code could have been possible according to the upper bound of Theorem 3, since for the 3-level flash memory, it is theoretically possible to store $\log_2(3) = 1.585$ bits per cell. While MLC and TLC flash memories attain their storage limits of 2 bits per cell and 3 bits per cell, 3-level flash memory is also expected to reach its storage limit, else its costs cannot be justified. Therefore, a desirable solution is to store 1.5 bits per cell or higher while 2 read thresholds are not permitted in order to read the two messages. An alternative solution then is to relax the RIO constraint of a single read threshold per page and permit some messages to be read with more a single read threshold. This motivates us to define a new family of RIO codes, called here *partial RIO codes*, which have relaxed constraints but inferior reading performance than RIO codes. The idea is that in order to decode some of the messages, the decoder will read two read thresholds instead of one. If for RIO codes, the i-th decoder reads the $(t+1-i)$-th read threshold then now the $(t+2-i)$-th threshold will be read as well. We note that this setup could be extended such that more and different read thresholds will be read, however we leave this generalization for an extended version of this paper. This model of two read

TABLE V. A $(4,8,2;1)$ PARTIAL RIO CODE

	0	1	2	3	4	5	6	7
0	1111	1112	1121	1211	2111	1122	2121	1221
1	1110	0002	1120	1210	2110	2210	2120	1220
2	1101	1102	0020	1201	2101	2201	1202	2102
3	1011	1012	1021	0200	2011	1022	2021	2012
4	0111	0112	0121	0211	2000	0122	0212	0221
5	1100	0012	0021	1200	2100	0022	2020	2002
6	1010	0102	1020	0201	2010	2200	0202	2112
7	0110	1002	0120	0210	2001	2211	1212	0220

thresholds corresponds in the WOM model to the case where both the encoder and decoder are informed with the previous state of the memory. In general there are four possible models which are extensively studied in [12]. Let us now formally define partial RIO codes.

Definition. *An $(n,M,t;t_1)$ partial RIO code for $1 \leqslant t_1 \leqslant t$ is a coding scheme comprising of n $(t+1)$-ary cells, t messages, an encoding map $\mathcal{E}$, and t decoding maps $\mathcal{D}_i$, for $1 \leqslant i \leqslant t$, defined as follows:*

$$\mathcal{E} : [M]^t \to [t+1]^n,$$

and the decoding map $\mathcal{D}_i$ is defined by

$$\mathcal{D}_i : [2]^n \to [M],\ 1 \leqslant i \leqslant t_1,$$
$$\mathcal{D}_i : [2]^n \times [2]^n \to [M],\ t_1+1 \leqslant i \leqslant t,$$

such that for all $\boldsymbol{m} = (m_1,\dots,m_t) \in [M]^t$, if $1 \leqslant i \leqslant t_1$, then $\mathcal{D}_i(RT_{t+1-i}(\mathcal{E}(\boldsymbol{m}))) = m_i$, and if $t_1+1 \leqslant i \leqslant t$, then $\mathcal{D}_i(RT_{t+1-i}(\mathcal{E}(\boldsymbol{m})), RT_{t+2-i}(\mathcal{E}(\boldsymbol{m}))) = m_i$

According to the last definition, parallel RIO codes are a special case of partial RIO codes in case $t_1 = t$. We could also define the nonparallel version of partial RIO codes, that is, the set of all t messages is not known in advance, however we omit this part due to the lack of space.

Example 2. The construction of a $(4,8,2;1)$ partial RIO code is given in the Table V. Notice that now we only need to guarantee that the binary vectors which are read for different symbols on the first page are different from each other. For the second page, it is enough to have the cell-state vectors to be different since both read thresholds are read. For example, if on the first page the read binary vector is 1010 or 0101 then the decoded symbol is 6 and for 0110, 1001, the decoded symbol is 7. For the second page, all cell-state vectors for different message symbols are different and thus it is possible to determine the stored message for each case.

In order to prove the optimality of the code construction in Example 2, we show that a (4,9,2;1) partial RIO code does not exist.

Theorem 9. *There does not exist a (4,9,2;1) partial RIO code.*

Proof: Assume in the contrary that such a code exist. Since there are 81 possible cell-state vectors, then in the encoding/decoding table, each of the 81 cell-state vectors will appear. In particular, there is a message $m \in [9]$ such that $\mathcal{D}_1(0000) = m$. However, there are 16 cell-state vectors $\boldsymbol{c} \in [3]^4$ such that $RT_2(\boldsymbol{c}) = 0000$ and thus only 9 of them could be used in the table. Therefore, there will be at least 7 cell-state vectors that can not appear in the table which results with a contradiction. ■

We conclude with the following condition on the existence of partial RIO codes.

Theorem 10. *An $(n,M,2;1)$ parallel RIO exists if and only if it is possible to divide the 2^n binary vectors into two sets of non-empty subsets $\mathcal{A} = \{A_1,\dots,A_M\}$ and $\mathcal{B} = \{B_1,\dots,B_M\}$ which satisfy the following conditions:*

TABLE VI. $(n,M,2;1)$ PARTIAL RIO CODES FOR $n \leqslant 10$

n	M	sum-rate	upper bound
4	8	1.5	1.5
5	14	1.5229	1.5229
6	24	1.5283	1.5480
7	43	1.5504	1.5598
8	76	1.5620	1.5620
9	132	1.5654	1.5702
10	230	1.5691	1.5753

1) For all $1 \leqslant i < j \leqslant M$, $A_i \cap A_j = \emptyset$,
2) For all $1 \leqslant i, j \leqslant M$, there exists $\boldsymbol{a}_i \in A_i$ and $\boldsymbol{b}_j \in B_j$ such that $\boldsymbol{a}_i \leqslant \boldsymbol{b}_j$.

Note that as apposed to Theorem 6, we did not need to add the constraint that $B_i \cap B_j = \emptyset$ since the vector of the first page is also read from the memory. We ran a computer search according to Theorem 10 in order to find more partial RIO codes for $t = 2$ and $n \leqslant 10$. The results are summarized in Table VI. The upper bound for each case on the sum-rate was derived in a similar way to the proof of Theorem 9.

Lastly in this section we discuss weather for MLC flash it is possible to construct meaningful partial RIO codes. The conventional read setup for this case is to read the lower page with a single read threshold and the upper page with 2 reads so the average number of read thresholds is 1.5. A meaningful RIO code for this case would be a $(n,M,3;2)$ code since a $(n,M,3;1)$ partial RIO code will have an average of 5/3 read thresholds. We state here that a $(3,3,3;2)$ partial RIO code does not exist, however a $(3,3,3;1)$ partial RIO code and a $(4,3,3;2)$ partial RIO code do exist.

REFERENCES

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

[2] Y. Cassuto and E. Yaakobi, "Short Q-ary fixed-rate WOM codes for guaranteed re-writes and with hot/cold write differentiation," *IEEE Trans. Inform. Theory*, vol. 60, no. 7, pp. 3942–3958, July 2014.

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

[4] A. Jiang, Y. Li, E. En Gad, M. Langberg, and J. Bruck, "Joint rewriting and error correction in write-once memories", *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 1067–1071, Istanbul, Turkey, July 2013.

[5] 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 Trans. Inform. Theory*, vol. 45, no. 1, pp. 308–313, September 1999.

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

[7] R.L. Rivest and A. Shamir, "How to reuse a write-once memory," *Inform. and Contr.*, vol. 55, no. 1–3, pp. 1–19, December 1982.

[8] F. Merkx, "WOM codes constructed with projective geometries," *Traitement du signal*, vol. 1, no. 2-2, pp. 227–231, 1984.

[9] E. Sharon and L. Alrod, "Coding scheme for optimizing random I/O performance," in *Non-Volatile Memories Workshop*, San Diego, April 2013.

[10] A. Shpilka, "Capacity achieving multiwrite WOM codes," *IEEE Trans. Inform. Theory*, vol. 60, no. 3, pp. 1481–1487, March 2014.

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

[12] 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.

[13] Y. Wu, "Low complexity codes for writing a write-once memory twice", *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 1928–1932, Austin, Texas, June 2010.

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