Construction of Random Input-Output Codes With Moderate Block Lengths

Eitan Yaakobi, *Member, IEEE*, and Ravi Motwani

***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. Multilevel 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. As a consequence, if a cell reads as bit 1 for a page (say page m), the cell also reads as bit 1 for all the successive pages $(m+1, m+2\cdots)$. Therefore, Sharon and Alrod showed that the design of RIO codes is equivalent to the design of *WOM codes*. The latter family of codes attracted substantial attention in recent years in order to improve the lifetime of flash memories by allowing writing multiple messages to the memory without the need for an erase operation. 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 maximum sum-rate of RIO codes, it allows the design of efficient high-rate codes with a moderate block length, which do not exist for WOM codes. We also study another family of RIO codes, called here *partial RIO codes*, that allow to find even more efficient codes while allowing to sense more than a single threshold to read some pages.**

***Index Terms*—Flash memory, random input/output codes, read latency, write once memories.**

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; see Figure 1. 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 are 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 page being read 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 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 read thresholds are required to read one page of data. Typically, 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.

A solution to making the number of sensings almost uniform for all pages was introduced by Bhat [6] in the form of using balanced Gray mapping. However, balanced Gray mapping does not permit the multi-step programming which is used to program pages in succession. Multi-step programming is essential to reduce the disturbing effects of the floating gate coupling. Further, the average number of sensings is still the same and hence there is no throughput improvement.

A solution to this problem of reducing the sensing overhead was recently proposed in [11] by Sharon and Alrod, 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. The authors of [11] showed a one-to-one correspondence between RIO codes and the well studied *WOM codes* [10]. Namely, a WOM code which permits writing k bits into n cells in t consecutive writes without erasing also gives a RIO code which permits writing t pages with 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

Manuscript received June 28, 2015; revised December 12, 2015; accepted February 23, 2016. Date of publication March 3, 2016; date of current version May 13, 2016. The material in this paper was presented in part at the 2014 IEEE Information Theory Workshop [9]. The associate editor coordinating the review of this paper and approving it for publication was K. Abdel-Ghaffar.

E. Yaakobi is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: yaakobi@cs.technion.ac.il).

R. Motwani is with the Non-Volatile Systems Group, Intel Corporation, Santa Clara, CA 95054 USA (e-mail: ravi.h.motwani@intel.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2016.2538219

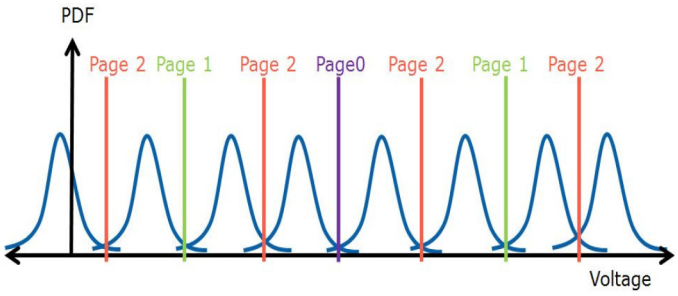


Fig. 1. The cell voltage distribution of an 8-level Flash memory.

cells without erasure permits writing 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 the 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 information availability 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 multiple read thresholds are allowed, 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 are not informed with the previous state of the memory; see [14] for a rigorous study of all four models and recent results in [7]. The traditional setting of WOM codes assumes the common and practical model, where 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 constructing 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 [11]. 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 with parameters that did not exist before. Finally, Section VI concludes the paper.

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 vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we say that $\boldsymbol{x} \geq \boldsymbol{y}$ if $x_i \geq y_i$ for all $1 \leq i \leq n$. We use $h(x)$ to denote the binary entropy function, that is $h(x) = -x\log(x) - (1-x)\log(1-x)$. For a function $F : X \to Y$, the set $Im(F)$ is the following set $Im(F) = \{F(x) \mid x \in X\}$. 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 [17]. 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 to the unrestricted-rate case as well. Furthermore, we assume that the write number is known on each write as this side information does not affect the asymptotic of the achievable rates; for more details, see [17].

Definition 1: 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 $1 \leq i \leq t$. 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 $\mathcal{E}_i(m, \boldsymbol{c}) \geq \boldsymbol{c}$ 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 $\mathcal{R}_{\text{sum}} = t\mathcal{R}$.

In [5], the capacity region of a binary t-write WOM was found to be

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

and $\log(t+1)$ was proved to be the maximum sum-rate. In [4], it was shown that the maximum sum-rate for q levels is $\log\binom{q+t-1}{t}$. It was also shown that all rate-tuples in the capacity region are achievable. Since then several code constructions were given with the intention of reaching high sum-rate, e.g. [1], [3], [8], [16]. Recently, some capacity achieving constructions were given both for two writes [2], [13], [15], [17] and

multiple writes [2], [12]. However, they all suffer either from a large block length and encoding/decoding complexities or are not applicable in the worst case. Thus, the problem of finding efficient WOM code constructions with moderate block length still remains an important challenge. For binary two-write WOM, the best sum-rate reported in the literature is 1.4928 [17] and a code construction which is capacity achieving was given. Recently, the work in [13] shows another capacity achieving construction with code length, encoding and decoding complexities being significantly better.

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, $1 \leq r \leq q-1$, between levels $r-1$ and r, called the r-th threshold. 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 $c_i \geq r$. That is, the vector $RT_r(\boldsymbol{c})$ is a length-n binary vector, where $RT_r(\boldsymbol{c})_i = 1$ if and only if $c_i \geq r$. The next proposition is an immediate property of the definition of this operation.

Proposition 2: For all $1 \leq r < s \leq q-1$, $RT_r(\boldsymbol{c}) \geq RT_s(\boldsymbol{c})$.

The idea behind RIO codes is to use $(t+1)$-ary cells in order to store t pages. 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 and 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 and $t-1$ of the non-programmed cells. This process repeats itself until we reach the t-th message and then only levels 0 and 1 are used of the non-programmed cells at that stage. Note that according to this process, in order to program the i-th page, $2 \leq i \leq t$, 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 Proposition 2 is that $RT_t(\boldsymbol{c}) \leq RT_{t-1}(\boldsymbol{c}) \leq \cdots \leq RT_1(\boldsymbol{c})$ 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 [11]; see Theorem 4 below.

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 3: 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 $1 \leq i \leq t$, defined as follows:

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

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

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

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

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

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

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

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

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

We note that in our constructions the range of the i-th encoding map, $\mathcal{E}_i$, is $\{0, t+2-i\}^n$, however we use $[t+1]^n$ to make this definition simpler and more general so it can apply to more code constructions. The following theorem was proved in [11] but is stated here for the completeness of the results.

Theorem 4 [11]: 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 well-known [3, 4, 2] WOM codes given by Rivest and Shamir [10]. 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 threshold is between levels 0 and 1.

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

Theorem 5: $\log(t+1)$ is an upper bound on the sum-rate of RIO codes and parallel RIO codes with t messages.

Proof: Since a RIO code uses $(t+1)$-ary cells, it is not possible to store together more than $(t+1)^n$ messages. Thus the sum-rate of these codes is bounded by $\log(t+1)$. ■

The upper bound from Theorem 5 can be achievable both for WOM codes and RIO codes only if the rates on different writes (or different messages for RIO codes) are not the same. However, for the fixed-rate case a better upper bound for WOM codes was given by Heegard [5] and thus holds for RIO codes as well. Hence, for example for $t=2$ the upper bound on the sum-rate is roughly 1.546 and for $t=3$ it is roughly 1.937. 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 consecutive messages to come simply because they are not available by the time of encoding. However, for parallel RIO codes, it is possible to program a different cell-state vector for a given message based on the consecutive 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 6: 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 exists. 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 $a\geq 2$. Otherwise, since the number of length-4 binary vectors of weight at most 1 is 5, 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, i.e. non-programmed cells, in $\boldsymbol{c}$ is at most 2. Therefore, we conclude that there does not exist a $[4,7,2]$ WOM code and a $(4,7,2)$ RIO code. ■

Next we show the existence of a $(4,7,2)$ parallel RIO code. The construction of such a code is given 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 in advance, 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. As another example, if the cell-state vector is 2011 and we seek to read the first message then by applying the threshold between levels one and two, we get the binary vector 1000 which corresponds to message 4. Similarly, if we seek to read the second message, then the binary vector 1011, which corresponds to message 3, is received by applying the threshold between levels zero and one. We will show that this construction is optimal by proving that a $(4,8,2)$ parallel RIO code does not exist. To do so, we first prove a more general theorem on the sufficient and necessary conditions for the existence of $(n,M,2)$ parallel RIO codes.

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

Theorem 7: An $(n,M,2)$ parallel RIO code exists if and only if there exist two sets of non-empty subsets of vectors $\mathcal{A}=\{A_0,\ldots,A_{M-1}\}$ and $\mathcal{B}=\{B_0,\ldots,B_{M-1}\}$, where for $0\leq i\leq M-1$, $A_i,B_i\subseteq\{0,1\}^n$, which satisfy the following conditions:

1) For all $0\leq i<j\leq M-1$, $A_i\cap A_j=\emptyset$ and $B_i\cap B_j=\emptyset$,
2) For all $0\leq i,j\leq M-1$, there exist $\boldsymbol{a}\in A_i$ and $\boldsymbol{b}\in B_j$ such that $\boldsymbol{a}\leq\boldsymbol{b}$.

Proof: We first show that if the condition in the theorem holds then there exists an $(n,M,2)$ parallel RIO code $\mathcal{C}$. Let $\mathcal{A}=\{A_0,\ldots,A_{M-1}\}$ and $\mathcal{B}=\{B_0,\ldots,B_{M-1}\}$ be the sets of subsets which satisfy the conditions in the theorem. We show how to construct the encoding and decoding maps, $\mathcal{E},\mathcal{D}_1,\mathcal{D}_2$, of the code $\mathcal{C}$. Assume that $(m_1,m_2)\in[M]^2$ are the two messages to be stored in the cells. According to the second condition, there exists $\boldsymbol{a}_{m_1}\in A_{m_1}$ and $\boldsymbol{b}_{m_2}\in B_{m_2}$ such that $\boldsymbol{a}_{m_1}\leq\boldsymbol{b}_{m_2}$. Then, the cell-state vector is encoded to be $\mathcal{E}(m_1,m_2)=\boldsymbol{c}$, where

$$c_i=\begin{cases}2 \text{ if } a_{m_1,i}=1,\\ 1 \text{ if } a_{m_1,i}=0, b_{m_2,i}=1,\\ 0 \text{ if } a_{m_1,i}=b_{m_2,i}=0.\end{cases}$$

For the decoding maps $\mathcal{D}_1$ and $\mathcal{D}_2$, for a binary vector $v\in[2]^n$, we let $\mathcal{D}_1(v)=i$, if $v\in A_i$, and similarly $\mathcal{D}_2(v)=j$ if $v\in B_j$. Note that $RT_2(\mathcal{E}(m_1,m_2))=\boldsymbol{a}_{m_1}$ and $RT_1(\mathcal{E}(m_1,m_2))=\boldsymbol{b}_{m_2}$, and according to the first condition in the theorem the decoding succeeds.

To prove the only if part of this theorem, let us assume that $\mathcal{C}$ is an $(n,M,2)$ parallel RIO code with an encoding map $\mathcal{E}$ and

decoding maps $\mathcal{D}_1$ and $\mathcal{D}_2$. Let $m_1 \in [M]$. For every $m_2 \in [M]$, let $\mathcal{E}(m_1, m_2)$ be the programmed cell-state vector, and let us define the set A_{m_1} to be

$$A_{m_1} = \{RT_2(\mathcal{E}(m_1, m_2)) \ : \ m_2 \in [M]\}.$$

Since $\mathcal{D}_1(RT_2(\mathcal{E}(m_1, m_2))) = m_1$ we conclude that all sets $A_0, \ldots, A_{m-1}$ are mutually disjoint. Similarly, we define the sets B_{m_2} for $m_2 \in [M]$

$$B_{m_2} = \{RT_1(\mathcal{E}(m_1, m_2)) \ : \ m_1 \in [M]\},$$

and conclude that they are mutually disjoint. Lastly, for given $m_1, m_2 \in [M]$ we let $\boldsymbol{a}_{m_1} = RT_2(\mathcal{E}(m_1, m_2))$ and $\boldsymbol{b}_{m_2} = RT_1(\mathcal{E}(m_1, m_2))$ and we get that $\boldsymbol{a}_{m_1} \leq \boldsymbol{b}_{m_2}$ while $\boldsymbol{a}_{m_1} \in A_{m_1}$ and $\boldsymbol{b}_{m_2} \in B_{m_2}$. ■

We now show that a $(4, 8, 2)$ parallel RIO code does not exist.

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

Proof: Assume in the contrary that such a code indeed exists. According to the conditions of Theorem 7, there exist two sets of subsets $\mathcal{A} = \{A_0, \ldots, A_7\}$ and $\mathcal{B} = \{B_0, \ldots, B_7\}$ which satisfy the conditions in Theorem 7. Since there are five vectors of length four and weight at most one, there exist at least three subsets in $\mathcal{A}$ which do not have any vectors of weight one. Furthermore, since there are six length-4 vectors of weight two, there exists at least one subset in $\mathcal{A}$ which has at most two vectors of weight two, and assume without loss of generality that this is the subset A_7. However, it is possible to verify that then there are at most seven different vectors which satisfy the second condition in Theorem 7. Thus, it is not possible to write 8 messages on the second page, in contradiction. ■

IV. Construction of Parallel RIO Codes

In this section, an algorithm to construct an (n, M, t) parallel RIO codes is outlined. When reading the i-th message, we apply a threshold between levels $t - i$ and $t - i + 1$, for $1 \leq i \leq t$. Let $\boldsymbol{c} \in [t + 1]^n$ be the cell-state vector and $RT_{t+1-i}(\boldsymbol{c}) \in [2]^n$ is the input to the i-th decoder $\mathcal{D}_i$ to decode the i-th message. 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 $RT_{t+1-i}(\boldsymbol{c}) = \boldsymbol{r}$. Evidently, the sum of the cardinality of all the sets $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$, over all $\boldsymbol{r} \in [2]^n$ is $(t + 1)^n$; that is, for $1 \leq i \leq t$, $\sum_{\boldsymbol{r} \in [2]^n} |\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}| = (t + 1)^n$.

A table of dimension t with M entries per dimension consists of M^t entries with each entry $\boldsymbol{c} \in [t + 1]^n$. A parallel RIO code can be represented by a t-dimensional table with each entry of the table representing the cell-state vector for that t-tuple message vector $\boldsymbol{m} \in [M]^t$. For $1 \leq i \leq t, m \in [M]$, a $(t - 1)$-dimensional sub-table $\mathcal{H}^i_m$ of this table with M^{t-1} entries corresponds to message i taking some value $m \in [M]$. The M^{t-1} entries of the $(t - 1)$-dimensional sub-table $\mathcal{H}^i_m$ will have all other possible symbols in $[M]$ for the other $t - 1$ messages. 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 $t - i + 1$ for any entry in the $(t - 1)$-dimensional sub-table $\mathcal{H}^i_m$. This parallel RIO code constraint can be mathematically expressed as

$$RT_{t+1-i}(\mathcal{H}^i_m) \cap RT_{t+1-i}(\mathcal{H}^i_{m'}) = \emptyset,$$

for all $1 \leq i \leq t$ and $m, m' \in [M], \ m \neq m'$. This constraint is equivalent to the condition that elements from $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$ can be used to populate a $(t - 1)$-dimensional sub-table $\mathcal{H}^i_m$ for only one possible $m \in M$ and no other $(t - 1)$-dimensional sub-table $\mathcal{H}^i_{m'}, m' \neq m$, for any i, $1 \leq i \leq t$. That is, the cell-state vectors from every set $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$ can belong to at most one $(t - 1)$-dimensional sub-table $\mathcal{H}^i_m$. The cardinality of $\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}$ plays an important role in the code construction. It is possible to verify that for $1 \leq i \leq t, \ \boldsymbol{r} \in [2]^n, \ |\mathcal{A}^i_{\mathcal{I}(\boldsymbol{r})}| = i^j (t + 1 - i)^{n-j}$, where $j = w_H(\boldsymbol{r})$.

Let us take for an example the $(4, 7, 2)$ parallel RIO code of Table III. For this RIO code, a table of dimension 2 $(t = 2)$ with $M = 7$ entries per dimension consisting of total 7^2 entries has to be populated. The 2-D table and one $(t - 1)$-dimensional which is a 1-D sub-table corresponds to message i taking a particular value m has M^{t-1} entries which corresponds to any row or column of Table III. Each column or row corresponds to message 1 or message 2 respectively taking a particular value. For one particular value of one message, the other message can take M possible values for $t = 2$. For $t = 2$, each row or column has $M^{t-1} = 7$ elements, which correspond to the other message resp. taking any of the 7 allowed values. In general, for one particular value of one message, the other messages can take M^{t-1} values which is called as a $(t - 1)$-dimensional sub-table. The figure for $t = 2$ is the 2-D table, like Table III. Each row and each column is a $(t - 1)$-dimensional sub-table. Each entry of the table is an intersection of the t $(t - 1)$-dimensional sub-tables, for example in Table III, the $(2, 2)$ entry of the table which corresponds to message 1 and message 2 both taking value 2, is 0020. For a given message, its M $(t - 1)$-dimensional sub-tables are disjoint. The columns correspond to the $(t - 1)$-dimensional sub-tables when reading the first message and the rows when reading the second message. When reading the first message columns, the possible binary length-4 vectors which are read for the seven different symbols in $\{0, 1, \ldots, 6\}$ are $0000, 0001, 0010, 0100, 1000, \{0011, 1010, 1101\}$ and $\{0101, 1100, 1001, 0111\}$, respectively. Similarly, for the second message, when reading the rows for the seven different symbols in [7] we respectively get the corresponding binary vectors $\{0000, 1111\}$, $\{0001, 1110\}$, $\{0010, 1101\}$, $\{0100, 1011\}$, $\{1000, 0111\}$, $\{0011, 1100\}$, $\{1010, 0101\}$. The $(t - 1)$-dimensional sub-tables for the first message for columns 1 through 7 are formed from subsets of the sets $\mathcal{A}^1_0, \mathcal{A}^1_1, \mathcal{A}^1_2, \mathcal{A}^1_4, \mathcal{A}^1_8, \mathcal{A}^1_3 \cup \mathcal{A}^1_{10} \cup \mathcal{A}^1_{13}, \mathcal{A}^1_5 \cup \mathcal{A}^1_7 \cup \mathcal{A}^1_9 \cup \mathcal{A}^1_{12}$, respectively, and for the second message, these are the sets $\mathcal{A}^2_0 \cup \mathcal{A}^2_{15}$, $\mathcal{A}^2_1 \cup \mathcal{A}^2_{14}$, $\mathcal{A}^2_2 \cup \mathcal{A}^2_{13}$, $\mathcal{A}^2_4 \cup \mathcal{A}^2_{11}$, $\mathcal{A}^2_7 \cup \mathcal{A}^2_8$, $\mathcal{A}^2_3 \cup \mathcal{A}^2_{12}$, $\mathcal{A}^2_5 \cup \mathcal{A}^2_{10}$, respectively.

The 2^n sets $\mathcal{A}^i_j, 0 \leq j \leq 2^n - 1$, for each message $i, 0 \leq i \leq t - 1$, have to be grouped into M groups so that each $(t - 1)$-dimensional sub-table can be populated from elements of a

TABLE IV
UPPER BOUNDS ON M FOR AN $(n, M, 2)$ PARALLEL RIO CODE FROM THE WEAK PERMISSIBILITY CONDITION

n	Upper Bound on M from WP	upper bound on sum-rate from WP
4	8	1.5
5	13	1.4802
6	23	1.5079
7	41	1.5307
8	72	1.5425
9	128	1.5556
10	219	1.555
11	381	1.5588
12	666	1.5632
13	1169	1.5679
14	2051	1.5717
15	3521	1.5709

group. The sets or union of sets selected to populate the $(t-1)$-dimensional sub-tables have to satisfy some conditions. We formulate these conditions now.

The M groups for each message are said to satisfy the *weak permissibility condition* if the number of elements in each group is at least M^{t-1}, and the total number of dropped elements (vectors not used in the parallel RIO code) in groups formed from single sets does not exceed $(t+1)^n - M^t$.

Lemma 9: Weak permissibility (WP) is a necessary condition for the grouping to satisfy in order for that grouping to generate a parallel RIO code.

Proof: For each message i, elements from the M groups are used to populate one $(t-1)$-dimensional sub-table. Since a $(t-1)$-dimensional sub-table has M^{t-1} elements, each group must have cardinality of at least M^{t-1}. The total number of cell state vectors are $(t+1)^n$. The parallel RIO code uses M^t of these vectors. Hence the maximum allowable dropped out elements are $(t+1)^n - M^t$. ■

It is easy to check the weak permissibility condition. From the weak permissibility condition, it is possible to obtain an upper bound on an achievable M for a given n. For $t=2$ case, this bound is given in Table IV.

We next introduce a permissibility condition which needs large number of computations and hence is not that easy to test for. For $1 \le i \le t$, a set of M disjoint subsets $\mathcal{C}_k^i \subset \{\mathcal{A}_j^i, 0 \le j < 2^n\}, 1 \le k \le M$, $\mathcal{C}_k^i = \bigcup_{b=0}^{f(k,i)} \mathcal{A}_{l(k,i,b)}^i$ is called *permissible* if

$$\left.\begin{array}{l} \sum_{b=0}^{f(k,i)} |\mathcal{A}_{l(k,i,b)}^i| \ge M^{t-1} \\ \sum_{i=1}^{t} \sum_{k=1}^{M} ((\sum_{b=0}^{f(k,i)} |\mathcal{A}_{l(k,i,b)}^i|) - M^{t-1}) \le (t+1)^n - M^t \end{array}\right\} \quad (1)$$

The subscript function of $\mathcal{A}_{l(k,i,b)}^i$ is a function of the page i, the message k and and the number of sets. The number of sets $f(k,i)+1$ is also a function of the page i and the message k, hence the running index b takes values from 0 until $f(k,i)$, meaning the $\mathcal{C}_k^i$ is contained in the union of $f(k,i)+1$ sets.

TABLE V
$(n, M, 2)$ PARALLEL RIO CODES FOR $n \le 6$

n	Upper Bound on M from (1)	M from Algorithm 1	sum-rate
4	7	7	1.4037
5	11	11	1.3838
6	19	19	1.4160

The first condition of permissibility ensures that the number of cell states available to populate the $(t-1)$-dimensional sub-table is at least the size of the $(t-1)$-dimensional sub-table. The second condition of permissibility implies that the number of elements dropped out does not exceed the maximum available limit.

Note that though the weak permissibility condition checks the first condition of (1), it checks the second condition only for those $\mathcal{C}_k^i$ which are formed from only one $\mathcal{A}_l^i$.

The computer simulation to generate the M groups for the i-th message satisfying the permissibility condition consists of populating an $M \times 2^n$ matrix $\mathcal{L}$ with ones and zeros. There can be at most one 1 entry in a column. The rows correspond to a group and the one entry in a particular row means that this set participates in that group. If $\boldsymbol{b}$ is a $2^n \times 1$ vector whose j-th entry is the cardinality of $\mathcal{A}_j^i$, then the first condition of (1) can be evaluated in the software as

$$\mathcal{L}\boldsymbol{b} \ge M^{t-1}[1\ 1\ \cdots\ 1]^T.$$

The second condition of (1) can be evaluated in the software as

$$\sum_{\text{all columns}} (\mathcal{L}\boldsymbol{b} - M^{t-1}[1\ 1\ \cdots\ 1]^T) \le (t+1)^n - M^t. \quad (2)$$

From the permissibility condition, it is possible to obtain an upper bound on achievable M for a given n. For $t=2$, using (2) for all possible $\mathcal{L}$ and M values, the obtained bound on M is given in Table V.

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

Lemma 10: For every (n, M, t) parallel RIO code, there exists a permissible set configuration.

Proof: Let the set of cell-state vectors corresponding to the message i, taking a value $m \in M$ for the existing parallel RIO code be $\mathcal{H}_m^i$. By definition of the parallel RIO code, $RT_{t+1-i}(\mathcal{H}_m^i) \cap RT_{t+1-i}(\mathcal{H}_{m'}^i) = \emptyset,\ \forall m, m' \in M,\ m \ne m',\ 1 \le i \le t$. Every $\mathcal{H}_m^i$ can hence be expressed as a subset of $\cup_l \mathcal{A}_{j_l}^i$. The collection of $\mathcal{C}_k^i = \cup_l \mathcal{A}_{j_l}^i, m \in [M], 0 \le i \le t-1$ is hence permissible. ■

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

TABLE VI
A (5,11,2) PARALLEL RIO CODE

	0	1	2	3	4	5	6	7	8	9	10
0	00000	11112	11121	11211	12111	11122	12211	22211	21111	21112	21211
1	00001	00002	11120	11210	12110	12120	12210	22210	21110	12220	21210
2	00010	11102	00020	11201	12101	22202	12201	22201	21101	21102	21201
3	00100	11012	11021	00200	12011	11022	12012	22011	21011	21012	12022
4	11000	00112	00121	00211	12000	00122	00222	22000	21000	20112	20211
5	00011	00012	00021	11200	12100	00022	12200	22100	21100	20020	21200
6	11010	00102	11020	00201	12010	12020	00202	22010	21010	02201	20022
7	10000	01112	01121	01211	02111	02222	02112	01221	20000	02212	02122
8	10110	01002	10120	10210	02001	20220	02002	10220	20110	20120	20210
9	01110	10002	01120	01210	02110	02120	02210	01220	20202	02220	02022
10	11001	11002	00120	00210	12001	22002	12002	22001	21001	21002	20200

TABLE VII
A (6,19,2) PARALLEL RIO CODE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	121211	111222	111111	111112	111121	111211	112111	121111	211111	111122	122211	112112	121112	211112	212211	221111	211221	122111	112211
1	121210	211210	111110	010012	111120	111210	112110	121110	211110	010022	122210	221210	020012	112220	111220	221110	211220	122110	112210
2	212101	211201	111101	111102	111201	111210	112101	121101	211101	221102	111202	221201	121102	211102	212202	221101	122102	122101	112201
3	212011	221021	111011	111012	222022	222011	112011	121011	211011	111022	122021	112012	121012	211012	211022	221011	121021	122011	121022
4	120211	220121	110111	110112	110121	110211	220222	120111	210111	110122	110212	120221	120112	210112	110221	220111	210221	210121	120122
5	202111	101222	101111	101112	101121	101211	102111	202121	201111	101122	101212	102112	102122	201112	202211	102121	201221	201121	102211
6	021211	011222	011111	011112	011121	011211	012111	021111	022222	011122	011212	012112	021112	012221	011221	012121	021121	022111	021122
7	212010	221020	001111	001112	111020	001211	112010	121010	211010	001122	001212	002112	002122	002221	001221	002121	121020	122010	002211
8	212100	211200	010111	010112	010211	012211	112100	020111	211100	010122	010212	221200	020112	200002	212200	221100	020121	122100	112200
9	020210	201200	011011	011012	011021	010210	012011	020110	201100	011022	022021	020220	012022	002220	010220	012021	020120	022011	102200
10	021201	200210	100110	011102	100120	100210	012101	021101	200110	100022	022201	012102	021102	200012	021202	220000	200220	022101	012201
11	202010	200211	100111	100112	011020	100211	012010	021010	200111	100122	022020	002002	020002	200112	100221	012020	021020	200121	200212
12	202011	200201	101011	100102	101021	100201	102011	202021	201011	101022	100202	102012	102022	201012	201022	102021	202022	201021	200202
13	202101	220020	011001	011002	110020	101201	012001	021001	201101	102202	101202	102102	021002	201102	202201	220010	120020	022001	201202
14	120200	210200	010101	010102	101120	010201	102110	020101	201110	220002	010202	220200	020102	210002	101220	220100	201220	201120	102210
15	212000	220021	001101	001102	110021	001201	002101	120011	210011	220012	001202	002102	120012	210012	210022	220011	120021	210021	002201
16	021210	210201	011110	110102	011120	011210	012110	021110	210101	220102	110202	220201	120102	210102	120202	220101	021120	022110	012210
17	021200	220120	001011	101002	001021	011200	012100	021100	201001	001022	022200	220210	002022	201002	110220	220110	210220	022100	012200
18	212001	000222	000111	000112	000121	000211	112001	121001	211001	000122	000212	112002	121002	211002	212002	221001	122002	122001	222002

The (5, 11, 3) and the (6, 18, 3) parallel RIO codes obtained using this algorithm are listed in Table VI and Table VII respectively.

Lemma 11: If an (n, M, t) parallel RIO code exists, Algorithm 1 will be successful in constructing it, if all the sets $\bigcap_{i=1}^{t} \mathcal{C}_{k_i}^{i}$ corresponding to the existing parallel RIO codes have a single unique element. k_i's here is the $i-th$ page messages which can take any value in $[M]$ for each i, $1 \leq i \leq t$.

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

We now comment on the complexity of algorithm 1 as a function of n. For each message, the 2^n sets have to be grouped into all possible M groups. For a given group for each message, the algorithm checks if the permissibility condition is satisfied. If it does, then the algorithm attempts to construct the code. As n increases, the number of sets increases exponentially and the search space quickly becomes memory and computationally intensive.

V. PARTIAL RIO CODES

In Lemma 8, 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 5, 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 than a single read threshold. This motivates us to define a new family of RIO codes, called here *partial RIO codes*, which have relaxed constraints and thus inferior read performance than RIO codes, but they can reach higher rates.

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 $\{(t+1-i)\text{-th}\}$ 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

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

input: the code parameters (n, M, t), t-dimensional table of length M per dimension with all-zero entries, sets $\mathcal{B}_i, 1 \leq i \leq t$.

output: populated t-dimensional table (if parallel RIO code exists).

1 Check if M satisfies the permissibility condition. If not, parallel RIO code does not exist, EXIT.

2 Generate all possible permissible sets $\mathcal{C}_k^i, 1 \leq k \leq M$, for each $1 \leq i \leq t$.

3 Enlist all possible unique permissible set configurations generated in step 1, denote the total number of such configurations by S;

4 $j = 1$;

5 Populate $(t-1)$-dimensional sub-tables $\mathcal{H}_m^i$ for each $m \in M$, for $1 \leq i \leq t$, by selecting elements from $\mathcal{C}_m^i$ from the $j-th$ permissible set configuration. The table element at the intersection of $\bigcap_{i=1}^t \mathcal{C}_{m_i}^i$ for different messages m_i can be assigned if $\bigcap_{i=1}^t \mathcal{C}_{m_i}^i \neq \emptyset$. If $\bigcap_{i=1}^t \mathcal{C}_{m_i}^i$ has a single element, that entry is populated. If $\bigcap_{i=1}^t \mathcal{C}_{m_i}^i$ has more than one element, any entry can be used. For example of the (4,7,2) parallel RIO code, the element at the intersection of the $(t-1)$-dimensional sub-tables $\mathcal{H}_6^1$ and $\mathcal{H}_2^2$ is populated with the element 2120 which belongs to the intersection of $\mathcal{C}_6^1$ and $\mathcal{C}_2^2$.

6 If $\bigcap_{i=1}^t \mathcal{C}_{m_i}^i = \emptyset$, this algorithm could not construct a parallel RIO code for this permissible set, increment j by 1, go to step 4.

7 Else, if all $(t-1)$-dimensional sub-tables are populated, then the populated table which is the parallel RIO code is constructed, EXIT.

8 If $j > S$, the algorithm could not construct a parallel RIO code, EXIT.

this generalization as it is application dependent and needs to be customized based on the system latency requirements. This model of two read 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 were extensively studied in [14], and recently in [7]. We now formally define partial RIO codes.

Definition 12: An $(n, M, t; t_1)$ **partial RIO code** for $1 \leq t_1 \leq t$ 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 $1 \leq i \leq t$, defined as follows:

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

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

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

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

TABLE VIII
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

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 non-parallel version of partial RIO codes, that is, the set of all t messages is not known in advance, however we refrain from diving into the details of this variation.

Example 2: The construction of a $(4, 8, 2; 1)$ partial RIO code is given in the Table VIII. 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 require different cell-state vectors since the two 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 if the read vector is 0110 or 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.

Lemma 13: There does not exist a (4,9,2;1) partial RIO code.

Proof: Assume in the contrary that such a code exists.

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, in order to read the first page only the read threshold between levels 1 and 2 is read and since there are 16 cell-state vectors $\boldsymbol{c} \in [3]^4$ such that $RT_2(\boldsymbol{c}) = 0000$, 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 in a contradiction. ■

The upper bound we derived in Lemma 13 for the case of $n = 4$ can be generalized for all values of n. We prove this result in the next theorem.

Theorem 14: For $n \geq 5$, if there exists an $(n, M, 2; 1)$ partial RIO code then

$$M^2 \leq 3^n - \sum_{i=0}^{\ell} \binom{n}{i} (2^{n-i} - M),$$

where $\ell = \left\lfloor \frac{(2-\log 3)n}{2} \right\rfloor$.

Proof: Assume that there exists an $(n, M, 2; 1)$ partial RIO code, so it can be represented in a two-dimensional table of size $M \times M$. Thus, we have that $M \leq 3^n$ or $M \leq 3^{n/2}$ and in particular for $n \geq 5$, $M < 2^{n-1}$. If there is no message $m \in [M]$ such that $\mathcal{D}_1(\boldsymbol{0}) = m$, then all 2^n ternary vectors $\boldsymbol{c}$ in which $RT_2(\boldsymbol{c}) = \boldsymbol{0}$ do not appear in the $M \times M$ table. If there exists a

message message $m \in [M]$ such that $\mathcal{D}_1(\mathbf{0}) = m$, then exactly M of these 2^n ternary vectors can appear in the table. Hence, we already conclude that $M^2 \leq 3^n - (2^n - M)$. Similarly, since $n \geq 5$, we have that $2^{2(n-1)} > 3^n$, and thus we conclude that $M < 2^{n-1}$. Therefore, for all n binary vectors of weight 1, we get that $n(2^{n-1} - M)$ ternary vectors won't appear in the table. Together we conclude that

$$M^2 \leq 3^n - (2^n - M) - n(2^{n-1} - M).$$

We repeat this process for all binary vectors of weight i as long as we can show that $M \leq 2^{n-i}$. We also know that $2^{2(n-i)} \leq 3^n$ and hence the maximum value of i in which we can claim that $M \leq 2^{n-i}$ is ℓ, where ℓ is the largest integer which satisfies $2^{2(n-\ell)} > 3^n$, i.e. $\ell = \left\lfloor \frac{(2-\log 3)n}{2} \right\rfloor$. Next, we conclude that there are at least $\sum_{i=0}^{\ell} \binom{n}{i}(2^{n-i} - M)$ ternary vectors which do not appear in the table and therefore the value of M satisfies

$$M^2 \leq 3^n - \sum_{i=0}^{\ell} \binom{n}{i}(2^{n-i} - M).$$

■

We conclude with the following condition on the existence of partial RIO codes. The proof follows the same outline as the proof of Theorem 7 and thus is omitted.

Theorem 15: An $(n, M, 2; 1)$ partial 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 $1 \leq i < j \leq M$, $A_i \cap A_j = \emptyset$,
2) For all $1 \leq i, j \leq M$, there exists $\boldsymbol{a}_i \in A_i$ and $\boldsymbol{b}_j \in B_j$ such that $\boldsymbol{a}_i \leq \boldsymbol{b}_j$.

Note that opposed to Theorem 7, we did not need to add the constraint that $B_i \cap B_j = \emptyset$ since the first message read is allowed for retrieving the second message. Note that for the $t = 2$, a partial RIO code has to satisfy constraints for only one message. There are no constraints imposed by the second message since it permits 2 reads. The permissibility condition for the partial RIO code for $t = 2$ is

$$\left.\begin{array}{l} \sum_{b=0}^{f(k,1)} |\mathcal{A}^1_{l(k,1,b)}| \geq M^{t-1} \\ \sum_{k=1}^{M}((\sum_{b=0}^{f(k,1)} |\mathcal{A}^1_{l(k,1,b)}|) - M^{t-1}) \leq (t+1)^n - M^t \end{array}\right\} \quad (3)$$

We ran a computer search according to (3) in order to find more partial RIO codes for $t = 2$ and $n \leq 15$. The results are summarized in Table IX. The upper bound for each case on the sum-rate was derived according to the bound in Theorem 14.

Lastly in this section we discuss whether 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 sensing operations 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.

TABLE IX
ATTAINABLE M FOR $(n, M, 2; 1)$ PARTIAL RIO CODES

n	M from (3)	sum-rate	upper bound on M
4	8	1.5	8
5	14	1.5229	14
6	24	1.5283	25
7	43	1.5504	44
8	76	1.5620	77
9	132	1.5654	134
10	230	1.5691	233
11	400	1.5716	403
12	688	1.5710	700
13	1216	1.5766	1218
14	2112	1.5778	2122
15	3660	1.5784	3668

VI. CONCLUSION

In this work we studied the design of several families of RIO codes. RIO codes are proven to be equivalent to WOM codes, however we made use of the fact that while encoding the information of all pages is available. This allowed us to define a new set of codes which we called parallel RIO codes. As expected, we achieve higher sum-rates with parallel RIO codes as compared to RIO codes. We proposed further generalizations by relaxing the read thresholds constraint and permitted reading some pages with more than one strobe. These codes which we called partial RIO codes have even higher sum-rates than parallel RIO codes. The tradeoff is that partial RIO codes have a higher read latency, but are suited for applications which require more efficient storage. We proposed an algorithm to construct parallel RIO codes and the same algorithm can also be used to construct partial RIO codes. We presented upper bounds on the message size based on permissibility condition for parallel and partial RIO codes.

ACKNOWLEDGMENT

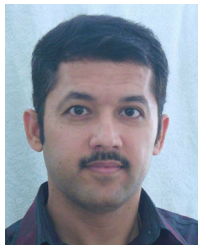
The authors thank three anonymous reviewers as well as the Associate Editor Prof. Khaled Abdel-Ghaffar for their valuable comments and suggestions, which have contributed for the clarity of the paper and its presentation.

REFERENCES

[1] A. Bhatia, M. Qin, A. Iyengar, B. Kurkoski, and P. H. Siegel, "Lattice-based WOM codes for multilevel flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 933–945, May 2014.
[2] D. Burshtein and O. Strugatski, "Polar write once memory codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5088–5101, Aug. 2013.
[3] G. D. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 5, pp. 697–700, Oct. 1986.
[4] 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. Inf. Theory*, vol. 45, no. 1, pp. 308–313, Sep. 1999.
[5] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 1, pp. 34–42, Jan. 1985.
[6] G. Bhat, "Balanced Gray codes," *Electron. J. Combin.*, vol. 3, no. 1, pp. 353–363, 1996.

[7] M. Horovitz and E. Yaakobi, "WOM codes with uninformed encoder," in *Proc. IEEE Inf. Theory Workshop*, Jerusalem, Israel, Apr./May 2015, pp. 1–5.
[8] F. Merkx, "WOM codes constructed with projective geometries," *Traitement Signal*, vol. 1, no. 2, pp. 227–231, 1984.
[9] R. Motwani and E. Yaakobi, "Construction of random input-output codes with moderate block lengths," in *Proc. IEEE Inf. Theory Workshop*, Hobart, TAS, Australia, Nov. 2014, pp. 602–606.
[10] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Inf. Control*, vol. 55, nos. 1–3, pp. 1–19, Dec. 1982.
[11] E. Sharon and I. Alrod, "Coding scheme for optimizing random I/O performance," in *Proc. Non-Volatile Memories Workshop*, San Diego, CA, USA, Apr. 2013.
[12] A. Shpilka, "Capacity achieving multiwrite WOM codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1481–1487, Mar. 2014.
[13] A. Shpilka, "New constructions of WOM codes using the Wozencraft ensemble," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4520–4529, Jul. 2013.
[14] 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.
[15] Y. Wu, "Low complexity codes for writing a write-once memory twice," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, pp. 1928–1932.
[16] Y. Wu and A. Jiang, "Position modulation code for rewriting write-once memories," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3692–3697, Jun. 2011.
[17] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5985–5999, Sep. 2012.

Eitan Yaakobi (S'07–M'12) received the B.A. degree in computer science and mathematics, the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, San Diego, CA, USA, in 2011. He is an Assistant Professor with the Computer Science Department, Technion—Israel Institute of Technology. Between 2011 and 2013, he was a Postdoctoral Researcher with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, USA. His research interests include information and coding theory with applications to nonvolatile memories, associative memories, data storage and retrieval, and voting theory. He was the recipient of the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010–2011.



Ravi Motwani received the B.E. degree in electronics engineering from Pune University, Pune, India, and the M.Tech. degree from IIT Kanpur, Kanpur, India, and the Ph.D. degree from IISc, Bangalore, India, both in electrical engineering. He holds the position of Principal Engineer and is the ECC Team Lead with the Nonvolatile Memory Systems Group, Intel Corporation, Santa Clara, CA, USA. He has held positions as Research Scientist with Philips Research Labs, Eindhoven, the Netherlands, from 1998 to 2000, and with Broadcom Corporation, USA, from 2005 to 2010. He has also held faculty positions at IIT Kanpur, Kanpur, India, from 2001 to 2003, and National University of Singapore, Singapore, from 2003 to 2005.