IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Information-Theoretic Sneak-Path Mitigation in Memristor Crossbar Arrays

Yuval Cassuto, Shahar Kvatinsky and Eitan Yaakobi

CCIT Report #864
July 2014

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

CCIT Report #864 July 2014

Information-Theoretic Sneak-Path Mitigation in Memristor Crossbar Arrays

Yuval Cassuto, *Senior Member, IEEE,*, Shahar Kvatinsky *Student Member, IEEE,*, and Eitan Yaakobi, *Member, IEEE*

Abstract

In a memristor crossbar array, functioning as a memory array, a memristor is positioned on each row-column intersection, and its resistance, low or high, represents two logical states. The state of every memristor can be sensed by the current flowing through the memristor. In this work, we study the sneak path problem in crossbar arrays, in which current can sneak through other cells, resulting in reading a wrong state of the memristor. Our main contributions are modeling the error channel induced by sneak paths, a new characterization of arrays free of sneak paths, and efficient methods to read the array cells while avoiding sneak paths. To each read method we match a constraint on the array content that guarantees sneak-path free readout, and determine the resulting capacity.

Index Terms

Codes for memories, sneak paths, constraint codes, Z channel, memristors, resistive memories, crossbar arrays.

I. Introduction

The memristor technology [14] allows packing storage cells in an unprecedented density, over a simple crossbar structure. The blessing of high storage density and architectural simplicity comes with a major caveat: *data-dependent behavior* [12]. The read accuracy, speed, and power consumption in memristor storage may all vary significantly depending on the instantaneous data stored in the crossbar array. This is clearly an undesired property for a storage medium, and a motivation for data representations that ensure that the physical content of the array corresponds to a well-behaving device. Memristor storage has already motivated a novel data representation for one instantiation of the data-dependence problem [9]. Here we address another very significant data-dependent phenomenon called *sneak paths* [12], causing the read correctness to depend on the array content. The importance of the sneak-path problem can be sensed by the significant body of research addressing it recently in the device and circuit literature [4]–[6], [8], [11], [12], [17], [18].

To understand the sneak-path problem in memristor arrays, we first show a simplified schematic of a memristor array in Fig. 1(a). Each row-column pair is connected by a resistor that can be in either the high-resistance state

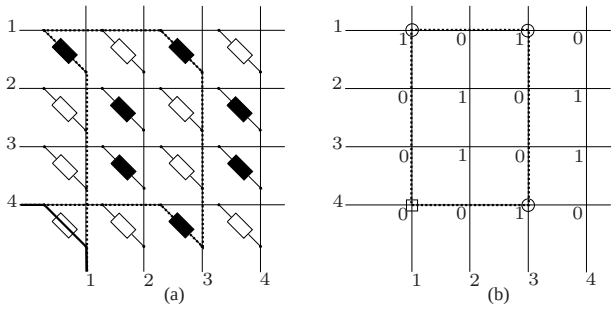

Fig. 1. (a) A memristor array as an array of programmed resistors – white: high resistance, black: low resistance. The high-resistance cell at location $(4, 1)$ has a sneak-path in parallel (plotted dashed), causing it to be read as low-resistance. (b) The corresponding logical values of the memristor array. The cell in the square frame has a sneak-path comprising of the three cells marked in circles.

Yuval Cassuto is with the Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa Israel (email: ycassuto@ee.technion.ac.il).

Shahar Kvatinsky is with the Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa Israel (email: skva@tx.technion.ac.il).

Eitan Yaakobi is with the Department of Computer Science, Technion – Israel Institute of Technology, Haifa Israel (email: yaakobi@cs.technion.ac.il).

The work of Y. Cassuto was supported in part by the European Union Marie Curie CIG grant, by the Intel Center for Computing Intelligence, and by the Israeli Ministry of Science and Technology. The work of E. Yaakobi was supported in part by the ISEF Foundation and by the Lester Deutsch Fellowship. Part of the results in the paper were presented at the IEEE International Symposium on Information Theory, Istanbul Turkey, July 2013 (reference [1]).

(marked white) or the low-resistance state (marked black). In Fig. 1(b) appear the corresponding logical values of the cells: logical "0" for the high-resistance state, and logical "1" for the low-resistance state. The sneak-path problem occurs when a resistor in the high-resistance state (white) is being read, while a series of resistors in the low-resistance state (black) exists in parallel to it, thereby causing it to be erroneously read as low-resistance. It is shown by the dashed line in Fig. 1(a) that the white resistor in (row,column) location $(4,1)$ has a sneak path that traverses the black resistors in locations $(4,3)$, $(1,3)$ and $(1,1)$. This dashed path is in parallel to the main current path of $(4,1)$ marked by a solid line.

In this paper we seek to combat memristor-array sneak paths using information-theoretic techniques. We first note that such an attempt was already presented in [16] by forcing the number of zeros and ones in every row and column to be the same. While this gives an elegant intuitive solution that can reduce the sneak path effect, our objective in this paper is to give a systematic information-theoretic study of the sneak-path problem and its mitigation. We begin the information-theoretic treatment of sneak paths in Section II by formulating sneak-path errors as a special kind of a *Z channel*, in which the (one-directional) transition probability depends on the array dimensions and on the distribution of the written bits. We give a precise closed-form calculation of the transition probability as a function of these parameters. This formulation of sneak-path arrays as error channels lays the foundation to coding for mitigating sneak-path errors.

An alternative approach to mitigating sneak paths is to eliminate them altogether by a proper constraint code. This approach is the subject of Sections III and IV. Here coding works to restrict the written bits in the array to bit assignments that do not induce any sneak path to any cell. The number of sneak-path free assignments to an $m \times n$ memristor array was calculated by Sotiriadis in [13]. Our contributions to the study of the sneak-path constraint are enabled by a new succinct characterization of sneak-path free arrays, which we give in section III. Then in Section IV, we give a capacity-achieving efficient encoder that maps unrestricted information to sneak-path free arrays. In Section V, we depart from the full-array model and consider two methods to avoid sneak paths by selectively grounding array rows. These methods enable a tradeoff between high power consumption (grounding many rows increases read power) and low storage capacity (grounding few rows enforces harder constraints and reduces capacity, in particular ungrounded full array results in zero capacity [13]). To the second grounding method we match a parametrized constraint, and calculate the resulting capacity. In this method, it turns out that a 1-dimensional (d,∞) run-length limited constraint provides sufficiency, and we prove that it also has the same capacity. The most interesting contribution from practical standpoint is that among these two approaches it is better to read a memristor array by grounding all rows outside a symmetric set of rows around the read row.

II. Analysis of Errors due to Sneak Paths

We start the information-theoretic treatment of sneak paths by examining the errors that they cause, and calculating the resulting error probabilities. In this section and throughout the paper we assume that a resistive cell at location (i,j) programmed to the "0" state (high resistance) is read in error as being at the "1" state (low resistance) when it is affected by at least one sneak path of any length, i.e., there exists a path as defined in Definition 1.

Definition 1. *Given a binary array A of size $m \times n$, we say that there is a **sneak path** of length $2k+1$ affecting the cell at position (i,j) if $a_{i,j} = 0$ and there exist $2k$ positive integers $1 \leqslant r_1, \ldots, r_k \leqslant m$ and $1 \leqslant c_1, \ldots, c_k \leqslant n$ for some $k \geqslant 1$ such that the following $2k+1$ cells satisfy*

$$a_{i,c_1} = a_{r_1,c_1} = a_{r_1,c_2} = \cdots = a_{r_{k-1},c_k} = a_{r_k,c_k} = a_{r_k,j} = 1.$$

The sneak path is a closed path originating from and returning to (i,j) and traversing "1"-state cells through alternating vertical and horizontal steps. The integers $r_1, \ldots, r_k$ and $c_1, \ldots, c_k$ are, respectively, the row and column indices of the traversed cells.

A. Calculating the sneak-path bit-error probability

In an $m \times n$ array we want to calculate the probability that a certain bit will be in error due to one or more sneak paths affecting it. To this end we restrict ourselves to sneak paths of length 3 ($k = 1$ in Definition 1), and define that a bit will be in error due to sneak path if the following two conditions are met:

1) The bit value is 0.
2) The bit location (i,j) has at least one combination c_1, r_1 that induces a sneak path defined by

$$a_{i,c_1} = a_{r_1,c_1} = a_{r_1,j} = 1. \tag{1}$$

According to the definition of sneak path given in (1), we only consider in the analysis sneak paths with 3 cells, which is a special case of the $2k+1$-cell sneak path given in Definition 1. We do so for two reasons. One is that sneak paths with more than 3 (5, 7, etc.) cells are less prone to errors because of their higher resistance. The second is that analyzing longer sneak paths is much more difficult. In Section III we show that when determining sneak-path existence in a full $m \times n$ array, it is sufficient to consider 3-cell sneak paths as in (1).

A bit error due to sneak paths can thus be characterized using the Z channel depicted in Figure 2. If a 0 is written, it is read in error with probability P, the probability that there is a sneak path affecting it. If a 1 is written, it is always read correctly.

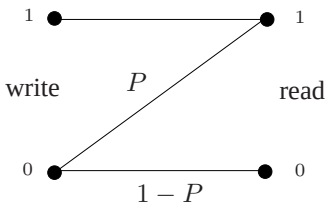

Fig. 2. Sneak-path read errors as a Z channel. A written 0 is read as 0 if no sneak path affects it and as 1 otherwise. A written 1 is always read correctly.

It is now our objective to calculate the sneak-path transition probability P. The main challenge in finding P stems from the fact that there are many possible c_1, r_1 combinations, and multiple of them may exist simultaneously for the same array assignment. We now define the problem formally.

Problem 2. *Given an array of dimensions $m \times n$, where bits are written to array locations such that $\Pr(a_{i,j} = 1) = q$, $\Pr(a_{i,j} = 0) = 1 - q$, i.i.d. for all (i, j). What is the probability P that a 0-written array bit is read in error due to sneak path?*

It is clear that the answer to Problem 2 depends on the parameter q. For example, it is possible to trivially make the sneak-path transition probability identically zero by setting $q = 0$ (hence having no 1s in the array to create sneak paths). However, this would not be a wise choice as the resulting information rate is zero. The sneak-path transition probability also depends on the array dimensions, hence we re-denote P as $P(m, n, q)$.

Theorem 3. *The transition probability due to sneak paths in an $m \times n$ array with parameter q equals*

$$P(m,n,q) = 1 - \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} \binom{m-1}{u} \binom{n-1}{v} q^{u+v} (1-q)^{m-1-u+n-1-v+uv}. \tag{2}$$

Proof: The proof proceeds by summing the probabilities of bit assignments for which there is *no* sneak path affecting cell (i, j). Taking the complement yields $P(m, n, q)$.
We consider a location (i, j) where the bit value is 0. Suppose column j has u 1s in row locations taken from $\{1, \ldots, m\} \setminus i$, and row i has v 1s in column locations taken from $\{1, \ldots, n\} \setminus j$. Then in order for cell (i, j) to have no sneak path, each intersection of a 1 in column j with a 1 in row i must have a 0 value. An example for an array with no sneak path for cell $(i, j) = (1, 1)$ is given in Figure 3.

The probability that all these intersections have 0s is $(1 - q)^{uv}$. Now all that is needed to obtain the second term of (2) is to sum over all u from 0 to $m - 1$ and all v from 0 to $n - 1$ and weight with their respective probabilities.

■

Theorem 3 gives an exact closed-form expression for the probability that randomly selecting the $m \times n$ array bits i.i.d. with parameter q will result in at least one sneak path affecting location (i, j). Note that the same expression applies to any array location, as (2) does not depend on (i, j). This implies a simple coding scheme where the encoder chooses a bias q, and obtains information reliability given by (2). It is important to observe that for two locations (i_1, j_1) and (i_2, j_2) on the same array, sneak-path error events are *not independent*. For example, if $i_1 = i_2$, knowing that (i_1, j_1) has a sneak path makes a sneak path for (i_2, j_2) more likely.

It may be the case in practice that one sneak path is not sufficient to cause a bit error. Rather, the sensing circuit may have sufficient margins to tolerate up to $L - 1$ sneak paths affecting an array location, in which case a bit error due to sneak paths requires at least L sneak paths affecting the same array location. For this case we derive a generalized expression for the bit-error probability due to L or more sneak paths.

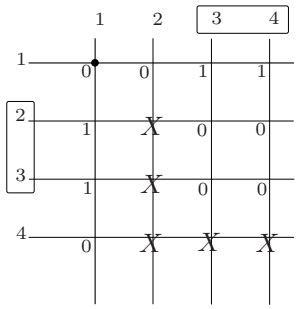

Fig. 3. An example of an array with no sneak path for cell $(1,1)$. Given 0/1 assignments in row $i=1$ and column $j=1$, the intersections of the 1-rows with the 1-columns must be set to zeros. In the rest of the array locations, the value can be arbitrary, marked with X.

Theorem 4. *The transition probability due to* L ***or more*** *sneak paths in an* $m \times n$ *array with parameter* q *equals*

$$P_L(m,n,q) = \sum_{l=L}^{(m-1)(n-1)} \sum_{u=1}^{m-1} \sum_{v=1}^{n-1} \binom{m-1}{u} \binom{n-1}{v} q^{u+v+l} (1-q)^{m-1-u+n-1-v+uv-l} \binom{uv}{l}. \tag{3}$$

Proof: Given u 1s in column j and v 1s in row i, an array will induce l sneak paths on location (i,j) if it has exactly l 1s out of the uv cells that intersect a 1-row of column j and a 1-column of row i. There are $\binom{uv}{l}$ combinations to choose these l 1s. To restrict to exactly l 1s in the probability expression, we add (on top of (2)) l to the exponent of q to get q^{u+v+l} and $uv-l$ to the exponent of $1-q$ to get $(1-q)^{m-1-u+n-1-v+uv-l}$. Summing for all l greater or equal to L yields (3). ■

One can verify that $P(m,n,q) = P_1(m,n,q)$, hence Theorem 3 is a special case of Theorem 4.

B. Asymptotic analysis

In this section we move to analyze the sneak-path transition probability when the array dimensions m and n tend to infinity. This analysis will help finding out whether reliable readout is possible in the limit of large memory arrays. We will explore the case where both m and n tend to infinity, and then the case where one of m,n tends to infinity while the other is a constant. The first question we pose is whether we can store bits reliably on arrays with both m and n tending to infinity. The following theorem answers this question to the negative.

Theorem 5. *The transition probability* $P(m,n,q)$ *tends to* 1 *if* m *and* n *tend to infinity, for any constant* q*.*

Proof: As we did in the proof of Theorem 3, we will look at the complement of $P(m,n,q)$ and prove that it tends to 0 for m and n tending to infinity. Suppose that column j is assigned a particular vector $\boldsymbol{u}$ with weight $u=1$. Then the probability that there is no sneak path conditioned on $\boldsymbol{u}$ in column j is given by

$$1 - P(m,n,q|\boldsymbol{u}) = \sum_{v=0}^{n-1} \binom{n-1}{v} q^v (1-q)^{n-1-v} (1-q)^v. \tag{4}$$

The power of q in (4) is the number of 1s in row i and the first power of $1-q$ is the number of 0s in row i. The second power of $1-q$ is the number of zeros required to not have sneak path when the weight of $\boldsymbol{u}$ is 1. Simplification of (4) yields

$$1 - P(m,n,q|\boldsymbol{u}) = (1-q)^{n-1}(1+q)^{n-1} = (1-q^2)^{n-1} \underset{n\to\infty}{\longrightarrow} 0. \tag{5}$$

It is clear that for any vector $\boldsymbol{u'}$ in column j with weight $u \geqslant 1$, the probability of having no sneak path conditioned on $\boldsymbol{u'}$ similarly tends to 0. (Intuitively, more 1s in column j mean fewer assignments to the remaining bits that do not cause sneak-path.) Hence the probability of no sneak path conditioned on the weight of column j being greater or equal to 1 tends to 0 as n tends to infinity. To prove that the same applies even without conditioning, we observe that the probability that column j has weight 0 ($\boldsymbol{u'}$ is the all-zero vector) tends to 0 as m tends to infinity. This completes the proof. ■

To overcome the impossibility result of Theorem 5, we resort to the solution of physically limiting the sneak-path effect to a constant number b of rows. Instead of being vulnerable to sneak paths from m rows, where m tends to infinity, we now only consider a subset of rows of size b, where b is a constant. Practically speaking, reducing the number of rows per array is simple, but incurs some implementation cost. The number of columns n is assumed as before to be large, and tending to infinity in the analysis. We call such $b \times n$ arrays *semi-infinite*. For a constant number b of rows (which include row i and $b-1$ additional rows), the probability that column j has all 0s equals

$$(1-q)^{b-1}.$$

Since all 0s in column j guarantees that there are no sneak paths, we can bound from above the transition probability

$$P(b,n,q) \leqslant 1-(1-q)^{b-1}. \tag{6}$$

The bound (6) applies to any n, and becomes tighter as n tends to infinity. For the forthcoming analysis, we use the notation $\hat{P}$ for this upper bound on the transition probability

$$\hat{P} \triangleq 1-(1-q)^{b-1}.$$

Using this upper bound, we next calculate the capacity of reliable storage on semi-infinite arrays with sneak paths.

C. Capacity of semi-infinite arrays with sneak paths

In this sub-section we seek to find the information capacity of semi-infinite arrays with sneak paths. Bits stored on such arrays are subject to errors due to sneak-paths, and the capacity – denoted $C(b)$ – gives the amount of information that can be stored reliably on one physical array bit (hence $0 \leqslant C(b) \leqslant 1$). Information theoretically, errors due to sneak paths in semi-infinite arrays are modeled using the Z channel depicted in Figure 2. The uniqueness of sneak-path errors over a standard Z channel is that the transition probability P depends on the input distribution through the parameter q. Accordingly, the calculation of the channel capacity will need to consider this coupling between the input and the channel.

Proposition 6. *The capacity of the semi-infinite sneak-path array with b rows is given by*

$$C(b) = \operatorname{argmax}_q \left\{ H\left[(1-q)(1-P(q))\right] - (1-q)H\left[P(q)\right] \right\}, \tag{7}$$

where

$$P(q) = \hat{P} = 1-(1-q)^{b-1},$$

and $H(\cdot)$ is the binary entropy function.

Proof: This is the standard Z-channel capacity, only with substitution of the coupled transition probability $P(q) = 1-(1-q)^{b-1}$ and maximization over q. ■

We calculated the capacity for several values of b and listed them in Table I. For each b the second column lists the resulting capacity, and the third column shows the q values at which the respective capacities are obtained.

b	$C(b)$	optimal q
2	0.383	0.287
3	0.245	0.203
4	0.181	0.157
5	0.143	0.128

TABLE I
THE CAPACITY OF SEMI-INFINITE SNEAK PATH ARRAYS FOR DIFFERENT VALUES OF b.

D. Discussion

There are a few insights to take from the results of this section. First, it is clear from Table I that even for semi-infinite arrays the capacity is very low – quite away from 1. Hence one cannot expect high-rate reliable storage without taking extra measures to combat sneak paths. Second, due to the dependence between sneak-path errors within the array, sneak-path error-correction coding needs to be performed *across arrays* (i.e., each bit in a codeword is assigned to a different array), so that errors within a code block can be assumed i.i.d. It is possible to assign multiple array bits to the same code block so long as the error dependence is accommodated into the error model. Third, the storage rate may be made much higher by reducing both m and n to finite values that give small transition probabilities, which can be calculated by (2).

III. Characterization of Sneak-Path Free Arrays

The approach manifested in the previous section is to regard sneak paths as a source of errors, and using error-correcting codes to combat them. Now in this section we explore an alternative approach of constraining the array to bit assignments that do not have any sneak path for any cell (i, j). Such an approach would give a complete solution to the sneak-path problem without need to employ error-correcting codes.

To obtain sneak-path free arrays, let us first define formally and mathematically the required constraint. Recall that for a cell at location (i, j) that is programmed to "0" a sneak path is defined according to Definition 1. Hence we say that an array A satisfies the **sneak-path constraint** if no cell within it has a sneak path *of any length*. In these cases we call the array A a **sneak-path-free array**.

The sneak-path constraint was already introduced and studied in [13] with application to nanowire resistive crossbar switching networks (R-CSNs). This previous work addressed the same problem of high-resistance cells being "short-circuited" by paths of cells at low-resistance state. The contributions of [13] include an exact count of the number of "0", "1" $m \times n$ arrays that are distinguishable by measuring resistance at the array row/column terminals. This count can be easily seen to be identical to the number of distinct sneak-path-free arrays. However, the more refined characterization of the sneak-path constraint pursued here allows obtaining superior storage information rates for more general sneak-path problems motivated by memristor arrays. For completeness and clarity we include in the presentation results for the simple sneak-path model, which can be implied by results in [13].

For the ability to extend sneak-path-free coding results to more general models, it is useful to represent the sneak-path constraint by a new, more succinct constraint, which is later shown to be equivalent. It turns out that the existence of sneak paths of any length in a memristor array can be perfectly characterized by an abstract constraint, which we call the *isolated zero-rectangle constraint*.

Definition 7. *A binary array A has an **isolated zero rectangle** if there are four positive integers $i_1 \neq i_2$ and $j_1 \neq j_2$ such that*

$$a_{i_1,j_1} + a_{i_1,j_2} + a_{i_2,j_1} + a_{i_2,j_2} = 3.$$

That is, the value of exactly one out of the four cells in the rectangle formed by these four positions is zero.

Note the similarity between Definition 7 and Definition 1 for the special case of $k = 1$. The difference is that Definition 1 characterizes sneak paths affecting the particular cell at location (i, j), while Definition 7 characterizes the existence of sneak paths affecting *any* cell in the array. An array A satisfies the **isolated zero rectangle constraint** if it has no isolated zero rectangles and then it is called an **isolated zero rectangle free array**.

According to the last definition, a "0" belongs to an isolated-zero rectangle if it is part of any rectangle in the array, all of whose remaining vertices are "1"s. For example, the cell in the $(4, 1)$ location in Fig. 1(b) belongs to an isolated zero rectangle because it is part of a rectangle (marked by a dashed line) with three "1"s at locations $(1, 1)$, $(1, 3)$ and $(4, 3)$. There are no other isolated zero rectangles in the array.

Next we show that a memristor array is free of sneak paths of any length if and only if it has no isolated zero-rectangles. Note that sneak paths may consist of any odd number of cells greater than one, not necessarily three as in the rectangle case. However, this property tells us that rectangles, i.e. sneak paths of length three, provide a complete characterization of the existence of sneak paths.

Theorem 8. *The sneak path constraint and the isolated zero rectangle constraint are equivalent.*

Proof: We will show that an array has a sneak path if and only if it has an isolated zero rectangle. We show only one direction as the other one is immediate.

Let us assume to the contrary that there exists an array A which has a sneak path affecting the (i, j) cell and yet it satisfies the isolated zero rectangle constraint. First note that $a_{i,j} = 0$ and there is a path as defined in Definition 1 starting at the i-th row and ending at the j-th column. Assume the vertices of this path are the cells at positions $(i, c_1), (r_1, c_1), (r_1, c_2), \ldots, (r_{k-1}, c_k), (r_k, c_k), (r_k, j)$ for some $k \geqslant 1$, and these array cells have value "1".

We will show by induction that for for all $1 \leqslant h \leqslant k$, $a_{r_h,c_1} = 1$. This property holds for $h = 1$ since the (r_1, c_1) cell is part of the sneak path. Assume the claim is true for some $1 \leqslant h < k$, that is, $a_{r_h,c_1} = 1$. We will show that $a_{r_{h+1},c_1} = 1$ as well. Note that the vertices $(r_h, c_{h+1}), (r_{h+1}, c_{h+1})$ belong to the sneak path and hence $a_{r_h,c_{h+1}} = a_{r_{h+1},c_{h+1}} = 1$. Therefore, in the rectangle formed by the vertices

$$(r_h, c_{h+1}), (r_h, c_1), (r_{h+1}, c_{h+1}), (r_{h+1}, c_1)$$

the first three cells have value one. Therefore, according to the assumption that there is no isolated zero rectangle we conclude that $a_{r_{h+1},c_1} = 1$.

From the last claim we get in particular that $a_{r_k,c_1} = 1$. Since the vertices $(i, c_1), (r_k, j)$ belong to the sneak path, we have $a_{i,c_1} = a_{r_k,j} = 1$ and since the sneak path affects the cell at position (i, j) we also have $a_{i,j} = 0$. Therefore, there exists a sneak-path with three cells $(i, c_1), (r_k, c_1), (r_k, j)$ in contradiction with the assumption that there are no isolated zero rectangles. ■

From the isolated zero rectangle characterization it is implied that for sneak paths to not exist in the array, the "1" cell locations in any pair of rows (or columns) must have either full overlap or no overlap. For example, rows 2, 3 in Fig. 1(b) have full overlap of "1"s, rows 2, 4 have no overlap of "1"s, and thus no sneak paths exist between these row pairs. However, rows 1, 4 have neither full-overlap nor no-overlap, and thus introduce a sneak path.

Lemma 9. *An array A is an isolated zero rectangle free array if and only if the "1"s in every two rows either completely overlap or are disjoint.*

Proof: It is clear that the condition is sufficient. If "1"s either completely overlap or have no overlap between every pair of rows, then every rectangle has either 0,1, 2 or 4 "1"s.

To prove necessity, assume to the contrary that the condition does not hold. That is, there are two rows, say the i-th and j-th rows, such that the ones in these rows neither overlap nor are disjoint. Assume without loss of generality that there are more ones in the i-th row and assume that there are $\ell_i \geqslant 2$ ones in positions $1, \ldots, \ell_i$. Since the ones in the two rows are not disjoint, there is $1 \leqslant k \leqslant \ell_i$ such that $a_{j,k} = 1$, and since they do not fully overlap, there is $1 \leqslant h \leqslant \ell_i$, $h \neq k$ such that $a_{j,k} = 0$. Thus, the rectangle formed by the vertices $\{(i, k), (i, h), (j, k), (j, h)\}$ is an isolated-zero rectangle and so the array A does not satisfy the isolated zero rectangle constraint. ■

Let $N(m, n)$ be the number of $m \times n$ arrays satisfying the isolated zero-rectangle constraint. An exact count of $N(m, n)$ (for an equivalent constraint) is derived in [13]. For the sake of completeness, we provide a proof of the result that uses the isolated zero rectangle constraint and its characterization in Lemma 9.

First, we denote by $S(k, \ell)$ the number of distinct ways that a set of k elements can be partitioned into ℓ nonempty subsets, where it is known that

$$S(k,\ell) = \frac{1}{\ell!}\sum_{t=0}^{\ell}(-1)^{\ell-t}\binom{\ell}{t}t^k = \frac{1}{\ell!}\sum_{t=0}^{\ell}(-1)^{t}\binom{\ell}{t}(\ell-t)^k.$$

This is known as the Stirling number of the second kind [15].

Lemma 10. *The value $N(m, n)$ is expressed by*

$$N(m,n) = 1 + \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\binom{m}{i}\binom{n}{j}\sum_{\ell=1}^{\min\{m-i,n-j\}} S(m-i,\ell)S(n-j,\ell)\ell!.$$

Proof: Assume A is an array which satisfies the isolated zero-rectangle constraint, which is not the all zero array. Assume that A has i zero rows and j zero columns where $0 \leqslant i \leqslant m-1$ and $0 \leqslant j \leqslant n-1$. There are $\binom{m}{i}$ options to choose these rows and $\binom{n}{j}$ to choose the columns. After removing these i rows and j columns we obtain an $(m-i) \times (n-j)$ array A' with no zero rows or zero columns.

According to Lemma 9, the rows of A' can be partitioned into some $1 \leqslant \ell \leqslant m-i$ sets such that the rows in every set are identical. The number of distinct ways to partition the $m-i$ rows into ℓ nonempty sets is $S(m-i, \ell)$. Note that if the rows are either identical or their "1" positions do not overlap then the same property holds for the columns. Therefore, the columns can be partitioned into ℓ nonempty sets, where $1 \leqslant \ell \leqslant n-j$ and the number of such options is similarly $S(n-j, \ell)$. Finally, there are $\ell!$ options to match between the ℓ sets of rows and ℓ sets of columns, yielding the expression

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\binom{m}{i}\binom{n}{j}\sum_{\ell=1}^{\min\{m-i,n-j\}} S(m-i,\ell)S(n-j,\ell)\ell!,$$

for the number of possible arrays A. Together with the all zero array, we get the result stated in the lemma. ■

The second, more compact, expression for $N(m, n)$ in [13] can similarly be obtained using the isolated zero rectangle constraint.

Lemma 11. *The value $N(m,n)$ can be expressed by*

$$N(m,n) = \sum_{\ell=0}^{\min\{m,n\}} S(m+1,\ell+1)S(n+1,\ell+1)\ell!.$$

Proof: The proof is very similar to the one given in Lemma 10. This time we add one more zero column and one more zero row so the array size is $(m+1)\times(n+1)$. Assume that A is an $(m+1)\times(n+1)$ array which satisfies the isolated zero rectangle constraint and its last row and column are all zeros. According to Lemma 9, the rows of A can be partitioned into groups such that all the rows in every group are identical. The columns are partitioned as well into nonempty disjoint sets. Since the last row and last column are all zeros there is definitely a set of columns corresponding to a set of rows which are all zeros. If ℓ corresponds to the number of sets of nonzero rows, then there are $S(n+1,\ell+1)$ options to partition the columns and $S(m+1,\ell+1)$ options to partition the rows. Since the last set of rows and the last set of columns correspond to the last zero row and last zero column we only need to match between the ℓ sets of non-zero columns and ℓ sets of non-zero rows. Hence, we get that the number of such arrays is given by

$$\sum_{\ell=0}^{\min\{m,n\}} S(m+1,\ell+1)S(n,\ell+1)\ell!.$$

■

Unfortunately, the asymptotic behavior of the value $N(m,n)$ for m and n large enough states that $\log_2 N(m,n) \approx (m+n)\log_2(m+n)$ in case both m and n approach infinity and the ratio m/n approaches some positive number [13]. Thus, under these conditions it is derived that

$$\frac{\log_2 N(m,n)}{mn} \longrightarrow 0, \tag{8}$$

which implies a 0 asymptotic storage capacity. In fact, this behavior holds for all values of m and n which approach infinity (that is, the ratio m/n does not have to approach to a positive number). This indicates that the sneak path constraint is too strong, and we need to find milder ways to avoid sneak paths without ending up with zero capacity. This will be the topic of Section V.

IV. Encoding of Sneak Path Free Arrays

Even though the asymptotic storage capacity of the sneak path constraint approaches zero for m and n large enough, the encoding problem of such arrays is still important. For simplicity we assume in this section that $n = m$ and they are both large enough.

In [13], a low complexity and very efficient mapping was presented, and the number of information bits that this mapping can carry is $n\log n$ (for simplicity it was assumed that n is a power of two but that can be easily modified for arbitrary n). However, according to the derivations in [13], the number of bits that can be represented by all sneak-path free arrays is roughly $2n\log n$. Thus, the mapping in [13] reaches approximately only a half of the number of bits that could be stored.

We show here another mapping that even though has higher encoding and decoding complexities, can asymptotically reach the maximum number of bits that can be represented, i.e. $2n\log n$. To simplify the mapping presentation, we dropped all floor and ceiling functions, so some of the values are not necessarily integers as required. This may incur a small loss in the number of stored bits, however this loss is negligible.

Let S_1 be the set of all partitions of the numbers $\{1,\ldots,n\}$ into L groups, each consisting of $\frac{n}{L}$ numbers. Alternatively, we can treat S_1 as the set of all multipermutations over $\frac{n}{L}$ numbers where each number appears L times. The size of S_1 is

$$s_1 = |S_1| = \frac{n!}{(\frac{n}{L})!^L \cdot L!}.$$

Assume for now that there is a one-to-one mapping with efficient encoding and decoding maps

$$F_1 : \{0,1\}^{\log s_1} \to S_1$$

between all binary vectors of length $\log s_1$ and S_1. Let S_2 be the set of all permutations of L numbers, so $s_2 = |S_2| = L!$, and similarly, assume that there is a mapping with efficient encoding and decoding maps

$$F_2 : \{0,1\}^{\log s_2} \to S_2.$$

Our approach follows the proof of Lemma 10, which uses the if and only if condition in Lemma 9. We encode only arrays where the rows, columns are partitioned into L sets of n/L rows, columns, respectively. Thus, every array is represented by: 1) a partition of the rows, that is, an element from S_1, 2) a partition of the columns, again, an element from S_1, and 3) a mapping between the L sets of rows and L sets of columns, i.e., an element from S_2. The encoding and decoding maps will be clear from the encoding and decoding of the mappings F_1 and F_2.

The number of bits that can be stored by this construction is $N = \log(s_1 \cdot s_1 \cdot s_2) = 2\log s_1 + \log s_2$. We approximate this value while taking $\log m! \approx m \log m$ for m large enough. Therefore,

$$\begin{aligned} N &= 2\log s_1 + \log s_2 = 2\log\left(\frac{n!}{(\frac{n}{L})!^L \cdot L!}\right) + \log(L!) \\ &= 2\log n! - 2L\log\left(\left(\frac{n}{L}\right)!\right) - \log(L!) \\ &\approx 2n\log n - 2L\cdot\frac{n}{L}\log\left(\frac{n}{L}\right) - L\log(L) \\ &= 2n\log n - 2n\log\left(\frac{n}{L}\right) - L\log(L) = (2n-L)\log(L). \end{aligned}$$

If we choose $L = \frac{n}{\log n}$ we get

$$N = (2n - \frac{n}{\log n})\log\left(\frac{n}{\log n}\right),$$

and for n large enough

$$\lim_{n\to\infty}\frac{N}{2n\log n} = 1.$$

Thus this mapping will be asymptotically optimal. We finally note that the functions F_1 and F_2 have efficient implementations. This can be done by different methods for the enumerations of permutations and multipermutations; see for example [2] and chapter 5.1 in [7].

V. Representations trading off sneak paths and power consumption

One way to eliminate memristor sneak paths without resorting to any information-theoretic tools is by electrically grounding all rows except the one being read [12]. The problem with grounding all other rows is that it significantly increases the power consumption of the read operation due to lower equivalent resistance through which flows the measurement current. Without information theoretic tools, this suggests a tradeoff between power consumption (from grounded rows) and read errors (from sneak paths). Alternatively, we propose to replace the power-correctness tradeoff with a power-density one, by combining partial grounding with sneak-path constraint codes. The key idea is to specify how many of the rows will be grounded in a read operation, and ensure that no sneak paths exist in the part of the array remaining "active" in the non-grounded rows. By doing that, we can control the power consumption of the read operation while guaranteeing read accuracy. Since many of the cells will be deactivated in grounded rows, maintaining sneak-path-free reads will be possible with good storage rates. There are several ways to obtain sneak-path-free sub-arrays, each resulting in an interesting information-theoretic problem.

A. Grounding based upon fixed subsets

In this section we study the capacity assuming the array rows are divided into disjoint subsets, and grounding all rows outside the subset of the read row. We will show that when the subset size is a constant, the capacity no longer goes to zero as in the full array.

Assume the array size is $m \times n$ and let b be some positive integer which is a divisor of m. The m rows are divided into m/b disjoint subsets of consecutive rows. Then, any of the m/b sub-arrays of size $b \times n$ is required to satisfy the isolated zero rectangle constraint. Since all these sub-arrays are disjoint and thus independent, we conclude that the number of arrays will be $N(b,n)^{m/b}$. Let us define the capacity of this constraint by $\mathbb{C}_1(b)$. Then, we get

$$\mathbb{C}_1(b) = \lim_{m,n\to\infty}\frac{\log\left(N(b,n)^{m/b}\right)}{mn} = \lim_{n\to\infty}\frac{\log\left(N(b,n)\right)}{bn}.$$

We first prove lower and upper bound on the value of $N(b,n)$.

Lemma 12. *For any $b = o(n)$ and n large enough the following holds*

$$(b+1)^n - b^{n+1} \leqslant N(b,n) \leqslant (b+1)!S(n+1,b+1).$$

Proof: According to Lemma 11

$$\begin{aligned} N(b,n) &= \sum_{\ell=0}^{b} S(b+1,\ell+1)S(n+1,\ell+1)\ell! \\ &\geqslant S(b+1,b+1)S(n+1,b+1)b! \end{aligned}$$

Since $S(b+1,b+1)=1$ and

$$\begin{aligned} S(n+1,b+1) &= \frac{1}{(b+1)!}\sum_{i=0}^{b+1}(-1)^i\binom{b+1}{i}(b+1-i)^{n+1} \\ &\geqslant \frac{(b+1)^{n+1}-(b+1)b^{n+1}}{(b+1)!} = \frac{(b+1)^n-b^{n+1}}{b!}, \end{aligned}$$

we get

$$N(b,n) \geqslant \frac{(b+1)^n-b^{n+1}}{b!}b! = (b+1)^n - b^{n+1}.$$

On the other hand, if $b=o(n)$ let us show that for n large enough the following holds for every $0 \leqslant \ell < b$

$$S(b+1,\ell+1)S(n+1,\ell+1)\ell! \leqslant S(n+1,b+1)b!.$$

First note that

$$\begin{aligned} &S(b+1,\ell+1)S(n+1,\ell+1)\ell! \\ &\leqslant \frac{(\ell+1)^{b+1}}{(\ell+1)!}\cdot\frac{(\ell+1)^{n+1}}{(\ell+1)!}\cdot\ell! = \frac{(\ell+1)^{n+b}}{\ell!}, \end{aligned}$$

and we saw that $S(n+1,b+1) \geqslant \frac{(b+1)^n-b^{n+1}}{b!}$. Now,

$$\begin{aligned} &\lim_{n\to\infty}\frac{\frac{(\ell+1)^{n+b}}{\ell!}}{\frac{(b+1)^n-b^{n+1}}{b!}} = \lim_{n\to\infty}\frac{b!(\ell+1)^{n+b}}{\ell!\left((b+1)^n-b^{n+1}\right)} \\ &\leqslant \lim_{n\to\infty}\frac{b!(\ell+1)^{n+b}}{(b+1)^n-b^{n+1}} \\ &= \lim_{n\to\infty}\frac{b!(\ell+1)^{n+b}}{(b+1)^n}\cdot\lim_{n\to\infty}\frac{(b+1)^n}{(b+1)^n-b^{n+1}}. \end{aligned}$$

Let us evaluate every term independently under the assumption that $b=o(n)$.

$$\begin{aligned} &\lim_{n\to\infty}\frac{(b+1)^n}{(b+1)^n-b^{n+1}} = \lim_{n\to\infty}\frac{1}{1-b\cdot\left(\frac{b}{b+1}\right)^n} \\ &\lim_{n\to\infty}\frac{1}{1-b\cdot\left(\left(1-\frac{1}{b+1}\right)^b\right)^{n/b}} \\ &= \lim_{n\to\infty}\frac{1}{1-b\cdot e^{-n/b}} = 1. \end{aligned}$$

Similarly,

$$\begin{aligned} &\lim_{n\to\infty}\frac{b!(\ell+1)^{n+b}}{(b+1)^n} \leqslant \lim_{n\to\infty}\frac{(\ell+1)^{n+b}}{(b+1)^{n-b}} \\ &\leqslant \lim_{n\to\infty}\frac{(\ell+1)^{n-b}}{(b+1)^{n-b}}\cdot b^{2b} = \lim_{n\to\infty}\left(\frac{\ell+1}{b+1}\right)^{n-b}\cdot b^{2b} \\ &= \lim_{n\to\infty}\left(1-\frac{b-\ell}{b+1}\right)^{n-b}\cdot b^{2b} \leqslant \lim_{n\to\infty}\left(1-\frac{1}{b+1}\right)^{n-b}\cdot b^{2b} \\ &\lim_{n\to\infty}\left(\left(1-\frac{1}{b+1}\right)^b\right)^{\frac{n}{b}-1}\cdot b^{2b} \\ &= \lim_{n\to\infty} e^{-\frac{n}{b}+1}\cdot b^{2b} = 0 \end{aligned}$$

Therefore, we get that for n large enough

$$\begin{aligned} N(b,n) &= \sum_{\ell=0}^{b} S(b+1,\ell+1)S(n,\ell+1)\ell! \\ &\leqslant (b+1)S(n+1,b+1)b! = (b+1)!S(n+1,b+1). \end{aligned}$$

■

Now we are ready to calculate the capacity $\mathbb{C}_1(b)$ for fixed values of b.

Lemma 13. *For any fixed b, $\mathbb{C}_1(b) = \frac{\log(b+1)}{b}$.*

Proof: According to Lemma 12

$$\begin{aligned} \lim_{n\to\infty} \frac{\log(N(b,n))}{bn} &\geqslant \lim_{n\to\infty} \frac{\log\left((b+1)^n - b^{n+1}\right)}{bn} \\ &= \lim_{n\to\infty} \frac{\log\left((b+1)^n\left(1-b\left(\frac{b}{b+1}\right)^n\right)\right)}{bn} \\ &= \frac{\log(b+1)}{b} + \lim_{n\to\infty} \frac{\log\left(1-b\left(\frac{b}{b+1}\right)^n\right)}{bn} = \frac{\log(b+1)}{b}. \end{aligned}$$

To prove the opposite inequality, again by Lemma 12 we get

$$\begin{aligned} \lim_{n\to\infty} \frac{\log(N(b,n))}{bn} &\leqslant \lim_{n\to\infty} \frac{\log((b+1)!S(n+1,b+1))}{bn} \\ &\leqslant \lim_{n\to\infty} \frac{\log((b+1)^{n+1})}{bn} = \frac{\log(b+1)}{b}. \end{aligned}$$

■

Finally, we note that very similarly the following property holds for $b = o(n)$,

$$\lim_{\substack{b\to\infty \\ b=o(n)}} \frac{\mathbb{C}_1(b)}{\frac{\log(b+1)}{b}} = 1.$$

B. Grounding sets based upon the read row

In this section we study the capacity assuming all rows are grounded outside a subset of rows which depends upon the read row. In particular, we study the case where all rows outside a subset of odd size *centered* at the read row are grounded. It turns out that a sufficient (but not necessary) condition to have a sneak-path free array in this case is that each column satisfies some run-length limited (RLL) [3] constraint, which depends on the number of ungrounded rows.

Under this model, we say that there is a ***b*-centered-path**, where b is odd, affecting the cell in position (i,j) if $a_{i,j} = 0$ and there is a path as defined in Definition 1 which can be confined between the $(i - \frac{b-1}{2})$-th row and the $(i + \frac{b-1}{2})$-th row. That is, for some $k \geqslant 1$, there exist $2k$ positive integers $\max\{i - \frac{b-1}{2}, 1\} \leqslant r_1, \ldots, r_k \leqslant \min\{i + \frac{b-1}{2}, m\}$, $1 \leqslant c_1, \ldots, c_k \leqslant n$ such that

$$a_{i,c_1} = a_{r_1,c_1} = a_{r_1,c_2} = \cdots = a_{r_{k-1},c_k} = a_{r_k,c_k} = a_{r_k,j} = 1.$$

Thus, we say that an array satisfies the ***b*-centered-path constraint** if it has no b-centered-paths.

For any odd $b \geqslant 1$, we denote by $N_2(m,n;b)$ the number of arrays that satisfy the b-centered-path constraint and we denote the capacity of this constraint by $\mathbb{C}_2(b)$, so

$$\mathbb{C}_2(b) = \lim_{m,n\to\infty} \frac{\log\left(N_2(m,n;b)\right)}{mn}.$$

Furthermore, we say that an array has a ***b*-isolated zero rectangle** if there are four positive integers $i_1 \neq i_2$, $j_1 \neq j_2$, and $|i_2 - i_1| \leqslant b-1$, such that $a_{i_1,j_1} + a_{i_1,j_2} + a_{i_2,j_1} + a_{i_2,j_2} = 3$. An array A satisfies the ***b*-isolated zero rectangle constraint** if it has no b-isolated zero rectangles and then it is called a ***b*-isolated zero rectangle free array**. The b-isolated zero rectangle constraint is the same as the isolated zero rectangle constraint from Definition 7 when applied to sub-arrays of A with b rows.

It is a matter of simple observation to get to the following correspondence between the centered path constraint and the isolated zero rectangle constraint:

Lemma 14. *The b-centered-path constraint and the $\frac{b+1}{2}$-isolated zero rectangle constraint are equivalent.*

To proceed, let us recall the one-dimensional RLL constraint. We say that a binary sequence satisfies the (d, k) RLL constraint if the number of zeros between every two consecutive ones is at least d and at most k. The capacity of the one dimensional (d, k) RLL constraint is denoted by $\mathbb{C}_{d,k}$. Next, we show that the capacity of the $(\frac{b-1}{2}, \infty)$ RLL constraint is a lower bound on $\mathbb{C}_2(b)$.

Lemma 15. *For any odd b, $\mathbb{C}_2(b) \geqslant \mathbb{C}_{\frac{b-1}{2},\infty}$.*

Proof: This result follows from the observation that if every column satisfies the $(\frac{b-1}{2}, \infty)$ RLL constraint then necessarily there are no pairs of ones in the same column at distance less than $\frac{b-1}{2}$ rows. In particular, there is no rectangle confined to $\frac{b+1}{2}$ rows with an isolated zero. ■

The reverse inequality on $\mathbb{C}_2(b)$ is proved in the next lemma.

Lemma 16. *For any odd b, $\mathbb{C}_2(b) \leqslant \mathbb{C}_{\frac{b-1}{2},\infty}$.*

Proof: Let $B_{m,n}$ be the number of $m \times n$ arrays where every column satisfies the $(\frac{b-1}{2}, \infty)$ RLL constraint. Let A be a b-centered-path-free array. According to Lemma 14, A is a $(\frac{b+1}{2})$-isolated zero-rectangle free array. Thus, as in the proof of Lemma 9, in every $\frac{b+1}{2}$ consecutive rows of A, every two rows are either the same or their ones are located at disjoint locations.

For a positive integer divisor d of m, we define a mapping $F_d : \{0,1\}^{m\times n} \to \{0,1\}^{m\times n}$, which transforms an array A to $F_d(A)$ as follows. Starting with the first d rows of A, if there are identical rows among these d rows, then the first row remains the same and the subsequent identical rows are replaced with all-zero rows. Then the same operation is performed on the new array with the next window of d rows, between the second and $(d+1)$-th row, and so on until reaching the last window consisting of the last d rows.

Let A' be the array resulting under this mapping with $d = \frac{b+1}{2}$ on the array A, that is $A' = F_{\frac{b+1}{2}}(A)$. The array A' holds the property that every column satisfies the $(\frac{b-1}{2}, \infty)$ RLL constraint.

We note that this mapping is many to one, as there can be several b-centered-path-free arrays A which will be mapped to the same array A'. Given an array A' we can bound the number of arrays A that are mapped to it. Assuming the array A' has x zero rows, then each row can be identical to any of the $\frac{b-1}{2}$ rows above it, or originally all-zero. Since there are m rows in the array, we can use a loose upper bound here (which will be sufficient for our goal), and say that at most m^m arrays will be mapped to the array A'. Therefore, we get the following relation

$$N_2(m, n; b) \leqslant m^m \cdot B_{m,n}.$$

Now we conclude that

$$\begin{aligned}
\mathbb{C}_2(b) &= \lim_{m,n\to\infty} \frac{\log N_2(m,n;b)}{mn} \leqslant \lim_{m,n\to\infty} \frac{\log(m^m \cdot B_{m,n})}{mn} \\
&= \lim_{m,n\to\infty} \frac{m\log m + \log B_{m,n}}{mn} \\
&= \lim_{m,n\to\infty} \frac{\log B_{m,n}}{mn} = \mathbb{C}_{\frac{b-1}{2},\infty}.
\end{aligned}$$

■

From Lemma 15 and Lemma 16, we get that

$$\mathbb{C}_2(b) = \mathbb{C}_{\frac{b-1}{2},\infty}.$$

It turns out that the symmetric grounding set method is better than the one based upon fixed subsets. In other words, we can prove the inequality $\mathbb{C}_2(b) \geqslant \mathbb{C}_1(b)$.

Theorem 17. *For all odd values of b, the following holds*

$$\mathbb{C}_2(b) \geqslant \mathbb{C}_1(b).$$

Proof: We need to show that $\mathbb{C}_{\frac{b-1}{2},\infty} \geqslant \frac{\log(b+1)}{b}$ for odd values of b. For $b < 250$ we numerically calculated the values of $\mathbb{C}_1(b)$ and $\mathbb{C}_2(b)$ to verify this inequality. For $b > 250$, we use a property from Problem 3.3 in [10] claiming that for every positive integer m, $\mathbb{C}_{\frac{b-1}{2},\infty} \geqslant \frac{\log(m+1)}{\frac{b-1}{2}+m}$. In particular, we choose $m = \lfloor (b+2)/4 \rfloor$ and get that

$$\mathbb{C}_{\frac{b-1}{2},\infty} \geqslant \frac{\log(\lfloor (b+2)/4 \rfloor + 1)}{\frac{b-1}{2} + \lfloor (b+2)/4 \rfloor} \geqslant \frac{\log(\lfloor (b+2)/4 \rfloor + 1)}{3b/4}.$$

Thus, it is enough to show that

$$\log(\lfloor (b+2)/4 \rfloor + 1) \geqslant (3/4) \cdot (\log(b+1),$$

or

$$(b+2)/4 \geqslant (b+1)^{3/4},$$

which holds for $b > 250$. ■

To conclude, we compare between the numerical values of the capacities of the two approaches we introduced here for $b \leqslant 11$.

b	$\mathbb{C}_1(b) = \frac{\log(b+1)}{b}$	$\mathbb{C}_2(b) = \mathbb{C}_{\frac{b-1}{2},\infty}$
2	0.792	-
3	0.667	0.694
4	0.580	-
5	0.517	0.551
6	0.468	-
7	0.423	0.465
8	0.396	-
9	0.369	0.406
10	0.346	-
11	0.326	0.362

VI. Conclusion

This work offers a detailed study of the memristor sneak-path problem through an information theoretic lens. The electric interference due to sneak paths was formulated in terms of abstract terms like *channel*, *constraint*, *capacity* etc. This abstract view allows future work to construct new codes, and extend the results to additional sneak-path models motivated by real memristor devices.

VII. Acknowledgments

The authors thank Ron M. Roth for pointing reference [13] to their attention, and an anonymous reviewer who found a flaw in the original proof of Lemma 16.

References

[1] Y. Cassuto, S. Kvatinsky, and E. Yaakobi. Sneak-path constraints in memristor crossbar arrays. In *IEEE International Symposium on Information Theory*, ISIT 2013.
[2] T.M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, January 1973.
[3] K.S. Immink. *Coding techniques for digital recorders*. Prentice-Hall, College Div., 1991.
[4] C.-M. Jung, J.-M. Choi, and K.-S. Min, "Two-step write scheme for reducing sneak-path leakage in complementary memristor array," *Nanotechnology, IEEE Transactions on*, vol. 11, pp. 611–618, May 2012.
[5] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of memristor-based memories," in *VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on*, pp. 386–391, Jan 2013.
[6] K.-W. Kim, S. Gaba, D. Wheeler, J. Cruz-Albrecht, H. Tahir, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano Letters*, vol. 12, no. 1, pp. 389–395, 2012.
[7] D.E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, 1998.
[8] J. Liang and H.-S. Wong, "Cross-point memory array without cell selectors – device characteristics and data storage pattern dependencies," *Electron Devices, IEEE Transactions on*, vol. 57, pp. 2531–2538, Oct 2010.
[9] E. Ordentlich and R.M. Roth. Low complexity two-dimensional weight-constrained codes. *IEEE Transactions on Information Theory*, 58(6):3892–3899, 2012.
[10] R.M. Roth, *Coding for Storage Systems*. Technion Lecture Notes.
[11] J. Shin, I. Kim, K. Biju, M. Jo, J. Park, J. Lee, S. Jung, W. Lee, S. Kim, S. Park, and H. Hwang, "Tio2-based metal-insulator-metal selection device for bipolar resistive random access memory cross-point application," *Journal of Applied Physics*, vol. 109, no. 3, 2011.
[12] S. Shin, K. Kim, and S. Kang. Analysis of passive memristive devices array: data-dependent statistical model and self-adaptable sense resistance for RRAMs. *Proceedings of the IEEE*, 100(6):2021–2032, 2012.

[13] P.P. Sotiriadis, "Information capacity of nanowire crossbar switching networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 3019–3032, July 2006.
[14] D. Strukov, G. Snider, D. Stewart, and R.S. Williams, "The missing memristor found," *Nature*, vol.,453, pp. 80–83, May 2008.
[15] J. van Lint and R. Wilson, *A Course in Combinatorics, second edition*. Cambridge UK: Cambridge University Press, 2001.
[16] P.O. Vontobel, W. Robinett, P.J. Kuekes, D.R. Stewart, J. Straznicky, and R.S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, October 2009.
[17] J. Yang, M.-X. Zhang, M. Pickett, F. Miao, J. Strachan, W.-D. Li, W. Yi, D. Ohlberg, B. Choi, W. Wu, J. Nickel, G. Medeiros-Ribeiro, and R.S. Williams, "Engineering nonlinearity into memristors for passive crossbar applications," *Applied Physics Letters*, vol. 100, no. 11, 2012.
[18] M. Zidan, H. H. Fahmy, M. Hussain, and K. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176 – 183, 2013.