ISIT'2013 1569719615

Sneak-Path Constraints in Memristor Crossbar Arrays

Yuval Cassuto[*], **Shahar Kvatinsky**[*], and **Eitan Yaakobi**[†]
[*]Electrical Engineering Department, Technion – Israel Institute of Technology
[†]Electrical Engineering Department, California Institute of Technology
ycassuto@ee.technion.ac.il, skva@tx.technion.ac.il, yaakobi@caltech.edu

***Abstract*—In a memristor crossbar 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 crossbars arrays, in which current can sneak through other cells, resulting in reading a wrong state of the memristor. Our main contributions are 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 calculate the resulting capacity.**

I. Introduction

The memristor technology [8] 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* [6]. 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 [4]. Here we address another very significant data-dependent phenomenon called *sneak paths* [6], causing the read correctness to depend on the array content.

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

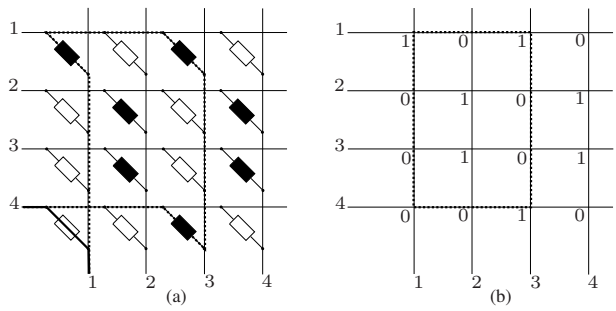

Fig. 1. (a) A memristor array as an array of programmed resistors – white: high resistance, black: low resistance. (b) The corresponding logical values of the memristor array.

a resistor that can be in either the high-resistance state (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 first such an attempt was already presented in [9] by forcing the number of zeros and ones in every row and column to be the same. While this solution can reduce the sneak path effect, it does not eliminate it completely, as we seek to study in this work. Considering a full $m \times n$ memristor array, exact counting of the number of array assignments that guarantee sneak-path free readout was found by Sotiriadis in [7]. Our contributions start in Section II by offering an alternative characterization of sneak-path free arrays, which enables the main results of the paper. In Section III we give a capacity-achieving efficient encoder that maps unrestricted information to sneak-path free arrays. In Section IV, 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 [7]). To the second grounding method we match a parameterized 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. Characterization of the Sneak Paths

Let us first define formally and mathematically the sneak-path constraint.

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*

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

An array A *satisfies the* ***sneak-path constraint*** *if it has no sneak paths and then it is called a* ***sneak-path-free array****.*

The sneak-path problem was already introduced and studied in [7] 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 [7] 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 [7].

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 in a memristor array can be perfectly characterized by an abstract constraint, which we call the *isolated zero-rectangle constraint*.

Definition 2. *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 vertices is zero.

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 if and only if it has no isolated zero-rectangles. Note that sneak paths may be of any odd length 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 3. *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 $(i,c_1),(r_k,c_1),(r_k,j)$ of length 3 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 4. *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 [7]. For the sake of completeness, we provide a proof of the result that uses the isolated zero-rectangle constraint and its characterization in Lemma 4.

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.

Lemma 5. *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 4, 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 [7] can similarly be obtained using the isolated zero-rectangle constraint (proof omitted).

Lemma 6. *The value $N(m,n)$ can be expressed by*

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

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 [7]. Thus, under these conditions it is derived that

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

which implies a 0 asymptotic storage capacity. In fact, we can show that this behavior holds for any 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 IV.

III. 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 [7], a low complexity and very efficient mapping was presented, however the number of information bits which 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 to arbitrary n). However, according to the derivations in [7], the number of bits that can be represented by all sneak-path free arrays is roughly $2n\log n$. Thus, the mapping in [7] 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 floors and ceilings, 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. 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 5, which uses the if and only if condition in Lemma 4. We encode only arrays where the rows resp. 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,

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

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 and briefly note that the functions F_1 and F_2 have efficient implementations. This can be done by different methods, see for example [1] and chapter 5.1 in [3].

IV. Representations trading off sneak paths and power consumption

One way to eliminate memristor sneak paths without resorting to any information-theoretic tools is by grounding all rows except the one being read. The problem with grounding all other rows is that it increases the power consumption of the read operation due to lower equivalent resistance through which flows the measurement current. Without information theoretic data representations, this suggests a tradeoff between power consumption (from grounded rows) and read inaccuracy/incorrectness (from sneak paths). Alternatively, we propose to replace the power-correctness tradeoff with a power-density one, by employing an information-theoretic data representation. 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 many 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 does no longer go 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 subarrays of size $b \times n$ will have to satisfy the isolated zero-rectangle constraint. Since all these subarrays 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

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

The proof of the next lemma is omitted due to the lack of space.

Lemma 7. *For any $b = o(n)$ and n large enough the following holds*

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

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

Lemma 8. *For any fixed b, $\mathbb{C}_1(b) = \frac{\log(b+1)}{b}$.*

Proof: According to Lemma 7

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

To prove the opposite inequality, again by Lemma 7 we get

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

■

Finally, we note that in a very similar way it is possible to show that if $b = o(n)$ then

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

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 these case is that each column satisfies some run-length limited (RLL) [2] constraint, which depends on the number of ungrounded rows.

Under this model, we say that there is a ***b-sneak-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-sneak path constraint*** if it has no b-sneak paths.

For any odd $b \geqslant 1$, we denote by $N_2(m,n;b)$ the number of arrays that satisfy the b-sneak-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***.

Before we 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 $\mathbb{C}_{d,k}$. As was shown in Theorem 3, it is possible to show the following.

Lemma 9. *The b-sneak path constraint and the $\frac{b+1}{2}$-isolated zero-rectangle constraint are equivalent.*

Lemma 10. *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 11. *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-sneak-path-free array. According to Lemma 9, A is a $(\frac{b+1}{2})$-isolated zero-rectangle free array. Thus, as in the proof of Lemma 4, 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.

Let us define a mapping $F_d : \{0,1\}^{m\times n} \to \{0,1\}^{m\times n}$, which transfers an array A to $F_d(A)$ as follows. Starting 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-sneak-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 10 and Lemma 11, 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 $\mathbb{C}_2(b) \geqslant \mathbb{C}_1(b)$. This can be done by using the property from Problem 3.3 in [5] that for every positive integers d, m, $\mathbb{C}_{d,\infty} \geqslant \frac{\log(m+1)}{d+m}$.

To conclude, we compare between the capacities of the three approaches we introduced here

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

V. Acknowledgments

This work was partially supported by the ISEF Fellowship, an Intel ICRI-CI award, and an EU Marie Curie CIG grant. The authors thank Ron M. Roth for pointing reference [7] to their attention, and an anonymous reviewer who found a flaw in the original proof of Lemma 11.

References

[1] T.M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, January 1973.
[2] K.S. Immink. *Coding techniques for digital recorders*. Prentice-Hall, College Div., 1991.
[3] D.E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, 1998.
[4] E. Ordentlich and R.M. Roth. Low complexity two-dimensional weight-constrained codes. *IEEE Transactions on Information Theory*, 58(6):3892–3899, 2012.
[5] R.M. Roth, *Coding for Storage Systems*. Technion Lecture Notes.
[6] 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.
[7] P.P. Sotiriadis, "Information capacity of nanowire crossbar switching networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 3019–3032, July 2006.
[8] D. Strukov, G. Snider, D. Stewart, and S. Williams, "The missing memristor found," *Nature*, vol.,453, pp. 80–83, May 2008.
[9] P.O. Vontobel, W. Robinett, P.J. Kuekes, D.R. Stewart, J. Straznicky, and S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, October 2009.