On the Channel Induced by Sneak-Path Errors in Memristor Arrays

Yuval Cassuto
Dept. of Electrical Engineering.
Technion – Israel Institute of Tech.
Haifa, 3200000, Israel
Email: ycassuto@ee.technion.ac.il

Shahar Kvatinsky
Dept. of Electrical Engineering.
Technion – Israel Institute of Tech.
Haifa, 3200000, Israel
Email: skva@tx.technion.ac.il

Eitan Yaakobi
Dept. of Computer Science.
Technion – Israel Institute of Tech.
Haifa, 3200000, Israel
Email: yaakobi@cs.technion.ac.il

***Abstract*—Memristors, also known as resistive RAMs, are very promising non-volatile media that can be packed in unprecedented density. However, the crossbar layout by which this high density is achieved entails major challenges arising from cell-to-cell interference. In particular, cell readout is affected by sneak paths, which are electric paths passing through other crossbar cells and affecting the outcome of the read operation. The existence of sneak paths and their severity depends upon the current bit assignment stored in the array. In this paper we study sneak-path errors by modeling the array as a single-parameter information theoretic channel. We calculate this parameter in closed form as a function of the array dimensions and the bias between 0s and 1s in the written bits. We extend this result to the case where error occurs only when at least L sneak-paths exist, and also examine the correlation between sneak-path errors in different cells within the array. The channel capacity is calculated in a flavor similar to the capacity of the Z channel, only with transition probability that depends on the array-bit distribution.**

***Index Terms*—Memristors, resistive memories, storage channels, inter-cell interference, coding for storage.**

I. Introduction

The memristor technology [6] allows packing storage cells in unprecedented density, over a simple crossbar structure. The blessing of high storage density and architectural simplicity comes with a major caveat: *severe cell-to-cell interference* [4]. The primary manifestation of cell-to-cell interference in memristor arrays is the *sneak-path* problem [4], causing the read outcome of a given cell to depend on the contents of other cells within the array.

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 (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

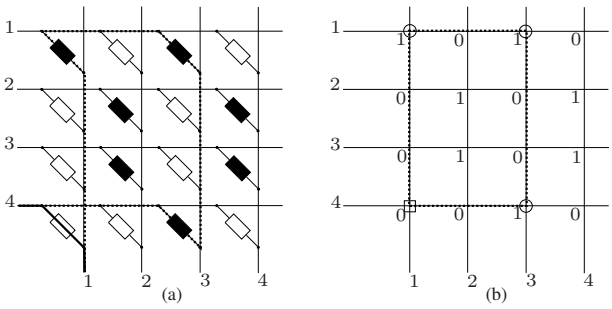

Fig. 1. (a) A memristor array as an array of programmed resistors – white: high resistance, black: low resistance. The high-resistance cell at location $(4,1)$ has a sneak-path in parallel (plotted dashed), causing it to be read as low-resistance. (b) The corresponding logical values of the memristor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

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 characterize sneak-path errors as an information-theoretic channel, serving as the foundation for a comprehensive coding theoretic treatment that will enable reliable memristor storage. Harnessing information-theoretic techniques for sneak-path mitigation has already proven effective in prior work. In [7] it was suggested to force the number of zeros and ones in every row and column to be the same (see also [3]), aiming at reducing the incidence of sneak paths. Completely eliminating sneak paths by constraint-coding of the array bits was studied in [1], [5]. The drawback of the elimination approach is that it imposes restrictions on the array content that are stronger than what is practically needed, thereby result-

978-1-4799-4665-5/14/$31.00 ©2014 IEEE

ing in a high rate loss. Here we adopt a more lenient approach where sneak paths do exist, and treating the resulting error-prone readout as an information-theoretic channel.

The information-theoretic treatment of sneak-path errors is explored in Section II. We first formulate 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. We then extend this calculation to the case where an error occurs only if at least some number L of sneak paths affect the cell. Section II continues with an asymptotic analysis showing that an *infinite* array has zero capacity, and then with a derivation of a lower bound on the capacity of *semi-infinite* arrays. Section III then studies the joint error probability for cells within the same array, in light of the fact that bit errors due to sneak paths are not independent.

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, 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 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

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

Note that 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 (e.g. 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.

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.

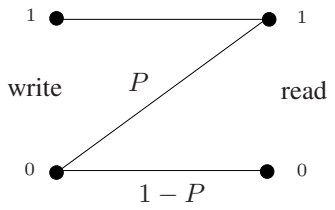

Fig. 2. Sneak-path read errors as a Z channel. A written 0 is read as 0 if no sneak path affects it and as 1 otherwise. A written 1 is always read correctly.

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 the expression in* (2).

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

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.

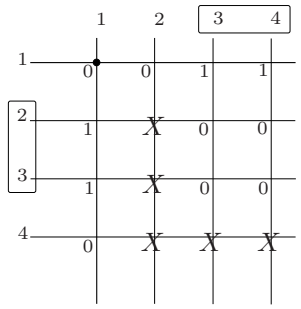

Fig. 3. An example of an array with no sneak path for cell $(1,1)$. Given 0/1 assignments in row $i=1$ and column $j=1$, the intersections of the 1-rows with the 1-columns must be set to zeros. In the rest of the array locations, the value can be arbitrary, marked with X.

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). Since sneak-path errors are caused by cells in 1 state, the error probability is monotone increasing in q. It is also clear that the error probability grows as we increase the array size, as a larger array contains more potential sneak paths affecting a given cell. These can be seen in Figure 4 plotting the error probability as a function of q for two array sizes.

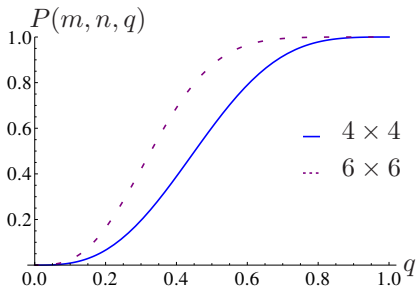

Fig. 4. Sneak-path error probability as a function of the bias q. Solid line for a 4×4 array; dashed line for a 6×6 array.

For a given array area (number of cells), square arrays suffer from higher sneak-path error probability compared to non-square arrays. This fact is depicted in Figure 5 comparing the error probabilities of a 4×4 (solid) and a 2×8 (dashed) arrays.

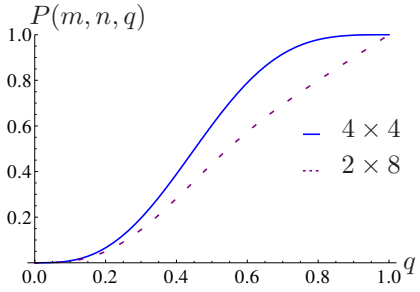

Fig. 5. Comparing sneak-path error probability for two array sizes with the same area. Solid line for a 4×4 array; dashed line for a 2×8 array.

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. The dependence between sneak-path errors within the array is discussed in more detail in Section III.

It may be the case in practice that one sneak path is not sufficient to cause a bit error. Rather, the 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.

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

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

We now want to analyze the sneak-path transition probability when the array dimensions m and n tend to infinity. In particular, we want to know whether reliable readout is possible in the limit of large memory arrays. The following theorem answers this question to the negative, i.e., that one cannot store bits reliably on arrays with both m and n tending to infinity.

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 $P(m,n,q)$, now proving 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

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

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 0s required to not have sneak path when the weight of $\boldsymbol{u}$ is 1. Simplification of (4) yields equation (5), which proves transition probability tending to 1 given $\boldsymbol{u}$. 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 sneak-path effect to a small subset of rows is done with a simple technique called *grounding* [2], which incurs some implementation cost in the form of higher power consumption. 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

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

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

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

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.

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.

III. SECOND MOMENT OF SNEAK-PATH ERRORS

The probability to have a sneak-path error is shown in the previous section to be independent of the array location, only depending on the array parameters. However, it is not hard to see that a sneak-path error in one array location is *not* independent from a sneak-path error in another location given the array parameters (m,n,q). For example, a sneak path affecting an array location increases the likelihood of a sneak path affecting another location in the same row or column, as sneak paths affecting locations in the same row/column have many cells in common. One option to mitigate this dependence is to perform sneak-path error-correction coding *across arrays* (i.e., each bit in a codeword is assigned to a different array with the same parameters), so that errors within a code block can be assumed i.i.d. However, restricting coding to be performed cross-array has significant practical drawbacks, mainly increased latency due to the need to read multiple arrays before the decoding of a code block can start. So to avoid this, we study in this section the dependence between sneak-path errors in the same array.

A. Joint sneak-path distribution between two cells in the same row/column

Consider an $m \times n$ array with bias q. Denote by $e_{i,j}$ the event that cell (i,j) is affected by a sneak path, and by $\bar{e}_{i,j}$ the complementary event that cell (i,j) is not affected by a sneak path. In the notation of Section II-A we write $\Pr(e_{i,j}) = P(m,n,q)$ and $\Pr(\bar{e}_{i,j}) = 1 - P(m,n,q)$. We now want to examine the probability that *two cells* within the same array column are affected by sneak paths. In other words, we want to calculate the joint probability

$$\Pr(e_{i,j}, e_{i',j}),$$

for some index triple i, i', j. We may similarly be interested in the joint probability $\Pr(\bar{e}_{i,j}, \bar{e}_{i',j})$ (one can be obtained from the other and the individual probability $\Pr(e_{i,j})$). Denote by u the number of 1s in column j, by v the number of 1s in row i, and by v' the number of 1s in row i'. In addition, denote by σ the number of column indices in which both rows i, i' have 1s. Hence $\sigma \leqslant \min\{v, v'\}$.

Theorem 7. *The joint probability $\Pr(\bar{e}_{i,j}, \bar{e}_{i',j})$ in an $m \times n$ array with parameter q equals the expression in* (8).

Proof: Given u, v, v' and the overlap σ, avoiding sneak paths for both cells (i,j) and (i',j) requires 0 assignments to the cells that intersect the u row indices where column j has 1s, with the union of the v and v' column indices where row i or i' (or both) have 1s. The size of this union is $v + v' - \sigma$, and this adds the right term $u(v+v'-\sigma)$ to the exponent of $(1-q)$ in (8). The remaining terms in the exponent of $(1-q)$, as well as the terms in the exponent of q, follow from the assignment of 0s and 1s to column j and rows i, i' prescribed by u, v, v'. The latter two binomial coefficients in (8) count the number of assignments of v' column indices having overlap σ with a given set of v column indices. ∎

$$\Pr(\bar{e}_{i,j},\bar{e}_{i',j}) = \sum_{u=0}^{m-2}\sum_{v=0}^{n-1}\sum_{v'=0}^{n-1}\sum_{\sigma=0}^{n-1}\binom{m-2}{u}\binom{n-1}{v}\binom{v}{\sigma}\binom{n-1-v}{v'-\sigma} q^{u+v+v'}(1-q)^{m-2-u+n-1-v+n-1-v'+u(v+v'-\sigma)}. \tag{8}$$

Thanks to symmetry between rows and columns[1] the formula (8) can be used also for the case of two cells in the same row.

We now use Theorem 7 to compare the joint error probability of two same-column cells to the probability of double bit error assuming independent errors. Intuitively, since two cells in the same column share the same number u of 1s in the column, we expect a higher likelihood of both erring together (when u is high), and also a higher likelihood of both not erring together (when u is low). We validate this intuition quantitatively in the following figures. Figure 6 compares the probability $\Pr(\bar{e}_{i,j},\bar{e}_{i',j})$ to the square of the probability $1-P(m,n,q)$ from Section II-A.

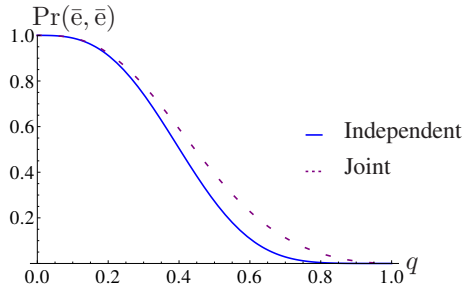

Fig. 6. Comparing the probability $\Pr(\bar{e}_{i,j},\bar{e}_{i',j})$ from Theorem 7 (dashed line) to the square of the individual-cell no-error probability from Theorem 3 (solid line).

Figure 7 compares the probability $\Pr(e_{i,j},e_{i',j})$ (calculated from $\Pr(\bar{e}_{i,j},\bar{e}_{i',j})$ and the individual error probability) to the square of the probability $P(m,n,q)$.

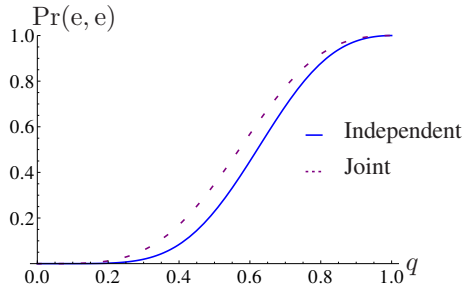

Fig. 7. Comparing the probability $\Pr(e_{i,j},e_{i',j})$ (dashed line) to the square of the individual-cell error probability (solid line).

[1]When we transpose an $m\times n$ array we obtain an $n\times m$ array where rows and columns switch roles.

Figures 6 and 7 show that both the joint no-error probability and the joint error probability of cells in the same column (or row) are higher than those probabilities had errors been independent.

IV. Conclusion

This paper lays the information-theoretic foundations for studying memristor arrays that are prone to read sneak paths. The errors due to sneak paths are characterized as a read channel whose parameter depends on the array dimensions and the $0/1$ bias of the stored bits. This characterization opens the way for extensive coding theoretic treatment of sneak-path errors that will enable reliable data storage on memristors arrays. In the present paper we considered a basic sneak-path model for which error rates get high even for small-size arrays. It is likely that a real memristor array will suit a refined model with lower absolute error rates (e.g. through better physical isolation between cells), but for which information theoretic treatment will follow similar lines.

V. Acknowledgments

This work 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.

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] A. Chen. "Accessibility of nano-crossbar arrays of resistive switching devices," In *IEEE 11th International Conference on Nanotechnology*, 2011.

[3] E. Ordentlich and R.M. Roth. "Low complexity two-dimensional weight-constrained codes," *IEEE Transactions on Information Theory*, 58(6):3892–3899, 2012.

[4] 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.

[5] P. P. Sotiriadis, "Information capacity of nanowire crossbar switching networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 3019–3032, July 2006.

[6] D. Strukov, G. Snider, D. Stewart, and S. Williams, "The missing memristor found," *Nature*, vol.,453, pp. 80–83, May 2008.

[7] 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.