Detection and Coding Schemes for Parallel Interference in Resistive Memories

Yuval Ben-Hur and **Yuval Cassuto**
Viterbi Department of Electrical Engineering, Technion – Israel Institute of Technology
yuvalbh@campus.technion.ac.il, ycassuto@ee.technion.ac.il

Abstract—**This paper studies the problem of reliable resistive-memory readout through the rigorous lens of communication theory. The most dominant reliability issue in resistive memory can be modeled as interference of resistances in parallel to a measured resistance. For this special type of interference we develop detection and coding schemes that are shown to effectively mitigate the effects of sneak-path errors. The uniqueness of this study is that the proposed models combine theoretical rigor with practical richness, hence enabling deep contributions to very real problems.**

I. Introduction

The ever-increasing demand for storage capacity drives a constant need to scale the storage density, while maintaining its power efficiency and reliability. As the flash technology for non-volatile memories (NVMs) seems to reach a scaling barrier, recent advancements in the fabrication of resistive devices suggest promising alternative technologies. First to emerge was the *memristor* technology [1], which was later joined by other technologies similarly implementing memory crossbar arrays of resistive cells. The key in all those technologies is that the memory cell is a passive two-terminal device that can be both read and written over the simple crossbar structure. This feature offers a huge density advantage, but at the cost of poor isolation between cells resulting in severe access and reliability issues. Mitigating these issues is a highly motivated objective, given the far-reaching impact resistive arrays can strike on future computing systems. In addition to storing bits, resistive arrays have been shown to be capable of performing logic operations [2]–[5], analog and neural computation [6]–[8], and vector similarity calculations [9]. In all those exciting applications too, we are in need to solve the fundamental issues of the crossbar array.

The importance of enabling reliable resistive arrays did not escape the eye of the research community, which contributed works toward modeling, detection and repair of faults in resistive arrays [10], [11]. The greatest efforts were pointed to solve the most fundamental problem of resistive arrays called *sneak paths*. When a cell in a crossbar array is read, a voltage is applied upon it, and current measurement determines whether it is in a low-resistance state (logical 1) or a high-resistance state (logical 0). Sneak paths are an effect by which in parallel to the desired measurement path, alternative current paths passing through other array cells distort the cell measurement, and result in read errors. The sneak-path problem was addressed by numerous works with different approaches and at various system layers. Alternative memory architectures, which include a modification of the cell technology and/or the entire array structure, have been proposed to decrease or eliminate sneak paths [12]–[14]. Other approaches concentrate on low-level electric analysis, which is meant to clean distorted measurements [15]–[18]. Finally, information-theoretic analysis and mitigation of sneak paths is studied in [19]–[22].

Despite these intensive research efforts over the last few years, sneak paths remain an open problem for designers of next-generation memory architectures. Our aim in this work is to lay down a structured framework for sneak-path mitigation, upon which deep and rigorous theoretical tools can be harnessed to solve the problem. Although related ideas have been considered before [23], [24], this is the first study that casts the fundamental problem of resistive-array readout as a *communication problem*, with sneak paths modeled as a special type of *interference*. To emphasize the fundamentals of the problem, we start the study one step before the complex realities of resistive memory: Section II formulates a *detection problem* where a resistance is measured with a random number of parallel interfering resistances plus an additive noise. For this formulation we develop two detection schemes: an optimal *maximum a posteriori probability (MAP) detector* and a simple *threshold detector*. In Section III we refine the model to fit the scenario of real resistive arrays. Toward that end we analyze the resistances of different *sneak-path types*, and incorporate *cell non-linearity* into the model. In Section IV we move to propose and analyze sneak-path reduction methods. Our main contribution in this section is a *coding scheme on* 2×2 *words* that is shown to reduce sneak-path incidence significantly, while enjoying simple encoding, decoding and analysis. The benefits of the tools developed in the paper are demonstrated in Section V showing simulation results of bit-error rates in various practical scenarios. Finally, we conclude the paper and suggest future work in Section VI.

II. Detection Schemes for Parallel Interference

A. Model formulation

Consider a set of N hypotheses $\{H_i\}_{i=1}^{N}$ that represent N different resistance values $\{R_i\}_{i=1}^{N}$. Assume the cell resistance is measured in parallel to other resistors in the array. We denote the measured resistance by r and the resistance of each parallel resistor by R. The number of such parallel resistors is denoted by L. We consider a situation where the interfering

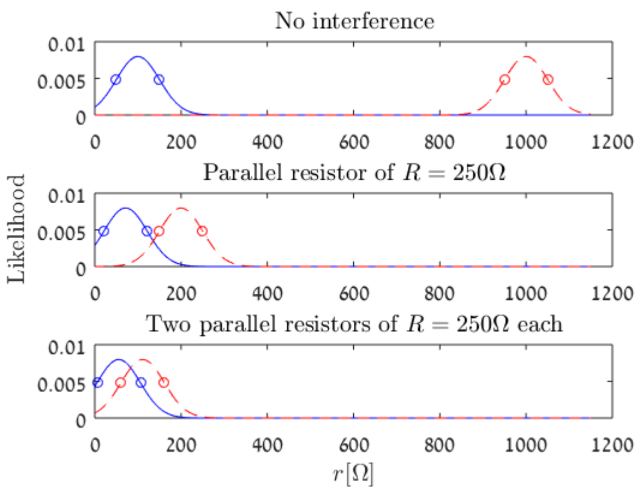


Fig. 1: Plots of the distributions of r for $R = 250\Omega$ with $L = 0, 1$ and 2. The $N = 2$ hypotheses: $H_1 : R_1 = 100\Omega$ and $H_2 : R_2 = 1000\Omega$. The noise standard deviation is $\sigma_\eta = 50\Omega$.

resistance R is known, but L is a data-dependent random variable, which is typically unknown during the measurement. In addition, we consider a Gaussian measurement noise η, which is added to the measured resistance and is independent of its value. Hence, given the i-th hypothesis, from elementary electric theory (resistors in parallel), the measured resistance can be written as the inverse sum of reciprocals of the original resistance and L components with resistance R:

$$r_i = \left(\frac{1}{R_i} + \frac{L}{R}\right)^{-1} + \eta. \tag{1}$$

Interestingly, this simple disturbance of the measured value imposes a fundamental challenge on the task of detecting the hypothesis. The difficulty stems from the unique nature of this disturbance: even relatively low values of L may result in severe drops in the measured resistance. For example, Fig. 1 depicts a scenario where even $L = 1$ shrinks the margin between hypotheses significantly. Detection errors occur when two hypotheses become close enough (due to the values of L and R) so that the additive noise causes cross-over between them. Practically, the detector assumes an upper bound L_{max} on the value of L, beyond which reliable detection is not possible. The value of L_{max} depends on the problem parameters and on the specific detector used.

B. Optimal Detection

An optimal detector for the model presented in (1) can be derived from the Maximum A-Posteriori (MAP) estimator of the hypothesis i, given the measurement r. We work under the assumption that the following distributions are known: the probability mass function of the interference order ($p_L(\cdot)$), the probability density function of the additive noise ($f_\eta(\cdot)$), and the prior probability of each hypothesis (q_i). Let us define the posterior function of the i-th hypothesis as a weighted sum of conditional posteriors

$$\Lambda_i(r) = q_i \sum_{0 \le L \le L_{max}} f_\eta\left(r - \left(\frac{1}{R_i} + \frac{L}{R}\right)^{-1}\right) p_L(L). \tag{2}$$

The function $p_L(L)$ needs to be obtained from the interference characteristics. In the next section we show how it can be derived analytically for resistive memory arrays. Given that, our proposed detector is

$$\hat{i}_{MAP} = \arg\max_{i \in \{1,\ldots,N\}} \Lambda_i(r), \tag{3}$$

which is the optimal MAP detector up to the assumption that the distribution of L is cut off at L_{max}.

C. Optimal Threshold Detection

The MAP detector of the previous sub-section may be optimal, but also very complex to implement. It essentially needs to consider every possible (H_i, L) pair in an exhaustive manner. Alternatively, a simpler detector may divide the r axis to fixed regions, and map each region to a hypothesis. In this formulation, we have to determine a set of $N-1$ thresholds $\{\tau_i\}_{i=1}^{N-1}$ that separate between the decision regions of different hypotheses. Formally, given a resistance measurement of r, the output of this detector is

$$\hat{i}_{TH}(r) = \{i : \tau_{i-1} \le r \le \tau_i\}.$$

Since the measured resistance is non-negative by definition, we can immediately define $\tau_0 = 0$ and $\tau_N = \infty$. The other τ_is need to be set such that the probability that $\hat{i}_{TH}(r) \neq i$, where i is the correct hypothesis, is minimized. Since r is monotone in the resistance R_i, the hypotheses H_i need to be ordered in ascending R_i, and the decision region for hypothesis i is continuous. Minimizing the error probability of the threshold detector $\hat{i}_{TH}(r)$ by optimization calculus gives the conditions

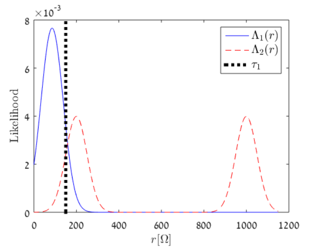
$$\Lambda_i(\tau_i) = \Lambda_{i+1}(\tau_i). \tag{4}$$

However, since a closed-form solution of (4) is difficult, we make an assumption that simplifies the derivation of the τ_is. We replace the sums of Gaussians in Λ_i and Λ_{i+1} (see (2)) by the single Gaussians that make the two hypotheses closest, that is, those that correspond to $L_i = 0$ and $L_{i+1} = L_{max}$. Hence, the threshold values with this approximation are

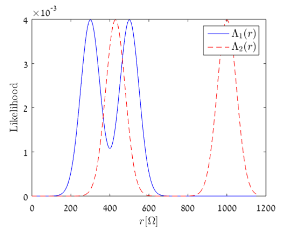
$$\tau_i \approx \frac{1}{2} \frac{\left[\frac{1}{R_{i+1}} + \frac{L_{max}}{R}\right]^{-2} - R_i^2 + 2\sigma^2 \log\left[\frac{q_i}{q_{i+1}} \frac{p_L(0)}{p_L(L_{max})}\right]}{\left[\frac{1}{R_{i+1}} + \frac{L_{max}}{R}\right]^{-1} - R_i}. \tag{5}$$

Note that the τ_i values depend on the assumed L_{max}. For good performance this L_{max} should be the largest value of L that still has significant probability to occur according to $p_L(\cdot)$. When we set $L_{max} = 0$, (5) degenerates to the threshold estimator for pulse-amplitude modulation (PAM) with non-uniform levels [25].

In Fig. 2 we show examples for detecting different distributions of r. In (a) a threshold detector works well, but in (b) threshold detection is sub-optimal because the L distribution interleaves the decision regions of H_1 and H_2.

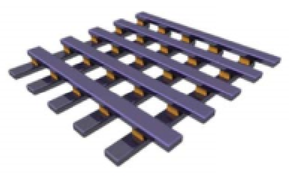


(a) $R_1 = 100\Omega$,$R_2 = 1000\Omega$,$R = 250\Omega$. Threshold detector, τ_1, is marked by a dashed line.

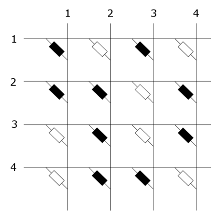


(b) $R_1 = 500\Omega$,$R_2 = 1000\Omega$,$R = 750\Omega$. Optimal regions are not continuous even if $L_{max} = 1$.

Fig. 2: Example of detectors for $N = 2$ hypotheses with different resistance values ($\sigma_\eta = 50\Omega$).



(a)



(b)

Fig. 3: (a): physical illustration of a crossbar array. Every row-column intersection is connected by a single resistive cell. (b): logical illustration of a crossbar array. White cells represent logical 0 bits, and black cells represent logical 1 bits.

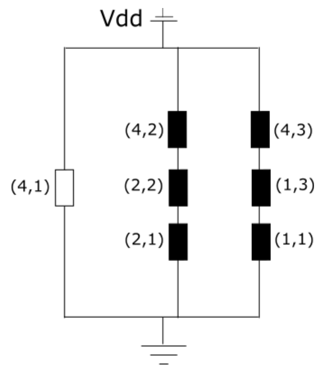
III. Application to Resistive Memory

Equipped with Section II's framework and tools for parallel-resistance detection, we now turn to refine them toward their application in resistive memories. A resistive crossbar array has m rows and n columns, and at the intersection of row i and column j lies the resistive cell (i, j). Parallel-resistance interference occurs in resistive crossbar arrays by an effect called *sneak paths*. When reading the cell (i, j), the resistance measurement of cell (i, j) is influenced by parallel (sneak) paths consisting of resistances of other array cells. Fig. 3 depicts a resistive crossbar in physical and logical illustrations. A sneak path is defined logically as a closed path originating from and returning to location (i, j), and traversing logical-1 cells through alternating vertical and horizontal steps. For example, cell $(4, 1)$ in Fig. 3b has a sneak path composed of cells $(4, 2)$, $(2, 2)$, and $(2, 1)$.

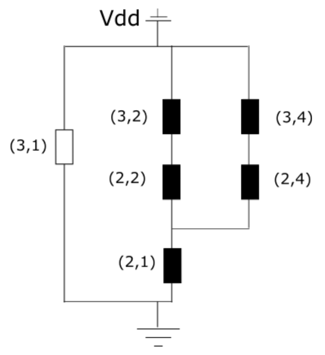
A. Heterogeneous Path Types and Cell Non-Linearity

In resistive memory a logical 1 bit is represented by low resistance, and a logical 0 bit by high resistance. To meet this convention, we adapt our notation and define $R(b)$ as the resistance value used to represent the logical bit $b \in \{0, 1\}$. With the notation of Section II we have $R(1) = R_1$ and $R(0) = R_2$.

To capture the interference induced by sneak paths, we need to find the resistance values that occur in parallel to the



(a) The cell (4,1) in Fig. 3b has $L = 2$ sneak paths involving $k_r = 2$ rows and $k_c = 2$ columns.



(b) The cell (3,1) has $L = 2$ sneak paths involving $k_r = 2$ rows and $k_c = 1$ column. Cell (2,1) is shared between paths.

Fig. 4: $L = 2$ sneak paths with different types.

resistance $R(b_{i,j})$ of the read cell at location (i, j). Recall that in Section II we assumed the simplistic model that the parallel resistance has a fixed value R, with a random multiplicity L. To extend the model to the more realistic scenario of sneak paths, we focus on the most dominant interference, which is caused by sneak paths of three cells all with the low resistance level $R(1)$. Sneak paths with more than three cells also exist, but their effect on data reliability is much less significant due to the much higher parallel resistances they induce. Because of the crossbar structure, the parallel resistance does not only depend on the *number* L of sneak paths affecting cell (i, j), but also on the *type* of the sneak-path combination. For L sneak paths affecting cell (i, j) we define the type as the number of rows k_r and the number of columns k_c participating in the L sneak paths (not counting row i and column j). In Fig. 4 we show two examples of $L = 2$ sneak paths having different types. For L up to 3, we characterize in Table I all types and their corresponding parallel resistance α_{L,k_r,k_c} in multiples of $R(1)$. Combinations of k_r, k_c not in the table can be obtained from included columns by row-column symmetry. For each L, the types are ordered by increasing interference severity (decreasing parallel resistance). The reason we stop at $L = 3$ is that we reached a parallel interference of $1 \cdot R(1)$ (right most column), which is as low as a legal hypothesis for the measured cell.

L	0	1	2		3			
# rows (k_r)	0	1	1	2	2	1	2	3
# columns (k_c)	0	1	2	2	2	3	3	3
α_{L,k_r,k_c}	∞	3	2	$3/2$	2	$5/3$	$6/5$	1

TABLE I: Types of $L = 0, 1, 2$ and 3 sneak-paths.

Another adaptation required to the model of Section II is considering the *non-linearity* of the memory cells. So far we assumed that the cells in the sneak path each contributes resistance $R(1)$ to its branch in the circuit. But real resistive cells are commonly manufactured with non-linearity, such that $R(1)$ is their resistance only under the full V_{dd} voltage. When participating in sneak paths, cells are under voltages that are fractions of V_{dd} (for example $V_{dd}/3$ in Fig. 4a), in which case their resistance is higher. In the refined model we define the cell non-linarity by the constant κ, setting the ratio between the cell resistance in a sneak path to its resistance when read with the full V_{dd}.

Combining both model refinements, we can re-write the expression for the read resistance value for cell (i,j) as

$$r_{b_{i,j}}(L,k_r,k_c) = \left(\frac{1}{R(b_{i,j})} + \frac{1}{\alpha_{L,k_r,k_c}\kappa R(1)}\right)^{-1} + \eta. \quad (6)$$

B. MAP and Threshold Detection

In this sub-section we provide the expressions for the MAP and threshold detectors with the refinements of the parallel-resistance model to resistive arrays with sneak paths. There are two hypotheses in this case: $b_{i,j}=1$ and $b_{i,j}=0$. The prior on the bit values is defined by $\Pr(b_{i,j}=1) = 1-\Pr(b_{i,j}=0) = q$, where q is known. We obtain the optimal MAP decision on $b_{i,j}$ by marginalization over all values of L, k_r, k_c, as follows

$$\frac{\sum_{L,k_r,k_c} f_\eta\left(r - r_1(L,k_r,k_c)\right)\Pr(L,k_r,k_c)}{\sum_{L,k_r,k_c} f_\eta\left(r - r_0(L,k_r,k_c)\right)\Pr(L,k_r,k_c)} \underset{\hat{b}=0}{\overset{\hat{b}=1}{\gtrless}} \frac{1-q}{q}.$$

The distribution $\Pr(L,k_r,k_c)$ can be calculated combinatorially in closed form, but we omit this for lack of space. Here too we can simplify the detector by truncating the sum at L_{max}.

For threshold detection, the expression for the threshold resistance between the two hypotheses is given by

$$\tau = \frac{\left[\frac{1}{R(0)} + \frac{1}{\alpha_{min}\kappa R(1)}\right]^{-2} - R(1)^2 + 2\sigma^2 \log\left[\frac{q}{1-q}\frac{p_L(0)}{p_L(L_{max})}\right]}{2\left(\left[\frac{1}{R(0)} + \frac{1}{\alpha_{min}\kappa R(1)}\right]^{-1} - R(1)\right)} \quad (7)$$

where α_{min} is chosen among the values of α_{L,k_r,k_c} with $0 \le L \le L_{max}$. We show bit-error rate (BER) results for the refined threshold and MAP detectors in Section V.

IV. Sneak-Path Reduction Methods

The principal scaling challenge of resistive memories is the fact that the occurrence of sneak paths significantly increases as the array dimensions grow. From our detection results in previous sections it is clear that reliable readout is not possible when the number of sneak paths affecting the read cell is above some threshold of L_{max} sneak paths. Thus, it is highly motivated to devise schemes that for given array dimensions will reduce the occurrence of multiple sneak paths, in particular more than L_{max} of them affecting an array cell. In this section we propose a method to reduce sneak-path occurance using a *coding scheme* specially designed for that purpose. Afterwards, we discuss a common hardware method, called *cell selectors*, to reduce sneak-path occurance.

A. Sneak-path Reducing Code

Since sneak paths is a data-dependent effect, it can also be mitigated by cleverly adapting the stored physical bits. The simplest and most immediate way to reduce sneak paths is by *distribution shaping* (also called *q shaping*), that is, changing the fraction of array bits that store the value 1 (low resistance). In Section III this fraction was denoted by q. It is clear that fewer 1s in the array mean fewer sneak paths affecting a read cell. But constraining the array bits to have low q also has an adverse effect on the storage rate of the array. Thanks to the simplicity of q shaping, we have a full characterization of the sneak-path distribution in the array as a function of q and the array dimensions m, n [22]. The following is a simple adaptation of a theorem from [22].

Proposition 1. *For a cell in a crossbar array whose bits are chosen i.i.d. Bernoulli with parameter q, the probability that there are exactly l sneak paths affecting the cell equals*

$$p_L(l) = \sum_{u=0}^{m-1}\sum_{v=0}^{n-1} p_{u,v}\binom{uv}{l} q^l (1-q)^{uv-l}, \quad (8)$$

where $p_{u,v} = \binom{m-1}{u}\binom{n-1}{v} q^{u+v}(1-q)^{m-1-u+n-1-v}$.

The q that gives the sneak-path distribution in (8) also induces a storage rate $\mathcal{R} = \mathcal{R}(q) \le 1$. To get a better sneak-path distribution than (8) for the same rate $\mathcal{R}$, we next propose a coding scheme based on encoding the array with 2×2 blocks.

Setting the parameter q in the distribution-shaping method "shapes" the array bits to have fewer sneak paths. Our main idea in the proposed coding scheme is that better sneak-path shaping can be obtained when imposing a richer structure on the array bits. The structure that we choose for the coding scheme is 2×2 blocks whose properties induce low incidence of sneak paths. 2×2 blocks give advantage in sneak-path mitigation, while their small size still allows simple encoding and decoding at a small access granularity of a pair of array rows, using known methods of distribution shaping.

The 2×2 shaping code. Consider the 16 possible assignments to a 2×2 bit word. To reduce sneak-path incidence, we first forbid using the 5 assignments that have 1s in more than half of the word (4 assignments with three 1s and the all-1 assignment). Now comes our key observation that among the assignments with two 1s, those with the two 1s in the same row or column are more prone to having multiple sneak paths affecting the same cell location. Because of that we forbid these 4 assignments as well. This leaves our code with 7 words out of the 16 possibilities. The 7 words divide into three symmetry classes according to their weight (see Table II), and the code is decided by determining the probabilities p_0, p_1, p_2 of words in these three respective classes. Before discussing

Word Probability	Words
p_0	$\begin{bmatrix}0&0\\0&0\end{bmatrix}$
p_1	$\begin{bmatrix}1&0\\0&0\end{bmatrix}, \begin{bmatrix}0&1\\0&0\end{bmatrix}, \begin{bmatrix}0&0\\1&0\end{bmatrix}, \begin{bmatrix}0&0\\0&1\end{bmatrix}$
p_2	$\begin{bmatrix}1&0\\0&1\end{bmatrix}, \begin{bmatrix}0&1\\1&0\end{bmatrix}$

TABLE II: 2×2 code-words and probabilities

the important problem of choosing the probabilities p_0, p_1, p_2

to optimize sneak-path reduction, we derive a closed-form expression for the sneak-path distribution given the chosen probabilities.

Theorem 2. *Let the crossbar array store bits whose 2×2 blocks are chosen i.i.d. from the 7 legal assignments with probability p_0, p_1, p_2 for each word with weight $0, 1$ and 2, respectively. The probability that an array cell is affected by exactly l sneak paths equals*

$$p'_L(l) = \sum_{u=0}^{\frac{n}{2}-1} \sum_{v=0}^{\frac{m}{2}-1} p'_{u,v} \cdot p'_{l|u,v}, \tag{9}$$

where

$$p'_{u,v} = \binom{\frac{n}{2}-1}{u}\binom{\frac{m}{2}-1}{v} \cdot (2p_1 + 2p_2)^{u+v}(1-2p_1-2p_2)^{n/2-1-u+m/2-1-v} \tag{10}$$

and

$$p'_{l|u,v} = \binom{uv}{l}(p_1+p_2)^l (1-p_1-p_2)^{uv-l}. \tag{11}$$

Proof. We derive the probability $p'_L(l)$ by conditioning on the number of 1s in the read cell's row, which we denote by u, and the number of 1s in the cell's column, which we denote by v. Observe the 2×2 words in Table II. Each 1 in the cell's row or column results from one of two weight-1 words (two other weight-1 words have no 1s in that row or column), or from one of the two weight-2 words. Also, we ignore the 1s from the word of the read cell itself, because they cannot cause sneak paths. This explains the expression for $p'_{u,v}$ in (10). Now given u, v, we have uv pairs of a 1 in the cell's row and a 1 in the cell's column. We examine the word intersecting the row and column of words corresponding to the pair of 1s. It can be seen that in all cases there are exactly two assignments to this word that cause a sneak path: one weight-1 word and one weight-2 word (the identities of these two words depend on the location of the pair 1s within their words). This explains the expression for $p'_{l|u,v}$ in (11). ☐

The proof of Theorem 2 reveals an important advantage of the 2×2 coding scheme: even though each row, column and their intersection, that potentially cause a sneak-path, can have one of $7^3 = 343$ word combinations, the symmetries in the problem simplify the analysis and yield compact and wieldy expressions. The simplest and most effective method to set the probabilities p_0, p_1, p_2 is *weight minimization*. Lower weights necessarily imply lower average number of ones within the array, and therefore less sneak-path in general. In that method, we are given a prescribed storage rate $\mathcal{R}$, and look for the combination of p_0, p_1, p_2 that satisfy the constraints: 1) $p_0 + 4p_1 + 2p_2 = 1$, 2) $\mathcal{R}(p_0, p_1, p_2) = \mathcal{R}$ (the rate associated with such a distribution is its entropy), and minimizing the expected codeword weight $\mathcal{W}(p_0, p_1, p_2) = p_1 + 2p_2$. In Fig. 5 we plot for each rate $\mathcal{R}$ the possible tail probabilities ($\Pr\{L > 3\}$) obtained by 2×2 coding with different p_0, p_1, p_2 distributions.

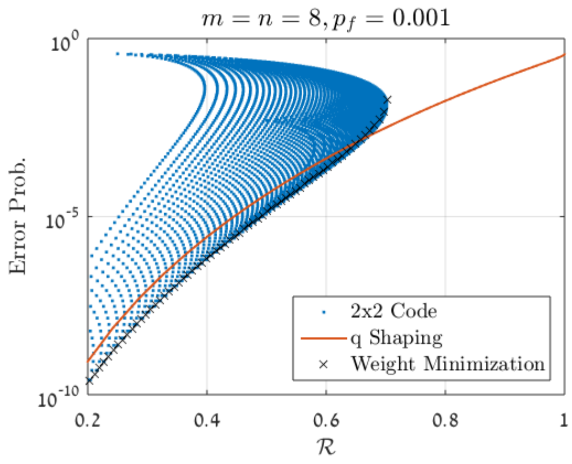


Fig. 5: Sneak-path tail probabilities as a function of storage rate. Showing 2×2 coding with weight minimization and other distributions, and compared to q shaping (with $0 \le q \le 0.5$).

For comparison, we show the same tail probability with simple q shaping. Weight minimization is shown to give the lowest tail probability among all p_0, p_1, p_2 distributions, and significantly lower than q shaping. In the next section we also show the favorable effect of 2×2 coding on the BER results.

B. Cell Selectors

A popular hardware method to mitigate the problem of resistive-array scaling due to sneak paths is by introducing *cell selectors*. To each memory cell a selector device is added in series, and blocks the reverse current flowing in sneak paths. Our model for cell selectors assumes they fail i.i.d. with probability p_f. When reading cell (i, j), if there is a 3-cell sneak path affecting cell (i, j) such that a selector failure occurs in the path cell not in row i and not in column j, then the sneak path will be active despite the selectors. With this behavior we now modify the sneak-path distribution to accommodate cell selectors. Given u 1s in row i and v 1s in column j, the probability to have l active sneak paths with selectors and q shaping is

$$\tilde{p}_{l|u,v} = \binom{uv}{l}(p_f q)^l (1-p_f q)^{uv-l}, \tag{12}$$

and the probability to have l active sneak paths with selectors and 2×2 coding is

$$\tilde{p}'_{l|u,v} = \binom{uv}{l}\left[(p_1+p_2)\,p_f\right]^l \left[1-(p_1+p_2)p_f\right]^{uv-l}. \tag{13}$$

The expressions for $p_{u,v}$ and $p'_{u,v}$ are unchanged with selectors compared to (8) and (9), respectively.

V. Simulation Results

In this section, the performance of the detection and coding schemes presented throughout this paper are evaluated via simulations. We consider the bit-error rate (BER) as a primary performance criterion for the various schemes. The simulations

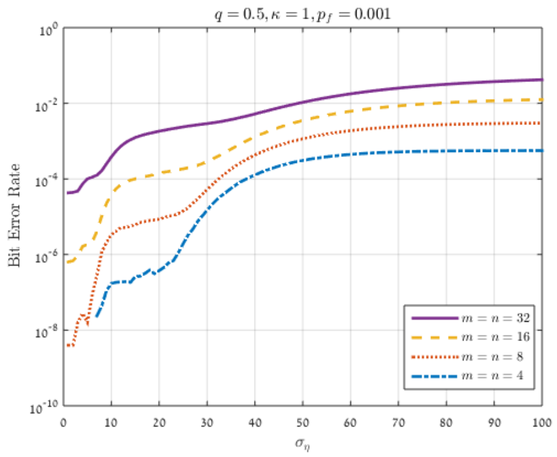


Fig. 6: BER of the MAP detector for several array sizes, with $q = 0.5$, $p_f = 10^{-3}$ and $\kappa = 1$.

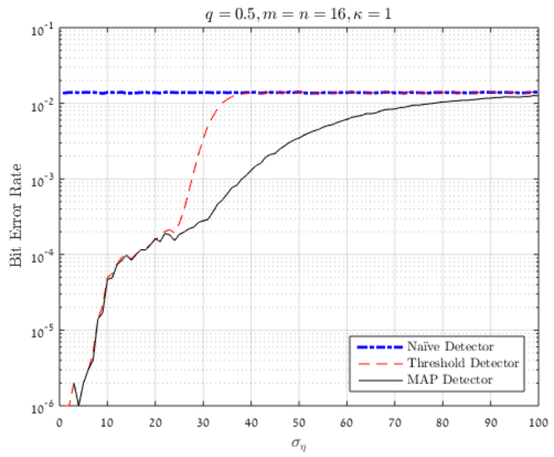


Fig. 7: BER comparison between different detectors for a 16×16 array with $p_f = 10^{-3}$ and $\kappa = 1$.

were conducted using resistance values of $R(1) = 10^2\,[\Omega]$ and $R(0) = 10^4\,[\Omega]$ with varying values of the noise standard deviation σ_η. m and n denote the crossbar array dimensions, and κ is the non-linearity coefficient. q is the prior input probability of bit assignments of 1, and $\mathcal{R}$ denotes the storage rate. All BER simulation of uncoded arrays ($q = 0.5$) were conducted with a selector fault probability of $p_f = 10^{-3}$, and the coded arrays were simulated without selectors ($p_f = 1$).

First, we conduct a comparison between the error rates of the MAP detector for several sizes of crossbars. Fig. 6 shows that as the array dimensions grow, so does the error rate. Specifically, the graph shows that error rates of around 10^{-6} can be achieved even in a noise regime of $20\%-30\%$ of $R(1)$. In addition, the step-like behavior of the error-rate plot can be observed: it abruptly changes its growth rate from steep to moderate and vice-versa. This phenomenon stems from the discrete nature of the interference itself. The Gaussians that assemble the likelihood functions expand their overlap as the noise magnitude increases. When a Gaussian related to some L overlaps with another Gaussian, the error rate grows steeply. But when they are already blended, the error rate reaches a certain saturation, until the overlap of the following Gaussian becomes dominant. These conclusions are supported also by Fig. 7, which shows a comparison of a naïve detector (threshold in $\frac{R(1)+R(0)}{2}$), the threshold detector and the MAP detector. It is seen that for low noise levels the threshold detector gives comparable performance to MAP, but at some point they diverge to the favor of the MAP.

In the context of 2×2 coding, we show in Fig. 8 a comparison of error rates with 2×2 coding and without (using q shaping instead), for several storage rates. The results show a clear advantage for the coded scheme, which improves the error rate in almost an order of magnitude for all of the examined storage rates. Using this code, or alternatively using more reliable selectors (i.e. smaller p_f), enables to enlarge the overall crossbar size while maintaining the error rate.

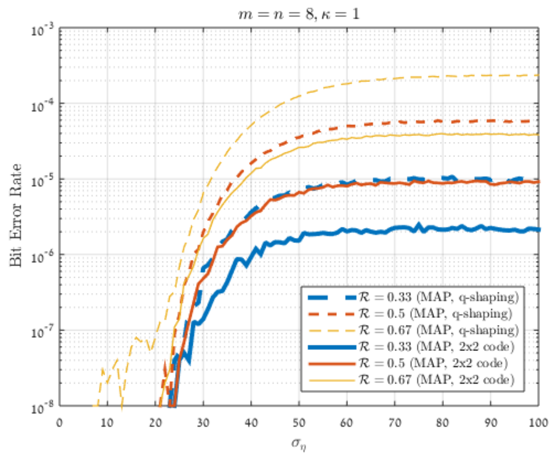


Fig. 8: BER of 2×2 coding (solid curves) vs. q shaping (dashed curves) for different storage rates. Array dimensions are $m = n = 8$, $p_f = 1$ and $\kappa = 1$.

VI. Conclusion

In this paper we provided a formal treatment for the problem of resistive-array readout. This treatment yielded constructive tools to combat sneak paths in both the detection and coding layers. Clearly the framework and tools developed here can be extended and improved in many directions. In the theoretical direction it is important to derive error analysis for different detectors. In the coding direction, the most interesting is to construct codes that optimize sneak-path reduction in the presence of cell selectors.

VII. Acknowledgement

The authors wish to thank Shahar Kvatinsky for valuable discussions. This work was supported in part by the Intel Center for Computing Intelligence and by the Israel Science Foundation.

REFERENCES

[1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[2] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proceedings of the 2009 IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33–36, IEEE Computer Society, 2009.

[3] T. Raja and S. Mourad, "Digital logic implementation in memristor-based crossbars," in *International Conference on Communications, Circuits and Systems (ICCCAS)*, pp. 939–943, IEEE, 2009.

[4] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (imply) logic: design principles and methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2014.

[5] R. Ben-Hur and S. Kvatinsky, *Processing within a Memristive Memory*. 2016.

[6] M. Di Ventra, Y. V. Pershin, and L. O. Chua, "Putting memory into circuit elements: memristors, memcapacitors, and meminductors [point of view]," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1371–1372, 2009.

[7] D. Chabi, W. Zhao, D. Querlioz, and J.-O. Klein, "Robust neural logic block (nlb) based on memristor crossbar array," in *IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 137–143, IEEE, 2011.

[8] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2408–2421, 2015.

[9] Y. Cassuto and K. Crammer, "In-memory hamming similarity computation in resistive arrays," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 819–823, IEEE, 2015.

[10] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Detection, diagnosis, and repair of faults in memristor-based memories," in *IEEE 32nd VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, April 2014.

[11] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Modeling, detection, and diagnosis of faults in multilevel memristor memories," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 822–834, 2015.

[12] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[13] M. A. Zidan, A. M. Eltawil, F. Kurdahi, H. A. Fahmy, and K. N. Salama, "Memristor multiport readout: A closed-form solution for sneak paths," *IEEE Transactions on Nanotechnology*, vol. 13, no. 2, pp. 274–282, 2014.

[14] X. Wang, M. Chen, Y. Shen, and X. Hu, "A new crossbar architecture based on two serial memristors with threshold," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, July 2015.

[15] 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, no. 42, p. 425204, 2009.

[16] S. Shin, K. Kim, and S.-M. Kang, "Analysis of passive memristive devices array: Data-dependent statistical model and self-adaptable sense resistance for rrams," *Proceedings of the IEEE*, vol. 100, pp. 2021–2032, June 2012.

[17] C. Liu and H. Li, "A weighted sensing scheme for reram-based cross-point memory array," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 65–70, IEEE, July 2014.

[18] M. Zidan, H. Omran, R. Naous, A. Sultan, H. Fahmy, W. Lu, and K. N. Salama, "Single-readout high-density memristor crossbar," *Scientific reports*, vol. 6, 2016.

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

[20] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 156–160, IEEE, 2013.

[21] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "On the channel induced by sneak-path errors in memristor arrays," in *International Conference on Signal Processing and Communications (SPCOM)*, pp. 1–6, IEEE, 2014.

[22] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Transactions on Information Theory*, vol. 62.

[23] R. Naous, M. A. Zidan, A. Sultan-Salem, and K. N. Salama, "Memristor based crossbar memory array sneak path estimation," in *14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, pp. 1–2, IEEE, 2014.

[24] T. Luo, O. Milenkovic, and B. Peleato, "Compensating for sneak currents in multi-level crosspoint resistive memories," in *49th Asilomar Conference on Signals, Systems and Computers*, pp. 839–843, IEEE, 2015.

[25] J. G. Proakis and M. Salehi, *Digital Communications*. McGraw-Hill, 2008.