Detection and Coding Schemes for Sneak-Path Interference in Resistive Memory Arrays

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

Resistive memory is a promising technology for achieving unprecedented storage densities and new in-memory computing features. However, to fulfill their promise, resistive memories require array architectures suffering from a severe interference effect called "sneak paths". In this paper we address the sneak-path problem through a communication-theory framework. Starting from the fundamental problem of readout with parallel-resistance interference, we develop several tools for detection and coding that significantly improve the memory reliability. For the detection problem we formulate and derive the optimal detector for a realistic array model, and then propose simplifications that enjoy similarly good performance and simpler implementation. Complementing detection for better error rates is done by a new coding scheme that shapes the stored bits to get lower sneak-path incidence. For the same storage rates, the new coding scheme exhibits error rates lower by an order of magnitude compared to known shaping techniques.

I. Introduction

The ever-increasing demand for data storage capacity drives a constant need to scale the storage density, while maintaining its power efficiency and reliability. As Flash technology for non-volatile memories seems to reach a scaling barrier, recent advancements in the fabrication of resistive devices suggest promising alternative technologies. The principal example of resistive memory is the *memristor* technology [2], but in addition other technologies, such as phase change memory (PCM) and spin torque transfer (STT), implement arrays with similar structure. The key in resistive technologies is that the memory cell is a passive two-terminal device that can be both read and written over a 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 [3]–[6], analog and neural computation [7]–[9], and vector similarity calculations [10]. 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 motivated extensive research efforts over the last few years, which contributed works toward modeling, detection and repair of faults in resistive arrays [11], [12]. By far 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 through other array cells distort the measurement, which may result in reading an erroneous state. 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 [13]–[15]. Other approaches concentrate on low-level electric analysis, which is meant to clean distorted measurements [16]–[19]. Additionally, models for the resistance distortion caused by sneak paths, with accompanying signal processing and data representation techniques, appear in [20], [21]. Finally, information-theoretic analysis and mitigation of sneak paths in crossbar memories was first studied in [22] and then extended specifically to resistive arrays in [23]–[25].

Despite all these impressive contributions, sneak paths remain a major problem for designers of next-generation memory architectures – part of the reason being that each scheme works with its own model and assumptions, with no clear way to accumulate their goodness in a bigger solution. Thus in this paper our approach at the sneak-path problem is to address it within a clear and general framework, which will allow the tools to extend to future technology and research advancements. The most suitable framework to deal with the problem is *communication theory*, because sneak paths can be modeled as a special kind of *interference* to the cell being read. The paper lays down a framework for resistive memory that encompasses the two central elements of communication theory: detection and coding/modulation. Our solutions to the *detection* problem start from the fundamental model of a parallel-resistance channel, and extend to practical settings with heterogeneous sneak-path configurations, cell-selectors implemented in hardware, and multiple reads. Our proposed *coding* scheme efficiently reduces the sneak-path incidence in the array, and its analysis enables to extend optimal detection to coded arrays as well. Our detection and coding schemes are devised with practical implementation in mind: the detectors are proposed with several degrees of simplification, and the code preserves our ability to access small data units within the array.

Part of the results in this paper was presented at the 2017 International Conference on Communications (ICC) held in Paris, France [1].

The results of the paper are presented as follows. In Section II the problem of detection with sneak paths is formalized as an estimation problem with interference and noise. Sneak-path interference comes from resistances *in parallel*, which introduces a unique and extremely challenging problem even when the margins between the low and high resistances are sizable. In Section III we formulate the optimal *maximum a posteriori (MAP)* detector for sneak paths, and derive an expression for it by calculating the incidence distribution of sneak paths of different types. Then we suggest an approximation of the MAP detector that is simpler to implement in precision-limited hardware, and show empirically that it offers essentially the same detection performance as the MAP detector. Toward real-memory implementation, in Section IV we study a simpler class of detectors: *threshold detectors*, whereby resistance measurements are compared against a single threshold value to decide between logical 1 (below threshold), or logical 0 (above threshold). In this section too we use the special structure of the problem to derive an approximately optimal threshold value that is much easier to compute, and is shown to give essentially the same performance as the optimal threshold. In Section V we introduce *coding* toward reduction of sneak-path incidence. We propose modulation (also called shaping) codes that at the same storage rates improve the error rates significantly compared to prior array-shaping techniques. The key idea of these codes is to shape the distribution of 2×2 blocks in the array, which is more efficient for sneak-path reduction than shaping individual-bit distributions. At the same time the 2×2 unit is small enough to allow read/write access to small parts of the array. We extend the calculation of the sneak-path distribution to 2×2-coded arrays, and hence can apply the detectors from the first part of the paper to coded arrays as well.

In summary, our work promotes the mitigation of sneak-path interference by a combination of rigorous formulations, analytical derivations, empirical insights, and practical simplifications. While the basic sneak-path problem is still inherently difficult due to the sheer number of potentially interfering paths, our proposed framework and tools mark clear routes toward the problem's containment in real memory arrays.

II. The Sneak-Path Channel

A. Sneak-Path Basics

Consider a resistive crossbar array, where at the intersection of row i and column j lies the resistive cell (i, j). The array is represented as a binary matrix A with m rows and n columns, where the bit in cell (i, j) is represented by $A_{i,j}$. Each cell can store a single bit, by modulating it to one of two predefined resistance levels. A logical 1 bit is represented by low resistance, and a logical 0 bit by high resistance. To meet this convention, we define $R(b)$ as the resistance value used to represent the logical bit $b \in \{0, 1\}$.

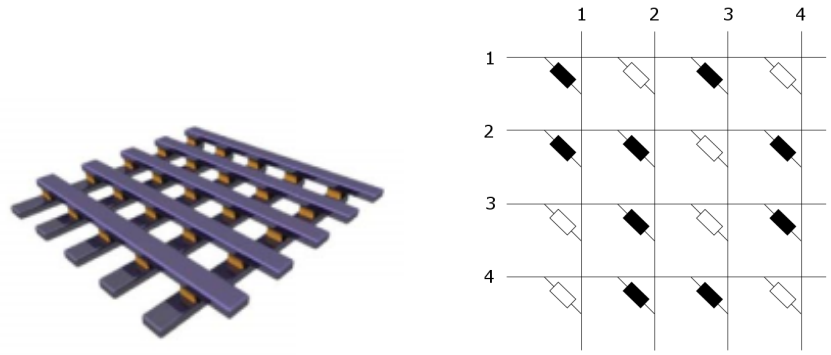
When reading the cell (i, j), the resistance measurement of the target cell is influenced by parallel (sneak) paths consisting of resistances of other array cells. In general, a sneak path is defined as a closed path originating from and returning to location (i, j), while traversing logical-1 cells through alternating vertical and horizontal steps. For example, cell $(4, 1)$ in Fig. 1b has a sneak path composed of cells $(4, 2), (2, 2)$ and $(2, 1)$. A longer sneak path, affecting cell $(1, 4)$, consists of 5 cells: $(3, 4), (3, 2), (2, 2), (2, 1)$ and $(1, 1)$. The interference caused by sneak paths is most dominant when the read cell has high resistance, while the parallel cells have low resistance values. This situation can cause the target cell to be erroneously read as low resistance, due to the equivalent value being sensed by the peripheral reading hardware. This issue has broad implications both on the storage reliability and on the scaling potential of currently developed resistive arrays.

In principle, sneak paths can have different (odd) lengths, where the length is the number of cells participating in the path in parallel to (i, j). However, in this work we consider only *simple* sneak paths, i.e., paths that contain exactly 3 cells in parallel to (i, j). As will be shown in the sequel, this length of sneak paths is by far the most dominant interference among all potential sneak-path lengths. In the rest of the paper we define a *sneak path* in parallel to cell (i, j) as a pair of row indices i, i' and a pair of column indices j, j', such that

$$A_{i,j'} = A_{i',j'} = A_{i',j} = 1 \tag{1}$$

and $i \neq i', j \neq j'$.

In actual resistive arrays, the most popular method to combat and mitigate the problem of sneak paths is by introducing *cell selectors*. A selector is an electrical device that allows current to flow only in one direction. Since sneak paths inherently produce reverse currents in at least one of the cells along the parallel path, placing a selector in series to each array cell completely eliminates sneak paths in the entire array. Our basic assumption is that imperfections in the production or maintenance of the memory cause cell selectors to fail. We model this behavior using a random fault model, in which for every read operation each selector within the array fails i.i.d. with probability p_f. For a sneak path to be active despite the selectors, it does not suffice that there will be a faulty selector along the path. Cells (i, j') and (i', j) still conduct current in the forward direction, and will therefore not be affected by selectors. Only the sneak current through cell (i', j') flows in the reverse direction compared to a regular-readout current flow. Therefore, a sneak path affecting cell (i, j) as in (1) will be active only if the selector in cell (i', j') fails. In other words, in a crossbar memory with selectors, sneak-path interference will occur for cell (i, j) if the condition in (1) holds *and* the selector in cell (i', j') fails. In this case, the sensed resistance value is the equivalent resistance of the read cell, $R(A_{i,j})$, in parallel to the sneak-path resistance.



(a) (b)

Fig. 1: (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.

B. Parallel-Resistance Channel

Given the mechanism explained in the previous sub-section, the nature of interferences caused by sneak paths can be modeled in the following manner: the desired resistor is measured in parallel to a path of other resistors created within the array during the measurement. Before addressing the complex realities of resistive-cell readout in arrays with sneak paths, we formulate in this sub-section a simpler model.

In the simpler model we have a target resistor to read that has in parallel to it L resistors, each with resistance R. Starting from this simplified setup is helpful for better seeing the underlying interference problem. Let H_0 and H_1 denote two hypotheses that represent the two possible resistance values $R(0)$ and $R(1)$ for the target cell. We denote the resistance measured in cell (i,j) by $r_{A_{i,j}}$ and the resistance of each parallel resistor by R. The number of interfering parallel resistors is denoted by L. A reasonable assumption is that the interfering resistance R is known, but L is a data-dependent random variable, which is typically unknown during the measurement. In addition, we consider Gaussian measurement noise η, which is added to the measured resistance and is independent of its value. Hence, given the hypothesis H_b, $b \in \{0,1\}$, from elementary electric theory (resistors in parallel), the measured resistance can be written as the inverse sum of reciprocals of the target resistance and L branches with resistance R:

$$r_{A_{i,j}} = \left(\frac{1}{R(A_{i,j})} + \frac{L}{R} \right)^{-1} + \eta. \tag{2}$$

η is a measurement noise that is added to the measured resistance and is independent of its value. It can be used to model thermal noise in the peripheral read circuits and/or variability in the fabrication of the memory. We call this equation the *parallel-resistance channel*. 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 small values of L result in severe drops in the measured resistance. For example, Fig. 2 depicts a scenario where even $L = 1$ significantly shrinks the margin between the two hypotheses. 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. For any practical scenario, L is assumed to be bounded by some constant upper bound L_{max}, beyond which reliable detection is simply impossible. The value of L_{max} depends on the problem parameters and on the specific detector being used.

C. Sneak Paths as Parallel-Resistance Channel

To capture the interference induced by sneak paths, we first need to find the resistance values that are measured in parallel to the resistance of the read cell. Because of the crossbar structure, the parallel resistance does not only depend on the *number* of active sneak paths affecting cell (i,j), denoted L in Section II-B, but also on the *type* of the sneak-path combination. We define the sneak-path type using the number of active sneak paths as well as number of rows and columns participating in those sneak paths. For ease of notation we use vector notation for the parameters of the sneak-path type.

Definition 1. *The type of a sneak path is a vector $\underline{\lambda} = (L; k_r, k_c)$, where L denotes the number of active sneak paths, k_r denotes the number of rows and k_c denotes the number of columns that participate in these L sneak paths.*

Note that k_r is the number of different i' indices (from (1)) that appear in the L sneak paths, and similarly k_c is the number of different j' indices. Each array cell can be associated with a corresponding sneak-path type. For example, if some array

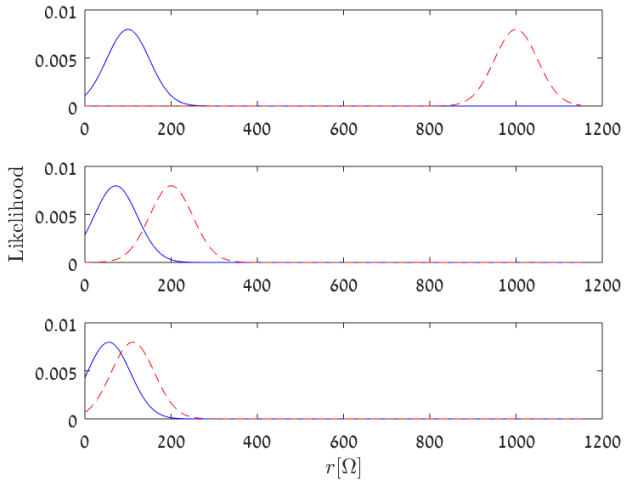
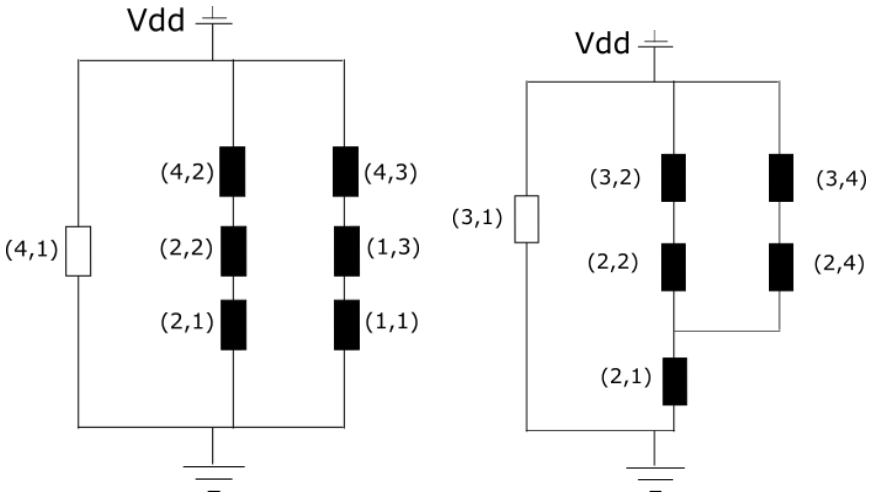


Fig. 2: Plots of the conditional distributions of r for the two hypotheses: $H_1 : A_{i,j} = 1$ and $H_0 : A_{i,j} = 0$. The different plots are for $L = 0, 1$ and 2 (upper, middle and lower plots, respectively). The parameters are $R(0) = 1000\Omega$, $R(1) = 100\Omega$, $R = 250\Omega$ and the noise standard deviation is $\sigma_\eta = 50\Omega$.



(a) The cell (4,1) in Fig. 1b 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. 3: $L = 2$ sneak-paths with different types.

cell has no sneak paths around it, its type is $(0; 0, 0)$. A less trivial example of sneak-path types is now given for two specific array cells from Fig. 1.

Example 1. *The importance of differentiating sneak path combinations by their type can be demonstrated using the array in Fig. 1b. The logical presentation of the array in this case is*

1	0	1	0
1	1	0	1
0	1	0	1
0	1	1	0

.

There are several cells within the array that have sneak paths, but let us focus on cells $(3, 1)$ *and* $(4, 1)$*. Both of these cells have* $L = 2$ *sneak paths, but each has a different type. For cell* $(3, 1)$ *the type is* $\underline{\lambda} = (2; 2, 1)$ *while cell* $(4, 1)$ *has type of* $\underline{\lambda} = (2; 2, 2)$*. This causes each cell to have a different interfering circuit, as depicted in Fig. 3. Therefore, although the number of sneak paths is equal in both cases, the magnitude of interference is not the same, since their equivalent resistance differs.*

Given a specific sneak-path type, we can derive an analytical expression for the sensed resistance value for cell (i, j). We

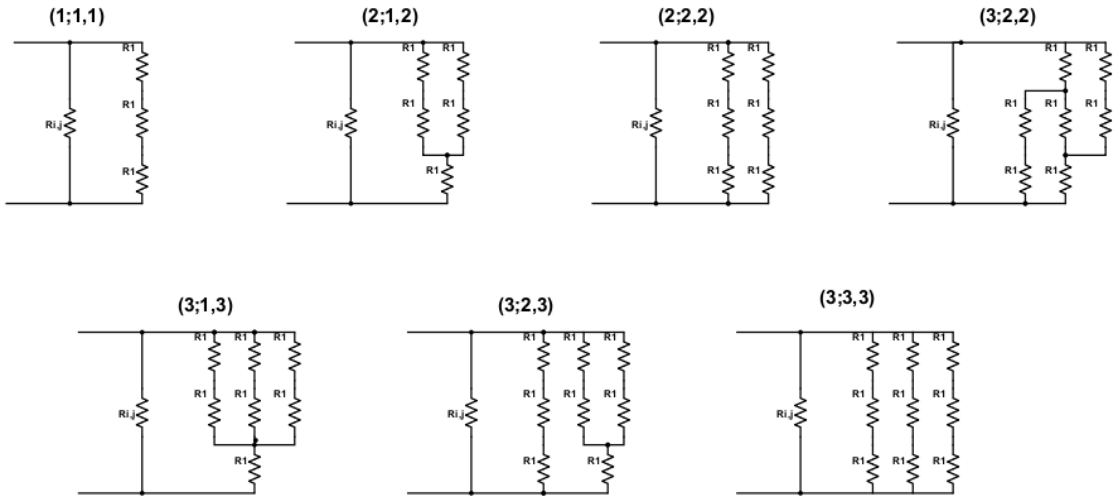


Fig. 4: Electrical circuits that describe the sensed resistance for values of $\underline{\lambda}$ in Table I.

model the measured resistance as a combination of cell's resistance value, the sneak-path interference and additive noise. Denoting the sneak-path equivalent parallel resistance as $\alpha(\underline{\lambda})R(1)$, we get

$$r_{A_{i,j}}(\underline{\lambda}) = \rho(A_{i,j}, \underline{\lambda}) + \eta, \tag{3}$$

where

$$\rho(A_{i,j}, \underline{\lambda}) \triangleq \left(\frac{1}{R(A_{i,j})} + \frac{1}{\alpha(\underline{\lambda})R(1)} \right)^{-1}.$$

The performance of the detection framework presented in the following sections depends on precise characterization of the sneak-path types $\underline{\lambda}$ and their incidence in the array. In Table I, the first four rows list all sneak-path types for L up to 3 and their corresponding values of $\alpha(\underline{\lambda})$. The last row adds important information about the type's incidence, which we discuss after presenting the detector in the next section. The values of $\alpha(\underline{\lambda})$ are obtained by simple electric-theory equivalent-resistance calculations. The circuits that correspond to these equivalent resistances appear in Fig. 4. 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). In the table we stop at $L = 3$ because $\alpha = 1$ in the last column means sneak-path resistance equaling $R(1)$, which is a "cutoff" value for being able to distinguish between $R(0)$ and $R(1)$ in the measured cell. We later designate L_{max} as the maximal number of sneak paths handled by the detection scheme, and for pure resistive cells we have $L_{max} = 3$ (when cells have non-linearity, we can get $\alpha > 1$ for more than 3 sneak paths, in which case we may have $L_{max} > 3$).

III. Optimal Detection Schemes

The model described in the previous section induces a detection problem, commonly referred to as *composite hypothesis testing* [26]. Unlike simple hypothesis testing, composite hypotheses involve hidden parameters that govern the posterior probabilities of the hypotheses. In our case, for each array cell (i, j) we have two composite hypotheses, H_0 and H_1, for the two different possible resistances, where $\underline{\lambda}$ is a hidden parameter. Although $R(0)$ and $R(1)$ are known, the interfering parallel resistance $\alpha(\underline{\lambda})R(1)$ depends on $\underline{\lambda}$ and is therefore random (since $\underline{\lambda} = (L; k_r, k_c)$ depends on specific assignment of bits in the array and is typically unknown during the measurement).

A. MAP Detector

An optimal detector for the model presented in (3) can be formulated using the Maximum A-Posteriori (MAP) estimator for the value of $A_{i,j}$, given the measurement $r_{A_{i,j}}$. Assume the binary data stored in the memory cells are distributed Bernoulli i.i.d., i.e.,

$$\forall i, j, \;\; \Pr(A_{i,j} = 1) \triangleq q_1 = 1 - q_0 \triangleq 1 - \Pr(A_{i,j} = 0). \tag{4}$$

q_1 is the parameter of the Bernoulli distribution, also denoted q in the sequel. Having this prior, as well as the statistics of the other variables in (3), let us now define the likelihood function as a weighted sum of conditional likelihoods

$$\Lambda_b(r) = \sum_{\underline{\lambda}'} f_\eta\Big(r - \rho\left(b, \underline{\lambda}'\right)\Big) p_{\underline{\lambda}}(\underline{\lambda}'), \tag{5}$$

where $b \in \{0,1\}$, and $f_\eta(\cdot)$ is the noise probability density function. For practical reasons, we restrict the sum to values of $\underline{\lambda}$ that correspond to $0 \le L \le L_{max}$ ($L_{max} = 3$ in Table I). The realization of such detector requires to obtain the prior probabilities of the sneak-path types $\underline{\lambda}$. The distribution $p_{\underline{\lambda}}$ can be calculated combinatorially in closed form for values of L up to $L_{max} = 3$, as shown in the following theorem. Note that the probabilities depend on the array size, the prior of each bit and the selector fault probability.

Theorem 1. *The probability that a cell has sneak-path type $\underline{\lambda}' = (L; k_r, k_c)$ in an array whose bits are chosen i.i.d. Bernoulli with parameter q equals*

$$p_{\underline{\lambda}}(\underline{\lambda}') = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} \mathcal{A}_{u,v}(\underline{\lambda}') p_{u,v} p_{L|u,v}, \tag{6}$$

where $p_{u,v} = \binom{m-1}{u}\binom{n-1}{v} q^{u+v}(1-q)^{m-1-u+n-1-v}$, $p_{L|u,v} = (p_f q)^L (1-p_f q)^{uv-L}$ and the values of $\mathcal{A}_{u,v}(\underline{\lambda})$ are as listed in Table I.

Proof. The probability $p_{\underline{\lambda}}(L; k_r, k_c)$ is derived by marginalizing over the number of 1s in the read cell's row, which we denote u, and the number of 1s in its column, which we denote v. The corresponding probability, $p_{u,v}$, can be easily derived using the Bernoulli distribution. Now, given u and v 1s in the row and column, respectively, the probability of having a specific combination of L sneak-paths is $(p_f q)^L (1 - (p_f q))^{uv-L}$, because $p_f q$ is the probability that the cell at location i', j' in the intersection of a 1-row and a 1-column is both written to 1 and has a faulty selector. The number of combinations for each type, denoted $\mathcal{A}_{u,v}(L; k_r, k_c)$, is as follows:

- $L = 0$ requires to force all uv intersecting cells to zero, yielding a single combination.
- The case $L = 1$ sneak path has only one type: $(k_r, k_c) = (1,1)$. In this case, there are uv possible locations for the intersecting 1.
- The case of $L = 2$ has more possibilities. In the case of $(1,2)$, we choose a row and then two locations in the same row, such that the order of selection does not matter. This yields $\frac{uv(v-1)}{2}$. The type $(2,2)$ induces $\frac{uv(u-1)(v-1)}{2}$ combinations, since we select a pair of rows and a pair of columns, and can match the pairs in two ways (hence the denominator 2 and not 4).
- Finally, for $L = 3$ we have 6 types, divided into 4 categories. The type $(1,3)$ requires to choose a row and then three cells in this row, yielding $\frac{uv(v-1)(v-2)}{6}$ combinations. For type $(2,3)$, we take a configuration of type $(1,3)$ and choose which of the three cells to move, and to which other row. This yields $\frac{uv(u-1)(v-1)(v-2)}{2}$ combinations. The type $(2,2)$ is similar to $L = 2$, only factor 2 larger, since we have to assign the third sneak path to one of two row-column pairs. The type $(3,3)$ requires to choose three rows and three columns, and then one of six ways to choose the three intersection cells, yielding $\frac{uv(u-1)(v-1)(u-2)(v-2)}{6}$ combinations.

The remaining cases can be easily calculated by noting that due to symmetry $\mathcal{A}_{u,v}(L; k_r, k_c) = \mathcal{A}_{v,u}(L; k_c, k_r)$. □

L	0	1	2		3			
k_r	0	1	1	2	2	1	2	3
k_c	0	1	2	2	2	3	3	3
$\alpha(\underline{\lambda})$	∞	3	2	$3/2$	2	$5/3$	$6/5$	1
$\mathcal{A}_{u,v}(\underline{\lambda})$	1	uv	$\frac{uv(v-1)}{2}$	$\frac{uv(u-1)(v-1)}{2}$	$uv(u-1)(v-1)$	$\frac{uv(v-1)(v-2)}{6}$	$\frac{uv(u-1)(v-1)(v-2)}{2}$	$\frac{uv(u-1)(v-1)(u-2)(v-2)}{6}$

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

The values of $\alpha(\underline{\lambda})$ in Table I combined with the probabilities calculated in Theorem 1 provide all the information required for applying optimal MAP detection truncated at $L_{max} = 3$.

Detector 1. *We obtain the optimal decision on $A_{i,j}$ by applying the MAP decision rule*

$$\frac{\Lambda_1\left(r_{A_{i,j}}\right)}{\Lambda_0\left(r_{A_{i,j}}\right)} \underset{\hat{A}_{i,j}=0}{\overset{\hat{A}_{i,j}=1}{\gtrless}} \frac{q_0}{q_1}. \tag{7}$$

Detector 1 maximizes the a-posteriori probability of the hypothesis without knowledge on $\underline{\lambda}$. However, even when the exact sneak-path type $\underline{\lambda}$ is known, the detection may still suffer errors due to the additive noise. For example, the likelihoods of 0 and 1 in a 4×4 array are depicted in Fig. 5. In addition to the likelihoods $\Lambda_b(r)$, which do not assume any prior knowledge on $\underline{\lambda}$ (except its distribution), we also plotted the actual probabilities given the values of $\underline{\lambda}$ for specific cells in the array from Example 1. As can be seen, the severity of the interference depends on the specific sneak-path types. Also, even when the types are known there is still some overlap between the conditional likelihoods.

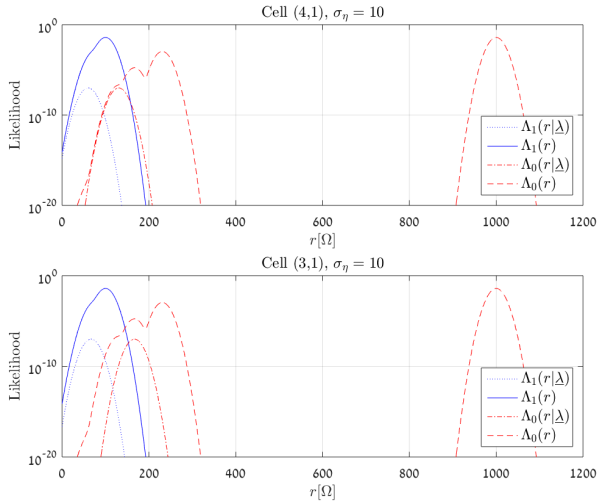


Fig. 5: Probability density functions of the resistance measured in cells $(4,1)$ ($\underline{\lambda} = (2;2,2)$) and $(3,1)$ ($\underline{\lambda} = (2;2,1)$) from the array in Example 1, with and without prior knowledge of $\underline{\lambda}$, and assuming additive Gaussian noise with $\sigma_\eta = 10\Omega$. The $\underline{\lambda}$-independent solid and dashed curves are identical between the two cells, but the $\underline{\lambda}$-dependent dotted and dash-dotted curves of cell $(4,1)$ are to the left of the ones in cell $(3,1)$, due to the stronger interference.

B. Error Probability

The performance of Detector 1 can be analyzed using numerical tools or by simulation. We start by presenting an expression for the error probability associated with Detector 1. Full analytic calculation of this expression is hard, due to the composite nature of the hypotheses, but evaluation via numerical calculations is indeed possible. In general, the error probability is described by the following expression

$$p_e^{MAP} = \sum_{b\in\{0,1\}} p(H_{\bar{b}}|H_b)q_b = p(H_1|H_0)q_0 + p(H_0|H_1)q_1.$$

We marginalize over the hidden-variable vector $\underline{\lambda}$ to get

$$p(H_{\bar{b}}|H_b) = \sum_{\underline{\lambda}} p(H_{\bar{b}}|H_b, \underline{\lambda})p(\underline{\lambda})$$

where

$$p(H_{\bar{b}}|H_b, \underline{\lambda}) = \int_{r:\frac{\Lambda_{\bar{b}}(r)}{\Lambda_b(r)} \geq \frac{q_b}{q_{\bar{b}}}} \frac{\exp\left(-\frac{1}{2\sigma_\eta^2}(r-\rho(b,\underline{\lambda}))^2\right)}{\sqrt{2\pi\sigma_\eta^2}} dr.$$

Now, since $p(\underline{\lambda})$ is known (see Theorem 1), all that is left for evaluating the error probability is to calculate the integral (numerically). The calculation of the error probability for a range of standard deviations of the Gaussian additive noise η appears in Fig. 6 (the error curves were also validated via bit-error rate (BER) simulations for the completeness of the analysis).

We conducted a comparison between the error rates of the MAP detector for several array sizes. It can be seen in Fig. 6 that the error rate rises as the array dimensions grow. Specifically, the figure shows that relatively low error rates (e.g. around 10^{-4} for 16×16 array) can be achieved even in a noise regime of $10\% - 20\%$ of $R(1)$, when cell selectors have $p_f = 10^{-3}$. 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. We conjecture that this phenomenon stems from the discrete nature of the interference itself. The Gaussians that constitute the likelihood functions expand their overlap as the noise magnitude increases. When a Gaussian related to some $\underline{\lambda}$ 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 next Gaussian becomes dominant.

C. Multiple-Reads Detector

A technique that can be used to enhance the detection performance, i.e., reduce the bit-error rate, utilizes multiple readouts of the same cell to improve the detection statistics. Let us assume we have a set of N measurements of the cell's resistance

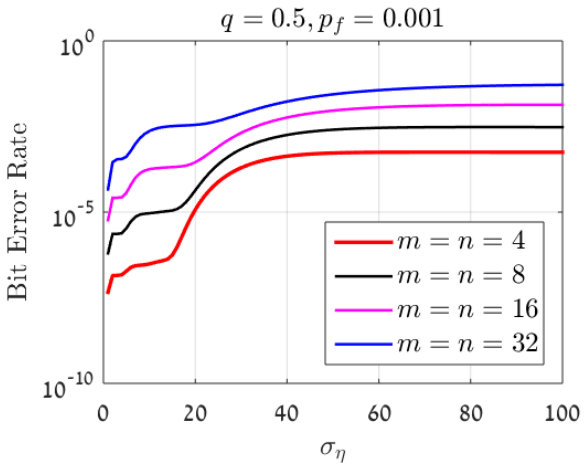


Fig. 6: Calculated error probability of the MAP detector for several array sizes, with $R(0) = 1000\Omega$, $R(1) = 100\Omega$, $q = 0.5$ and $p_f = 10^{-3}$. As the array size grows, the rising sneak-path probability adversely affects the error rate.

denoted $r_{A_{i,j}}(k)$ where $k = 1, \ldots, N$. Each measurement is as in (3). The measurements are represented as an N-dimensional vector

$$\underline{r}_{A_{i,j}} = \left(r_{A_{i,j}}(1), \ldots, r_{A_{i,j}}(N)\right)^T.$$

The MAP detector for multiple measurements is

$$\frac{\Lambda_1\left(\underline{r}_{A_{i,j}}\right)}{\Lambda_0\left(\underline{r}_{A_{i,j}}\right)} \underset{\hat{A}_{i,j}=0}{\overset{\hat{A}_{i,j}=1}{\gtrless}} \frac{q_0}{q_1}, \tag{8}$$

where $\Lambda_b(\underline{r}_{A_{i,j}}) = p(\underline{r}_{A_{i,j}}|b)$ is the likelihood of the measured vector given the hypothesis b. We assume the measurements are collected with i.i.d. additive noise but note that the parameter vector $\underline{\lambda}$ remains unchanged between measurements. In other words, the same information bits and sneak-path interference bits are measured N times with independent noise. Under this setup, the multi-dimensional likelihood function $p(\underline{r}_{A_{i,j}}|b)$ can be decomposed into a summation similar to the one-dimensional case

$$\Lambda_b(\underline{r}_{A_{i,j}}) = p(\underline{r}_{A_{i,j}}|b) = \sum_{\underline{\lambda}} p(\underline{r}_{A_{i,j}}|b, \underline{\lambda}) p_{\underline{\lambda}} = \sum_{\underline{\lambda}} \frac{p_{\underline{\lambda}}}{\left(2\pi\sigma_\eta^2\right)^{N/2}} \exp\Big(- \frac{1}{2\sigma_\eta^2} \sum_{k=1}^{N} (\underline{r}_{A_{i,j}}(k) - \rho(b, \underline{\lambda}))^2 \Big). \tag{9}$$

The resulting detector is a direct generalization of the single-measurement MAP detector from (7) (Detector 1). Since MAP detection only requires to find the b that maximizes (9) (weighted by its prior), a standard simplification is to only consider the terms that depend on the hypothesis, thus getting

$$\hat{A}_{i,j} = \arg\max_{b\in\{0,1\}} q_b \Lambda_b(\underline{r}_{A_{i,j}}) = \arg\max_{b\in\{0,1\}} q_b \sum_{\underline{\lambda}} p_{\underline{\lambda}} \exp\Big(- \frac{N\rho(b, \underline{\lambda})^2}{2\sigma_\eta^2} \Big) \exp\Big(\frac{\rho(b, \underline{\lambda})}{\sigma_\eta^2} \sum_{k=1}^{N} \underline{r}_{A_{i,j}}(k) \Big). \tag{10}$$

We notice that the evaluation of (10) for optimal decision entails the computational challenge of summing (over $\underline{\lambda}$) terms spanning a very large number range. Implementing this summation in a precision-limited hardware may result in detection errors due to rounding. To avoid this, we propose the following detector that does not suffer from these computational issues.

Detector 2. *First calculate*

$$\hat{\underline{\lambda}} \triangleq \arg\max_{\underline{\lambda}} p(\underline{\lambda}|\underline{r}_{A_{i,j}}) = \arg\max_{\underline{\lambda}} \sum_{b\in\{0,1\}} q_b p_{\underline{\lambda}} p(\underline{r}_{A_{i,j}}|\underline{\lambda}, b), \tag{11}$$

and then make the decision

$$\hat{A}_{i,j}|\hat{\underline{\lambda}} = \arg\max_{b\in\{0,1\}} \Big\{ \log q_b + \frac{\sum_{k=1}^{N} \underline{r}_{A_{i,j}}(k)}{\sigma_\eta^2} \rho(b, \hat{\underline{\lambda}}) - \frac{N}{2\sigma_\eta^2} \rho(b, \hat{\underline{\lambda}})^2 \Big\}. \tag{12}$$

Detector 2 works in two stages: first estimate the hidden parameter $\underline{\lambda}$, and then determine the most likely hypothesis given the estimated parameter value. As a result, when iterating over the different possible $\underline{\lambda}$s we only need the *maximization* operation, which is easier to implement with limited precision; summation is now done over just two b hypotheses instead of many $\underline{\lambda}$s

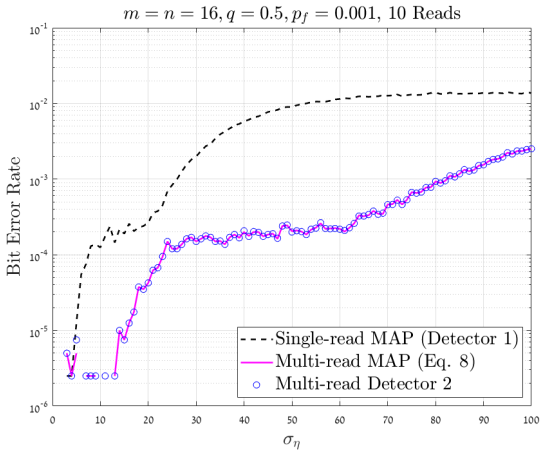


Fig. 7: Simulation of bit-error rates for multiple-read detectors (MAP and Detector 2) with 10 reads, compared to single-read MAP detector.

in (10). Once we have the most likely estimate $\hat{\underline{\lambda}}$, in the second stage (12) there is no summation, and we can thus work in the more stable log domain. Theoretically speaking, Detector 2 is not equivalent to MAP, because the separation to two stages may, in principle, lead to a decision that is not globally optimal. However, we find in extensive empirical evaluation that the performance of Detector 2 is in practice equivalent to MAP. In Fig. 7 we present the bit-error rates of single- and multiple-read detectors. As expected, the multiple-read detector shows an advantage over the single-read detector. It is also observed that Detector 2 reaches error rates almost identical to the multiple-read MAP detector. We conjecture that the property that allows this equivalent performance is that given any sneak-path type $\underline{\lambda}$, one bit hypothesis b has a negligible likelihood $p(\underline{r}_{A_{i,j}}|\underline{\lambda}, b)$ compared to the other hypothesis, and thus in practice the sum in (11) is dominated by the contribution of the more likely b.

IV. Low Complexity Detection Schemes

The full MAP detector may be optimal, but it is also complex to implement. Even with the simplification of Detector 2 it essentially requires to consider every possible assignment of the parameter vector $\underline{\lambda} = (L; k_r, k_c)$, and evaluate the likelihood function in an exhaustive manner. A much simpler alternative, inspired by methods used in Flash memories, divides the resistance axis to pre-defined *fixed and convex* regions, and maps each region to a different hypothesis. In this formulation, we have to determine a threshold resistance value, τ, that separates between the decision regions of the two possible hypotheses.

A. Single-Read Threshold Detection

For single-read detection, given a resistance measurement of $r_{A_{i,j}}$, the threshold detector is simply

$$r_{A_{i,j}} \underset{\hat{A}_{i,j}=1}{\overset{\hat{A}_{i,j}=0}{\gtrless}} \tau. \tag{13}$$

Optimally, the threshold value should be set such that the corresponding error probability is minimized, i.e.,

$$\tau = \arg\min_{0 \le t \le \infty} P_e^{TH}(t),$$

where $P_e^{TH}(t)$ is the error probability associated with a threshold t. This probability is given by

$$P_e^{TH}(t) = \Pr\{r_{A_{i,j}} > t | A_{i,j} = 1\} q_1 + \Pr\{r_{A_{i,j}} \le t | A_{i,j} = 0\} q_0. \tag{14}$$

However, direct derivation of the threshold in this way is hard since the expressions resulting from (14) are hard to manipulate. Therefore, we next try to apply reasonable assumptions to simplify the expressions. Suppose that the dominant errors occur when the noise causes the detector to mix between 0s with a particular type of sneak path $\underline{\lambda}^{(0)}$ and 1s with another particular type of sneak path $\underline{\lambda}^{(1)}$. The identities of $\underline{\lambda}^{(0)}$,$\underline{\lambda}^{(1)}$ will be discussed later. In that case, the error probability of the threshold detector simplifies to

$$\tilde{P}_e^{TH}(t, \underline{\lambda}^{(0)}, \underline{\lambda}^{(1)}) \triangleq q_1 p_{\underline{\lambda}^{(1)}} Q\Big(\frac{t - \rho(1, \underline{\lambda}^{(1)})}{\sigma_\eta}\Big) + q_0 p_{\underline{\lambda}^{(0)}} \Big(1 - Q\Big(\frac{t - \rho(0, \underline{\lambda}^{(0)})}{\sigma_\eta}\Big)\Big), \tag{15}$$

where $Q(\cdot)$ is the complementary cumulative density function of a Gaussian random variable. Now we define the optimal threshold detector for $\underline{\lambda}^{(0)}$,$\underline{\lambda}^{(1)}$ as

$$\tau(\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}) \triangleq \arg\min_{0\le t\le\infty} \tilde{P}_e^{TH}(t,\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}), \tag{16}$$

and can find it in closed form

$$\tau(\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}) = \frac{\rho(0,\underline{\lambda}^{(0)})^2-\rho(1,\underline{\lambda}^{(1)})^2-2\sigma_\eta^2\ln\left(\frac{q_0 p_{\underline{\lambda}^{(0)}}}{q_1 p_{\underline{\lambda}^{(1)}}}\right)}{2(\rho(0,\underline{\lambda}^{(0)})-\rho(1,\underline{\lambda}^{(1)}))} = \frac{\rho(1,\underline{\lambda}^{(1)})+\rho(0,\underline{\lambda}^{(0)})}{2} - \frac{\sigma_\eta^2\ln\left(\frac{q_0 p_{\underline{\lambda}^{(0)}}}{q_1 p_{\underline{\lambda}^{(1)}}}\right)}{\rho(0,\underline{\lambda}^{(0)})-\rho(1,\underline{\lambda}^{(1)})}. \tag{17}$$

The assumption made on having specific $\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}$ simplified the likelihood functions in (14) from Gaussian mixtures to single Gaussians, thus enabling the closed-form expression in (17). The intuition for considering a single $\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}$ pair is that the r-axis separation between the Gaussian centers makes one pair of Gaussians dominant in introducing errors, while the others are negligible. In other words, Gaussian mixtures induced by the sneak-path types mix only "lightly". For choosing the pair $\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}$ we need to consider: 1) the type distribution $p_{\underline{\lambda}}$ ranking the likelihoods of the Gaussian centers, and 2) the noise level σ_η determining how wide each Gaussian stretches toward the boundary with the opposite bit hypothesis. Noticing that both $p_{\underline{\lambda}}$ and σ_η appear in (17), we propose the following simplified threshold detector.

Detector 3.

$$\tau' = \min_{\underline{\lambda}}\Big\{\tau\left(\underline{\lambda},(0;0,0)\right)\Big\}, \tag{18}$$

where $\tau\left(\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}\right)$ *is defined in (17).*

We will next justify Detector 3's choices of $\underline{\lambda}^{(1)}=(0;0,0)$ and taking $\underline{\lambda}^{(0)}$ as the type minimizing the threshold. The justification lies on the assumption that $\underline{\lambda}=(0;0,0)$ has the highest $p_{\underline{\lambda}}$ among all types. The detector can be adapted to the case where $(0;0,0)$ is not the most probable, but by the nature of $p_{\underline{\lambda}}$ induced by (6), this happens only when the sneak-path incidence is hopelessly severe. The idea of Detector 3 is to choose the $\underline{\lambda}^{(0)},\underline{\lambda}^{(1)}$ pair that is most dominant in introducing errors. For $\underline{\lambda}^{(1)}$ it is clear that $(0;0,0)$ is the most error dominant because the other types are less likely and farther from the Gaussians of $\underline{\lambda}^{(0)}$. Hence $\underline{\lambda}^{(1)}=(0;0,0)$. For $\underline{\lambda}^{(0)}$ taking the minimum threshold among all $\underline{\lambda}$ types is justified because for a fixed $\underline{\lambda}^{(1)}=(0;0,0)$, the first term of the right-hand side of (15) is monotone decreasing with t, so a low $\tau\left(\underline{\lambda}^{(0)},(0;0,0)\right)$ necessarily means a high second term of the right-hand side of (15), and an error-dominant type $\underline{\lambda}^{(0)}$ that should be chosen.

B. Error Probability

Unlike the MAP detector, the proposed threshold detector, given as Detector 3, has a closed-form expression for the error probability, given the threshold value τ. By marginalizing (14) (with $t=\tau$) over $\underline{\lambda}$, we get

$$P_e^{TH} = \sum_{\underline{\lambda}} p_{\underline{\lambda}}\Big[q_0\Big(1-Q\Big(\frac{\tau-\rho(0,\underline{\lambda})}{\sigma_\eta}\Big)\Big)+q_1 Q\Big(\frac{\tau-\rho(1,\underline{\lambda})}{\sigma_\eta}\Big)\Big]. \tag{19}$$

In Fig. 8 we plot using (19) the error probability of Detector 3 (circle markers) in comparison to the optimal MAP Detector 1 ($\times$ markers, calculated numerically). It can be seen that the error probabilities of both detectors are essentially the same, despite Detector 3 being much simpler to calculate. We also plot in Fig. 8 the error probabilities for the optimal thresholds given four values of $\underline{\lambda}^{(0)}$ (recall that Detector 3 takes the minimum of those thresholds). The threshold-minimizing sneak-path type is $\underline{\lambda}^{(0)}=(1;1,1)$ for σ_η above 10Ω, and for lower noise levels $\underline{\lambda}^{(0)}=(2;2,2)$ is the threshold minimizer.

C. Multiple-Reads Threshold Detector

As done in Section III-C for the MAP detector, we can enhance the performance of the threshold detector by multiple readouts of the same cell. Let us consider again the N measurements of a cell's resistance, denoted by the vector $\underline{r}_{A_{i,j}}$. A threshold detector for a vector of measurements has the form

$$\underline{a}^T\underline{r}_{A_{i,j}} \underset{\hat{A}_{i,j}=1}{\overset{\hat{A}_{i,j}=0}{\gtrless}} C,$$

where $\underline{a}$ is a vector and C is a scalar that together represent the affine hyperplane separating the decision regions. For the MAP vector detector it is seen in (10) that the sum $\sum_{k=1}^{N}\underline{r}_{A_{i,j}}(k)$ is a sufficient statistic for optimal decision. It can be similarly shown that for the optimal threshold detector $\underline{a}^T=(1,\ldots,1)$, and the generalization of Detector 3 to the vector case is

$$\frac{1}{N}\sum_{k=1}^{N}\underline{r}_{A_{i,j}}(k) \underset{\hat{A}_{i,j}=1}{\overset{\hat{A}_{i,j}=0}{\gtrless}} \min_{\underline{\lambda}}\Big\{\tau\left(\underline{\lambda},(0;0,0)\right)\Big\}. \tag{20}$$

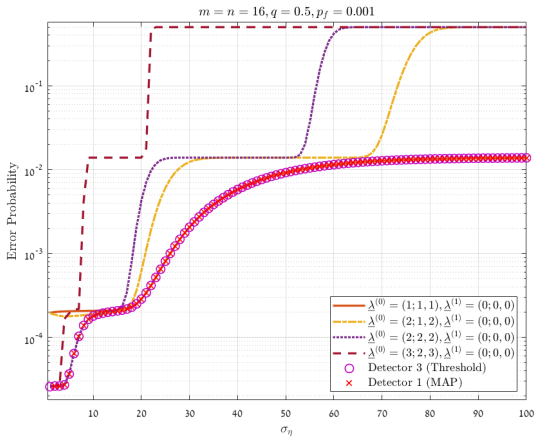


Fig. 8: Error probability of the threshold Detector 3 ($\circ$ markers) vs. the MAP Detector 1 ($\times$ markers). Also plotted are the error probabilities of thresholds calculated for four values of $\underline{\lambda}^{(0)}$. Array parameters are: size 16×16, $q = 0.5$, and $p_f = 10^{-3}$.

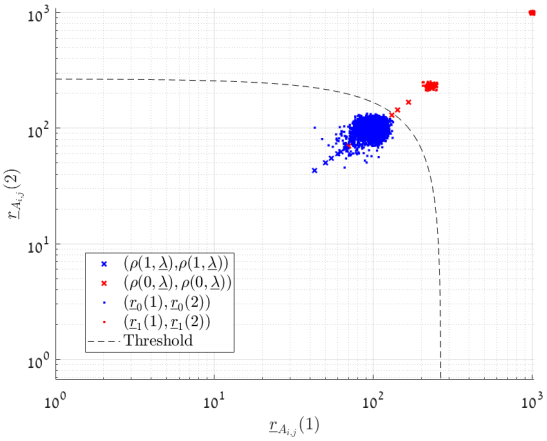


Fig. 9: Multi-read ($N = 2$) threshold detection (simulated for 16×16 array with $\sigma_\eta = 10\Omega$). The dashed curve (affine line in logarithmic scale) is the threshold separating the 1 (lower left) and 0 (upper right) decision regions.

An example of a threshold detector for $N = 2$ is depicted in Fig. 9. The diagonal points marked with $\times$ markers are $(\rho(1,\underline{\lambda}),\rho(1,\underline{\lambda}))$ and $(\rho(0,\underline{\lambda}),\rho(0,\underline{\lambda}))$ for different values of $\underline{\lambda}$. The points marked with $\cdot$ markers are $(\underline{r}_{A_{i,j}}(1),\underline{r}_{A_{i,j}}(2))$ noisy readouts. The vector threshold detector from (20) is marked by the dashed curve (note that it is an affine line transformed to logarithmic axes). The performance of the vector threshold detector is depicted in Fig. 10, where it can be seen that in most noise levels it gives lower error rates than the one-dimensional ($N = 1$) MAP Detector 1 (for $N = 1$ Detector 3 reached the same results as Detector 1). Therefore, by using a single additional measurement we can simultaneously enhance the performance and significantly reduce the computational load of the detection process.

V. Sneak-Path Reducing Code

Although the detectors described throughout Sections III and IV achieve far better performance than standard detection methods (for example, a naïve threshold at $\frac{R(0)+R(1)}{2}$), there is still a need to improve their BER performance. Until now, we have assumed that the sneak-path probabilities are given (based on Theorem 1), and the detection schemes developed above allowed us to reliably extract the written resistance value under interference and noise. However, the performance of any detector highly depends on the actual number of sneak paths affecting the read cell. For good performance it does not suffice to apply the detectors directly on raw data written into the crossbar array. Hence, a pre-write coding scheme that reduces the average number of sneak paths per cell can dramatically enhance the performance of the detector used at read time. In this section we develop such a coding scheme to map the written data to arrays with fewer sneak paths.

Since sneak paths are a data-dependent effect, they 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_1 and q. It is intuitively

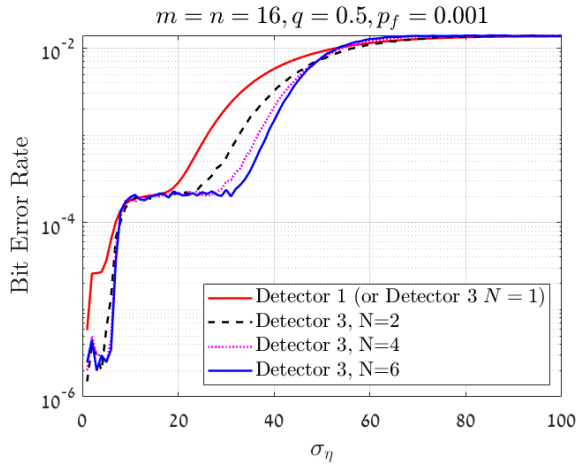


Fig. 10: Comparison of bit-error rates (BER) of single-read MAP detector and multiple-reads threshold detectors. Array size is 16×16 with $q = 0.5$ and $p_f = 10^{-3}$.

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 [25]. The following is a simple adaptation of a theorem from [25].

Proposition 2. *For a cell in an 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*

$$p_L(L') = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} p_{u,v} p_{L'|u,v}, \tag{21}$$

where $p_{u,v} = \binom{m-1}{u}\binom{n-1}{v} q^{u+v}(1-q)^{m-1-u+n-1-v}$ *and* $p_{L'|u,v} = \binom{uv}{L'}(p_f q)^{L'}(1-p_f q)^{uv-L'}$.

Note that Proposition 2 characterizes only the *number* of sneak paths L', rather than their full type $\underline{\lambda}$. We choose in this section to work with the simpler distribution on L to show more clearly how the analysis extends to coded arrays.

The q that gives the sneak-path distribution in (21) also induces a storage rate $\mathcal{R} = \mathcal{R}(q) \leq 1$. To get a better sneak-path distribution than (21) for the same rate $\mathcal{R}$, we next propose a coding scheme based on encoding the array with 2×2 blocks. 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 than the bit-wise shaping method with a single parameter q. 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.

A. 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 set of 7 words, which we denote $\mathcal{C}$, divides 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 the important problem of choosing the probabilities p_0, p_1, p_2 to optimize

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 corresponding probabilities

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

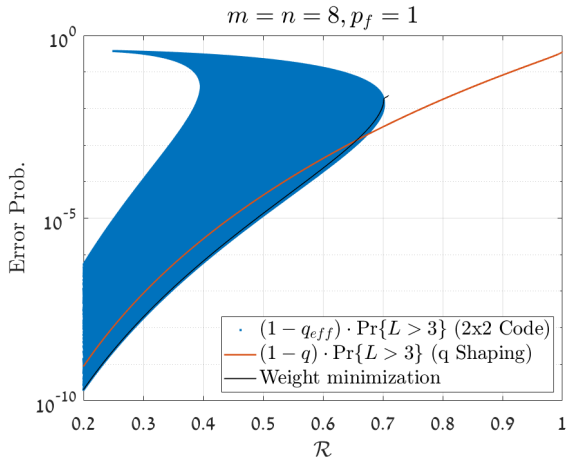


Fig. 11: Sneak-path tail probabilities of 2×2 coding compared to q shaping (with $0 \leq q \leq 0.5$) as a function of storage rate. Painted area represents 2×2 codes with every possible word distributions. The black solid curve depicts the specific distributions selected by weight minimization.

Theorem 3. *Let the 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*

$$p'_L(L') = \sum_{u=0}^{\frac{n}{2}-1} \sum_{v=0}^{\frac{m}{2}-1} p'_{u,v} \cdot p'_{L'|u,v}, \tag{22}$$

where

$$p'_{u,v} = \binom{\frac{n}{2}-1}{u} \binom{\frac{m}{2}-1}{v} (2p_1 + 2p_2)^{u+v} (1 - 2p_1 - 2p_2)^{n/2-1-u+m/2-1-v} \tag{23}$$

and

$$p'_{L'|u,v} = \binom{uv}{L'} [(p_1 + p_2)\, p_f]^{L'} [1 - (p_1 + p_2)p_f]^{uv-L'}. \tag{24}$$

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 can ignore the 1s from the word of the read cell itself, because in none of the legal words a 1 in this word can be part of a sneak path affecting a 0 in the same word. These facts explain the expression for $p'_{u,v}$ in (23). 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 (24). □

The proof of Theorem 3 reveals an important advantage of the 2×2 coding scheme: even though each row, column and their intersection can have one of $7^3 = 343$ word combinations, the symmetries in the problem simplify the analysis and yield compact and wieldy expressions.

Note that for a given storage rate $\mathcal{R}$, there are multiple combinations of word probabilities, p_0, p_1, p_2 that induce that rate. The simplest and most effective method to set the probabilities p_0, p_1, p_2 is *weight minimization*. 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 minimizes the expected word Hamming weight $\mathcal{W}(p_0, p_1, p_2) = 4p_1 + 4p_2$ while satisfying the constraints: 1) $p_0 + 4p_1 + 2p_2 = 1$, 2) $\mathcal{R}(p_0, p_1, p_2) = \mathcal{R}$. In Fig. 11 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. For comparison, we show the same tail probability with simple q shaping (for the same storage rates). Sneak paths only affect read cells that are 0s, thus the tail probabilities are multiplied by $1 - q$ in q shaping and by $1 - q_{eff}$ in the 2×2 code, where q_{eff} is the expected fraction of 1s given p_0, p_1, p_2. Weight minimization is shown to achieve almost the optimal (lowest) tail probability among all p_0, p_1, p_2 distributions, and significantly lower than q shaping.

In Fig. 12 we show error rates obtained in a 2×2-coded array compared to q shaping, for several chosen storage rates $\mathcal{R}$. For both schemes we use the bit-wise MAP Detector 1, with the corresponding distributions of L. The results show a clear

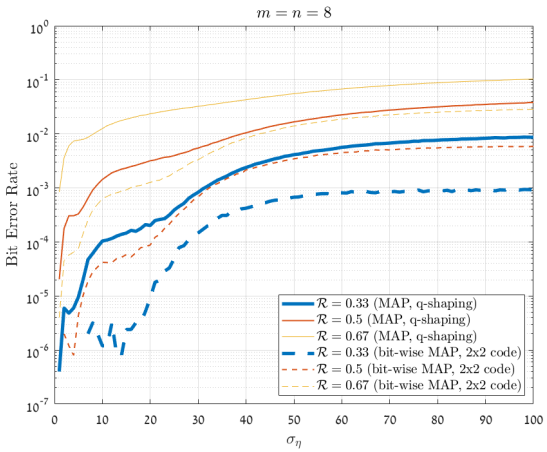


Fig. 12: Bit error rate (BER) of 2×2 coding (solid curves) vs. q shaping (dashed curves) for different storage rates. Array dimensions are $m = n = 8$ and $p_f = 1$. For a given storage rate (≤ 0.7), 2×2 coding is clearly preferred over simple q shaping.

advantage for the 2×2 coding scheme, which improves the error rate by almost an order of magnitude for all of the examined storage rates.

B. 2×2 *Optimal Detector*

Having calculated in Theorem 3 the sneak-path distribution in a coded array allows using the bit-wise detectors from Sections III and IV for reading coded arrays as well. Fig. 12 shows that with the same MAP detector adjusted to the coded sneak-path distribution, coded arrays significantly improve error-rate performance. However, the detection performance in the case of 2×2-coded arrays can be further enhanced by incorporating the code structure into the detector, and deciding jointly on the four bits of the codeword. Instead of deciding on each array bit independently, the mutual dependence of bits in 2×2 blocks can be utilized. Let $\underline{r}$ be a 2×2 matrix that corresponds to a block of resistance values measured in the array. The MAP estimator for the 2×2 codeword stored in the array is given in the following.

Detector 4. *The MAP detector for a* 2×2 *codeword* $\underline{c}$ *is*

$$\hat{\underline{c}} = \arg\max_{\underline{c} \in \mathcal{C}} p(\underline{c}|\underline{r}) = \arg\max_{\underline{c} \in \mathcal{C}} p_{w(\underline{c})} \prod_{i,j=1}^{2} \Lambda_{\underline{c}_{i,j}}(\underline{r}_{i,j}), \tag{25}$$

where $p_{w(\underline{c})}$ *is one of* p_0, p_1, p_2 *of Table II according to the Hamming weight* $w(\underline{c})$ *of the codeword* $\underline{c}$.

Note that the simplicity of Detector 4 is only possible thanks to the special property of the 7-codeword $\mathcal{C}$ that the sneak-path distribution for a bit in a 2×2 block is independent of the bits in the same block. Since in Table II no 1 in a codeword shares a row or a column with another 1 in the same codeword, there can be no sneak paths caused by 1s in the currently-read codeword. As a result, we can decide the MAP codeword assuming a fixed sneak-path distribution that does not depend on the (unknown) codeword itself.

Fig. 13 shows simulation results demonstrating the improvement in error rates of the bit-wise MAP detector and the 2×2 Detector 4 compared to uncoded (q shaping). The improvement is plotted as the ratio (in units of dB) between coded and uncoded BER

$$10 \log_{10} \left(\frac{P_e^{q-shaping}}{P_e^{2\times 2-coded}} \right),$$

where both arrays have the same rate. The difference between the solid curves (Detector 4) and the dashed curves (bit-wise) shows an extra performance advantage for deciding on 2×2 blocks jointly. Overall the 2×2 coding scheme offers significant performance improvement by both suppressing sneak paths and allowing joint detection. This extra performance is obtained without significant complexity cost: changing the basic unit from a single bit to 4 bits, still very small compared to the full array size or the row/column size.

VI. Conclusion

In this paper we provided a formal communication-theoretic treatment for the problem of resistive-array readout with sneak paths. This treatment yielded constructive tools to combat sneak paths in both the detection and coding layers. Some of the ideas

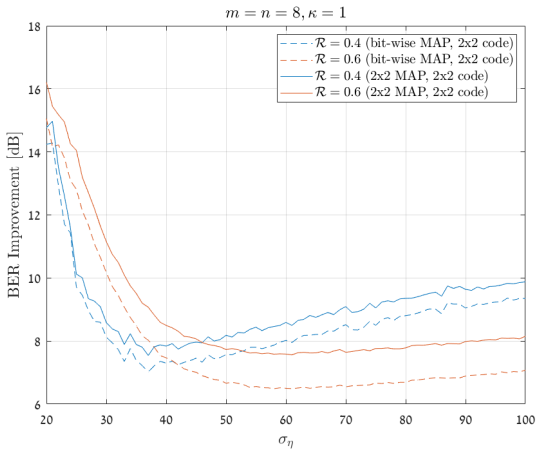


Fig. 13: Bit error rate (BER) improvement for bit-wise detector (dashed) and 2×2 detector (solid). The performance improvement is measured as the ratio of the coded BER to the same-rate uncoded BER (q shaping), in units of dB ($10 \log_{10}(\cdot)$). Array dimensions are $m = n = 8$ and $p_f = 1$.

underlying these tools are admittedly quite natural, but thanks to the rigorous treatment we were able to find simplifications that perform close to the optimal schemes. The same framework can lead in future work to even better tools, and for richer setups capturing more of the realities of memory arrays. In the theoretical direction it is most interesting to further analyze and bound the error probabilities of natural detectors. It is also important to construct codes that meet certain prescribed sneak-path incidence properties, for example an upper bound on the number of sneak paths affecting a read cell. In the practical direction it is useful to apply similar treatment to existing sneak-path mitigation techniques (e.g., the pilot setting in [20]), and then merge together multiple schemes in a bigger solution.

VII. Acknowledgments

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

References

[1] Y. Ben-Hur and Y. Cassuto, "Detection and coding schemes for parallel interference in resistive memories," in *IEEE International Conference on Communications, ICC 2017, Paris, France*, 2017, pp. 1–7. [Online]. Available: https://doi.org/10.1109/ICC.2017.7996547

[2] 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.

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

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

[5] 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.

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

[7] 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.

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

[9] 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.

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

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

[12] ——, "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.

[13] 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.

[14] 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.

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

[16] P. O. Vontobel, W. Robinett, P. J. Kuekes, D. R. Stewart, J. Straznicky, and R. S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, p. 425204, 2009.

[17] 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, no. 6, pp. 2021–2032, June 2012.
[18] C. Liu and H. Li, "A weighted sensing scheme for reram-based cross-point memory array," in *2014 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, July 2014, pp. 65–70.
[19] 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.
[20] R. Naous, M. A. Zidan, A. Sultan-Salem, and K. N. Salama, "Memristor based crossbar memory array sneak path estimation," in *2014 14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*. IEEE, 2014, pp. 1–2.
[21] T. Luo, O. Milenkovic, and B. Peleato, "Compensating for sneak currents in multi-level crosspoint resistive memories," in *2015 49th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2015, pp. 839–843.
[22] P. P. Sotiriadis, "Information capacity of nanowire crossbar switching networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 3019–3032, 2006.
[23] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*. IEEE, 2013, pp. 156–160.
[24] ——, "On the channel induced by sneak-path errors in memristor arrays," in *2014 International Conference on Signal Processing and Communications (SPCOM)*. IEEE, 2014, pp. 1–6.
[25] ——, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 4801–4813, 2016.
[26] R. N. McDonough and A. D. Whalen, *Detection of Signals in Noise*. Academic Press, 1995.