Performance Limit and Coding Schemes for Resistive Random-Access Memory Channels

Guanghui Song, *Member, IEEE*, Kui Cai, *Senior Member, IEEE*, Xingwei Zhong
Jiang Yu, and Jun Cheng, *Member, IEEE*

Abstract—Resistive random-access memory (ReRAM) is a promising candidate for the next generation non-volatile memory technology due to its simple read/write operations and high storage density. However, its crossbar array structure causes a severe interference effect known as the "sneak path." In this paper, we propose channel coding techniques that can mitigate both the sneak-path interference and the channel noise. The main challenge is that the sneak-path interference is data-dependent, and also correlated within a memory array, and hence the conventional error correction coding scheme will be inadequate. In this work, we propose an across-array coding strategy that assigns a codeword to multiple independent memory arrays, and exploit a real-time channel estimation scheme to estimate the instantaneous status of the ReRAM channel. Since the coded bits from different arrays experience independent channels, a "diversity" gain can be obtained during decoding, and when the codeword is adequately distributed over different memory arrays, the code actually performs as that over an uncorrelated channel. By performing decoding based on the scheme of treating-interference-as-noise (TIN), the ReRAM channel over different memory arrays is equivalent to a block varying channel we defined, for which we propose both the capacity bounds and a coding scheme. The proposed coding scheme consists of a serial concatenation of an optimized error correction code with a data shaper, which enables the ReRAM system to achieve a near capacity limit storage efficiency.

Index Terms—ReRAM, sneak path, across-array coding, data shaping

I. INTRODUCTION

Resistive random-access memory (ReRAM) is an emerging non-volatile memory technology that changes the resistance value of a memristor to represent two states of the binary user data: the High-Resistance State corresponding to logic 0 while the Low-Resistance State corresponding to logic 1. The memristor cell is positioned on each row-column intersection of a crossbar structure, which offers a huge density advantage for ReRAM systems [1].

When a cell in a memory array is read, voltage is applied to the memristor cell, and the current flows through the memristor and senses the resistance value. If the memristor is detected with a High-Resistance State, the bit is decided to be a '0'; if it is detected with a Low-Resistance State, the bit is determined to be a '1'. A fundamental drawback of the ReRAM crossbar array is the sneak-path problem [2]. Sneak paths are undesirable paths in parallel to the selected cell for reading. The current goes through the sneak paths and degrades the measured resistance value. Sneak paths are detrimental when a cell with a High-Resistance State is read because the resistance degradation may lead to an erroneous decision.

In the literature, several works [3]–[5] tackled the sneak-path problem by using information and communication theoretical frameworks. In particular, Y. Cassuto *et al.* [3] studied the maximum storage efficiency when the constrained codes are employed to completely avoid sneak paths. This method incurs a high code rate loss, especially when the array size is large. Y. Cassuto *et al.* proved that as the array size approaches infinity, the storage information rate approaches 0 in order to achieve a sneak-path free channel. On the other hand, a commonly used method to eliminate the sneak paths is to introduce a cell selector in series to each array cell. However, these selectors are also prone to failure due to the imperfections of the memory fabrication and maintenance process, leading to the reoccurrence of the sneak paths [4] [5]. Y. Ben-Hur [4] and Zehui *et al.* [5] considered ReRAM systems with imperfect selectors which fail with a certain probability. They built a probabilistic sneak-path model and developed the corresponding data detection schemes. A main challenge for the ReRAM channel is that the sneak-path induced interference to the channel is data-dependent. Moreover, the sneak-path interference is also correlated between different locations of the crossbar array. Previous work [4] developed single-cell data detection schemes that detect the data for each memory cell independently. More sophisticated joint-cell data detection schemes were developed in [5] by introducing pilot cells. However, the probabilistic model in [5] becomes too complex when the array size is getting large. No error correction code (ECC) was employed in previous research works [4] [5].

According to the information theory, an efficient way to achieve reliable data storage is to apply ECC to the system. Such a design should not be a straightforward extension of the conventional ECC designed for the symbol-wise independent and identically distributed (i.i.d.) channels. Other than the channel noise, the code must overcome the sneak-path interference which is data-dependent and also correlated within the crossbar array.

In this paper, we propose efficient coding and decoding schemes for ReRAM channels. To reduce the correlation of the sneak-path interference within a codeword, we propose an across-array coding strategy which spreads a codeword to multiple independent memory arrays, and also exploit a real-time channel estimation scheme to obtain the channel status of each memory array. Since the coded bits from different memory arrays experience independent channels, the overall channel will be averaged and a "diversity" gain will be obtained during decoding. In this way, we can design the

coding scheme over a symbol-wise i.i.d. channel. By further applying the treating-interference-as-noise (TIN) decoding, the ReRAM channel is equivalent to a block-varying channel whose status does not depend on the input data, based on which we derive the lower and upper bounds of its channel capacity and propose a coding scheme. The proposed coding scheme consists of an outer irregular repeat-accumulate (IRA) code being concatenated with an inner data shaper, which is used to change the input data distribution so as to achieve the maximum information rate. A low-complexity joint message-passing decoding for the IRA code and the data shaper is developed based on the state-of-the-art sparse-graph coding theory. With an optimized IRA code, our ReRAM system achieves a near capacity limit storage efficiency.

The rest of this paper is organized as follows. In Section II, we present the ReRAM channel model and describe the data-dependent feature of the sneak-path interference. The across-array coding strategy and the capacity bounds for the ReRAM channel are proposed in Section III. In Section IV, we propose a coding scheme for ReRAM channels and present both numerical and simulation results. In Section V, we conclude the paper.

II. ReRAM Channel Model

Consider an $m \times n$ crossbar memory array. The memristor that lies at the intersection of row i and column j denotes memory cell (i, j). Each array is able to store an $m \times n$ binary data matrix $\boldsymbol{X} = [x_{i,j}]_{m \times n}$, where bit $x_{i,j} \in \{0, 1\}$ is stored in memory cell $(i, j), i \in \{1, ..., m\}, j \in \{1, ..., n\}$. During the writing process, each bit is written into the memory cell by changing the resistance value of the memristor, i.e., for a logical "0" bit, the cell is changed to a high resistance of R_0, referred to as the High-Resistance State, and for a logical "1" bit, it is changed to a low resistance of R_1, referred to as the Low-Resistance State.

During the reading process, the data bit can be detected by measuring the resistance value of the corresponding cell. If it is in the High-Resistance State, the bit is identified as a '0'; if it is in the Low-Resistance State, the bit is identified as a '1'. However, due to the existence of the sneak-path interference, as well as a mix of other noises which can be modeled as an additive Gaussian noise, the memory reading becomes unreliable [3]–[5]. When the cell (i, j) in a memory array is read, certain voltage is applied to the target cell to measure its resistance. A sneak path is defined as a closed path that originates from and returns to location (i, j) while traversing logical-1 cells through alternating vertical and horizontal steps. An example is shown in Fig. 1, where $(3, 2)$ is a target cell for reading and the green line shows the desired path to measure the resistance. However, $(3, 2) \rightarrow (3, 4) \rightarrow (1, 4) \rightarrow (1, 2) \rightarrow (3, 2)$ forms a sneak path (red line) in parallel of the selected cell $(3, 2)$. Since sneak paths always degrade the measured resistance value, they actually benefit the data detection when a cell in a Low-Resistance State (logic 1) is read. The detrimental effect only occurs when a High-Resistance State cell (logic 0) is read, making it more vulnerable to noise. In this paper, we only
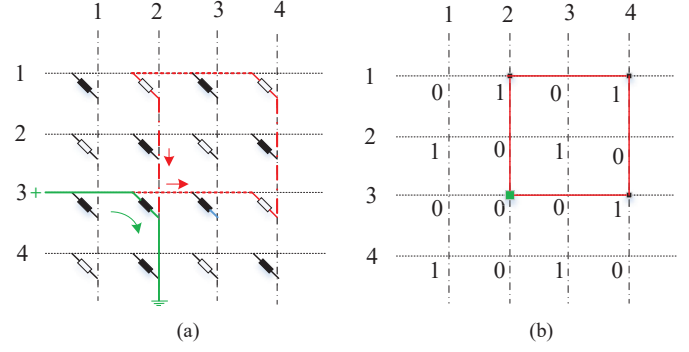

Fig. 1. (a) Example of a 4×4 memory array. (b) Corresponding logical values of memory array. $(3, 2)$ is a target cell for reading. Voltage is applied to memristor cell $(3, 2)$ and green line is the desired path for resistance measuring. However, $(3, 2) \rightarrow (3, 4) \rightarrow (1, 4) \rightarrow (1, 2) \rightarrow (3, 2)$ forms a sneak path (red line) in parallel of target cell $(3, 2)$ that degrades the measured resistance value. Arrows show current flow directions. Note that the sneak path produces a reverse current across cell $(1, 4)$.

consider the sneak path when a High-Resistance State cell is read.

The most popular method to mitigate the sneak-path interference is to introduce a cell selector in series to each array cell. A cell selector is an electrical device that allows current to flow only in one direction across the cell. Since sneak paths inherently produce reverse currents in at least one of the cells along the parallel path, the cell selector can completely eliminate sneak paths from the entire array. In this paper we follow the previous work [4], [5] and consider the 1D1R structure, even though our proposed approaches can be extended to other structures as well. Although cell selectors can effectively eliminate sneak paths, they are also prone to failures due to the imperfections in the production or the maintenance of memory array, leading to the reoccurrence of the sneak paths. Following previous work [4], [5], we assume that the selectors in a memory array fail i.i.d. with probability p_f. However, our work is based on the fact that once a selector fails it will by no means recover, and hence the locations of the failed selectors are fixed during the reading of the whole array. This is different from the assumption made by the previous work [4], [5] that the locations of the failure cells vary randomly in the crossbar array during the reading of each cell. Although these two assumptions may not affect the detection performance significantly for the uncoded ReRAM systems, they are fundamentally different for the coded systems. The previous assumption [4], [5] actually leads to a near i.i.d. model for the sneak-path interference of each cell. In this work, the sneak-path events for the cells at different locations in the same array are highly correlated, which is much more difficult to be tackled by the channel coding scheme.

We define a sneak-path event indicator $e_{i,j}$ for cell (i, j) to be a Boolean variable with the value $e_{i,j} = 1$ if the cell (i, j) is affected by sneak paths, otherwise, $e_{i,j} = 0$. According to the previous work [5], sneak-path events occur during the reading of cell (i, j) and lead to $e_{i,j} = 1$ if and only if the following three conditions are satisfied:

[Sneak-Path Condition:]
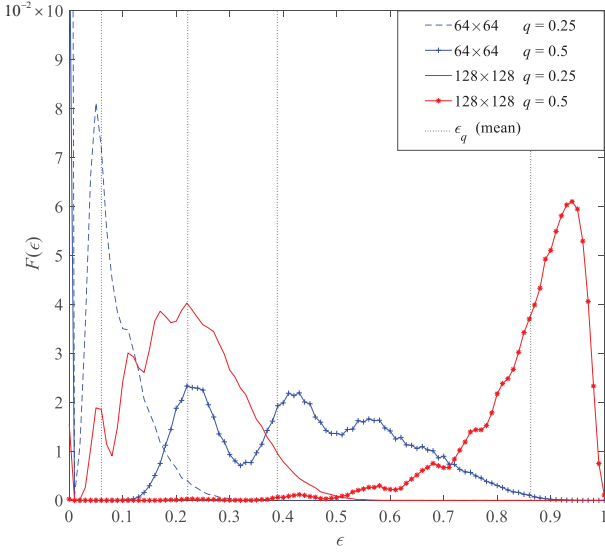1) The cell (i, j) is in a High-Resistance State, i.e., $x_{i,j} = 0$.

Fig. 2. PMFs of sneak-path rate within single memory array simulated for array sizes $m \times n = 64 \times 64, 128 \times 128$ and input distributions with $q = 0.25, 0.5$. The mean values of sneak-path rates for the four cases, indicated by the dashed lines, are $\epsilon_q = 0.06, 0.3888, 0.2216$, and 0.8626, respectively.

2) There exists at least one combination of $k \in \{1, ..., m\}, \ell \in \{1, ..., n\}$ that induces a sneak path, defined by

$$x_{i,\ell} = x_{k,\ell} = x_{k,j} = 1. \tag{1}$$

3) The selector at the diagonal cell (k, ℓ) fails. Due to the circuit structure of the crossbar array, cells (i, l) and (k, j) will conduct current in the forward direction and not be affected by their selectors. Only when the selector of the cell (k, l) is faulty, a sneak path will be formed.

The above Sneak-Path Condition definition limits the sneak path to length of 3, i.e., traversing three cells [5]. More sophisticated cases of the sneak paths were considered in [4]. The principle of our work can be extended to those cases.

Building on the above Sneak-Path Condition, we define a *[ReRAM Channel:]*

Let $X = [x_{i,j}]_{m \times n}$ be the stored data array and $Y = [y_{i,j}]_{m \times n}$ be the corresponding readback signal for the crossbar array. Let $\mathcal{R}$ be the set of real numbers. An ReRAM channel is a channel with input $X \in \{0, 1\}^{m \times n}$ and output $Y \in \mathcal{R}^{m \times n}$:

$$y_{i,j} = \begin{cases} (\frac{1}{R_0} + \frac{e_{i,j}}{R_s})^{-1} + \eta_{i,j} & \text{if } x_{i,j} = 0 \\ R_1 + \eta_{i,j} & \text{if } x_{i,j} = 1 \end{cases} \tag{2}$$

where R_s is the parasitic resistance value brought by sneak paths. Here $\eta_{i,j} \sim \mathcal{N}(0, \sigma^2), i = 1, ..., m, j = 1, ..., n$ is an additive white Gaussian noise (AWGN) with mean 0 and variance σ^2 [4]. It is used to model a mix of various noises of the ReRAM system.

The fundamental problem of the ReRAM channel is to recover the stored data array X based on readback signal Y in the presence of sneak-path interference $[e_{i,j}]_{m \times n}$ and Gaussian noise $[\eta_{i,j}]_{m \times n}$. The ReRAM channel $\{0, 1\}^{m \times n} \to \mathcal{R}^{m \times n}$ with input and output size $m \times n$ actually consists of mn symbol-wise channels with $\{0, 1\} \to \mathcal{R}$. Since the sneak-path indicator

$e_{i,j}$ of each target cell is related to the entire data array, these mn symbol-wise channels are data-dependent and correlated.

The ReRAM channel is also asymmetrical, whose channel status (sneak-path occurring probability) is highly related to the channel input distribution. We define the input distribution as i.i.d. Bernoulli (q), i.e., $\Pr(x_{i,j} = 1) = q$ and $\Pr(x_{i,j} = 0) = 1 - q$ for $i = 1, ..., m, j = 1, ..., n$.

For a fixed input distribution, we investigate the fraction of sneak-path affected cells in the crossbar array and define a sneak-path rate over the array as $\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} e_{i,j}}{mn(1-q)}$. Its mean value is derived as a function of q:

$$\epsilon_q \overset{\Delta}{=} E\left[\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} e_{i,j}}{mn(1-q)} \right] \tag{3}$$

$$= \Pr(e_{i,j} = 1 | x_{i,j} = 0)$$

$$= 1 - \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} \binom{m-1}{u} \binom{n-1}{v} q^{u+v} (1-q)^{m-1-u+n-1-v}$$

$$\times (1 - p_f q)^{uv}. \tag{4}$$

When m or n is large, (4) is difficult to calculate. However, when $p_f q$ is small, using the Taylor expansion $(1 - p_f q)^{uv} \approx 1 - uv p_f q + \alpha \binom{uv}{2} p_f^2 q^2$, we can approximately calculate (4):

$$\epsilon_q \approx 1 - \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} \binom{m-1}{u} \binom{n-1}{v} q^{u+v} (1-q)^{m-1-u+n-1-v}$$

$$\times \left(1 - uv p_f q + \alpha \binom{uv}{2} p_f^2 q^2 \right) \tag{5}$$

$$= (m-1)(n-1) p_f q^3 - \alpha \left(2q \binom{m-1}{2} \binom{n-1}{2} \right.$$

$$\left. + (n-1) \binom{m-1}{2} + (m-1) \binom{n-1}{2} \right) p_f^2 q^5 \tag{6}$$

where α is a balance factor for the last term of the Taylor expansion. Here a good setting for α is 0.8.

We define the probability mess function (PMF) of the sneak-path rate as

$$F(\epsilon) = \Pr\left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} e_{i,j}}{mn(1-q)} = \epsilon \right). \tag{7}$$

For a memory array size of $m \times n = 64 \times 64, 128 \times 128$ and input distributions with $q = 0.25, 0.5$, we simulate $F(\epsilon)$ and the results are illustrated by Fig. 2. In particular, we generate a large amount of input data arrays, compute the sneak-path rate of each array, and obtain the PMF statistically. In the simulations as well as the numerical results of this paper, we assume the selectors fail i.i.d. with probability $p_f = 10^{-3}$. Fig. 2 shows that a larger value of q, or a larger array size, leads to higher sneak-path rates, i.e., worse channels. The values of the sneak-path rate are quite diverse for different input data patterns since the PMF spreads in a large range over the abscissa, which indicates that the channel varies significantly for different input data patterns. This is because the occurrence of sneak-path events depends on the input data pattern. This creates a big challenge for designing the coding scheme for the ReRAM channels since the code directly designed based on the average sneak-path rate of $\epsilon_q = \sum_{\epsilon} \epsilon F(\epsilon)$ will be inadequate.
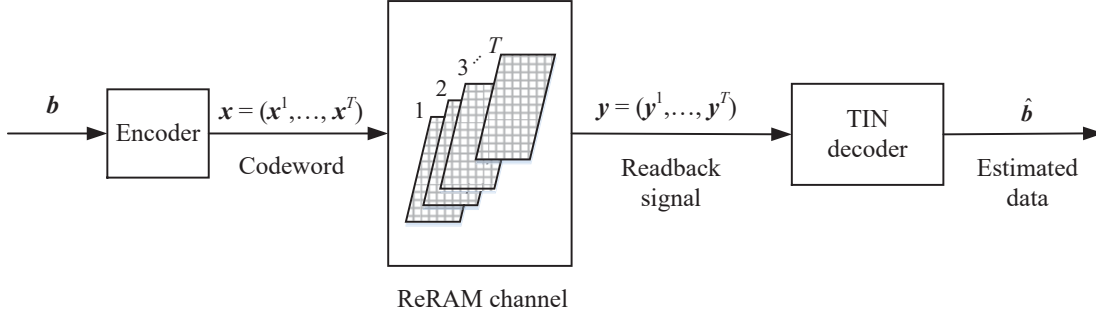
Fig. 3. Across-array coding strategy for ReRAM.

III. ACROSS-ARRAY CODING STRATEGY AND CHANNEL CAPACITY BOUNDS

To mitigate the variability of the ReRAM channel, we propose an across-array coding strategy that assigns a codeword to multiple crossbar arrays. Since the coded bits at different arrays experience independent channels, the sneak-path rate within one codeword will be close to its mean value and hence the channel is closer to an i.i.d. channel. Based on the across-array coding strategy, we further investigate the ReRAM channel capacity bounds. As the sneak path is dependent on the data message, it is difficult to derive the exact capacity of the ReRAM channel. By treating the sneak-path interference as the i.i.d. noise during decoding, the ReRAM channel over multiple memory arrays resembles a block-varying channel that we will define in Section III-B, whose status does not depend on the input data. We then derive the capacity bounds of the block-varying channel, which can be regarded as an approximation of the ReRAM channel capacity.

A. Across-Array Coding Strategy

The proposed across-array coding strategy is illustrated in Fig. 3. Consider the processing of the data vector $\boldsymbol{b} = (b_1, b_2, ..., b_{NR})$, where N is the code length and R is the code rate. Here, $\boldsymbol{b}$ is encoded into codeword $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ which is assigned to T memory arrays, where $N = sT$ for some integer s. Thus, we split $\boldsymbol{x}$ into T equal-length vectors, each of which is assigned to an independent memory array. Without loss of generality, we assign $\boldsymbol{x}^t = (x_1^t, x_2^t, ..., x_{N/T}^t)$ with $x_i^t = x_{(t-1)N/T+i}, i = 1, ..., N/T$, to the t-th memory array for $t = 1, ..., T$. Since each memory array is of size $m \times n$, mnT/N codewords can be stored by these T memory arrays, where mnT/N is assumed to be an integer. As the code rate is R, the storage efficiency is R bits/cell.

Each codeword is decoded independently based on its readback signal. The codeword $\boldsymbol{x} = (x^1, ..., x^T)$ is decoded based on its readback signal $\boldsymbol{y} = (y^1, ..., y^T)$, where y^t is the readback signal of $\boldsymbol{x}^t$ from the t-th memory array, $t = 1, ..., T$, to obtain the estimated data $\hat{\boldsymbol{b}}$. Here we employ a suboptimal decoding scheme known as the TIN decoding [6]. The TIN decoder ignores the correlation between the channel input and the sneak paths and treats the sneak-path interference as the i.i.d. noise.

To illustrate the advantage of across-array coding strategy more explicitly, in Fig 4, we evaluate the PMF of the sneak-path rate over one codeword that is stored in T memory arrays. In the figure, we employ the array sizes and the code lengths of $N = m \times n = 64 \times 64$ and 128×128. Since the coded bits are distributed in T memory arrays, the sneak-path rate as well as its PMF are rewritten as $\sum_{t=1}^{T} \sum_{i=1}^{N/T} e_i^t/(mn(1-q))$ and $F_q^T(\epsilon) = \Pr(\sum_{t=1}^{T} \sum_{i=1}^{N/T} e_i^t/(mn(1-q)) = \epsilon)$, where e_i^t is the sneak-path event indicator during the reading of the i-th bit that belongs to the t-th array. For each case of $m \times n = 64 \times 64$ and 128×128 and the given input distribution with $q = 0.25$ and 0.5, as T increases, the spread of the PMF of the sneak-path rate gets smaller and concentrates closer to the mean value ϵ_q and the channel becomes more stable. The reason is that since the codeword is assigned to T independent memory arrays, the sneak-path rate is averaged over the T arrays. Based on the law of large numbers, as $T \rightarrow \infty$, the sneak-path rate converges exactly to the mean value with probability 1, and therefore, we can design a code based on this mean value to guarantee error free decoding. Note that the across-array coding strategy does not change the channel correlation. It actually reduces the correlation of the sneak-path interference within one codeword since the readback signals for coded bits from different memory arrays are independent with each other. As all the coded bits are encoded from the same message data, this resembles a "code diversity" strategy for block fading channels [7].

The drawback of a joint coding across T arrays is a potential increase of the read/write latency. Note that such additional latency is unavoidable for many sneak-path mitigating approaches in the literature. For example, the multistage reading technique presented in [8] requires three readings and three writings to get a better estimation of the sneak current; [4] investigates the multiple-read detector and shows that it can achieve near-optimum performance with 10 reads. We remark that for our proposed across-array coding strategy, the additional read/write latency can be minimized if a parallel reading/writing circuit [9] is adopted across different crossbar arrays. Moreover, the additional latency incurred will become negligible if the two-dimensional crossbar arrays are stacked to form a 3D structure, which naturally enables the parallel reading/writing across different arrays [10].
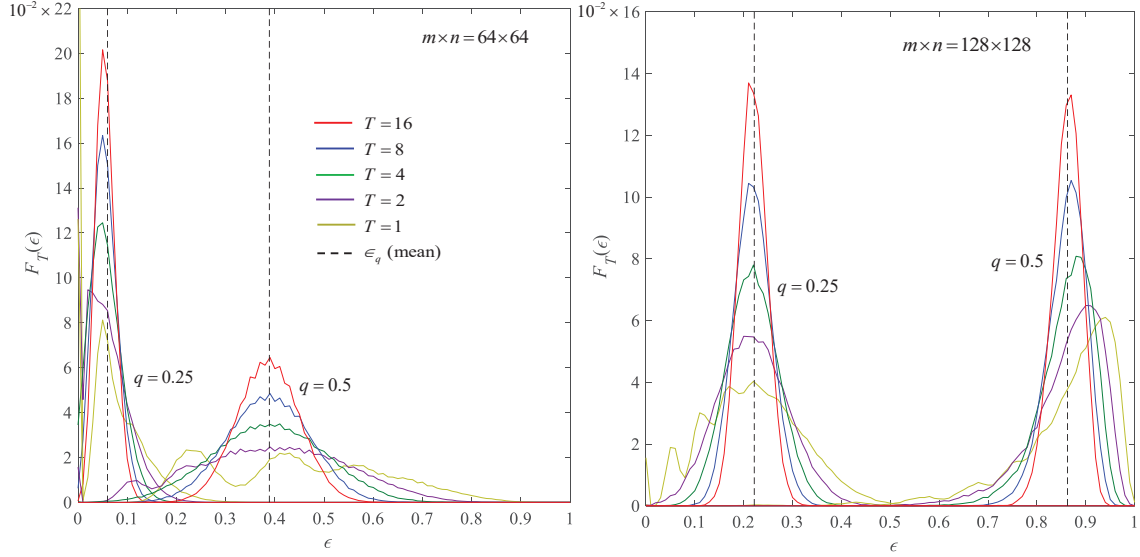
Fig. 4. PMFs of sneak-path rates over one codeword distributed in T memory arrays with $T = 1, 2, 4, 8, 16$. Code length and memory array size are $m \times n = 64 \times 64$ (left) and 128×128 (right).
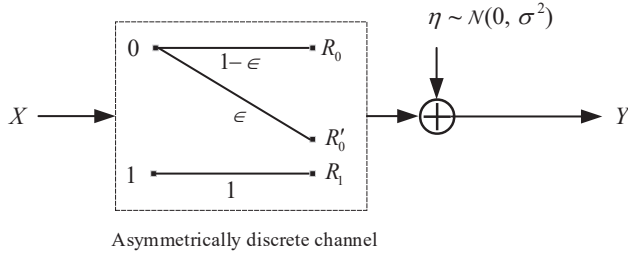

Fig. 5. (ϵ, σ)-channel model with $R_0' = \left(\frac{1}{R_0} + \frac{1}{R_s}\right)^{-1}$.

B. Channel Equivalence and Capacity Bounds

We first define a block-varying (ϵ, σ)-channel and show how the ReRAM channel capacity is related to that of the block varying (ϵ, σ)-channel.

To begin, we first define an (ϵ, σ)-channel. As illustrated in Fig. 5, an (ϵ, σ)-channel is a concatenation of an i.i.d. asymmetrical discrete channel and an i.i.d. additive Gaussian channel, and therefore, it is also an i.i.d. channel without channel correlation. The asymmetrical discrete channel is with binary-input $X \in \{0, 1\}$ and ternary-output from $\{R_0, R_0', R_1\}$ with $R_0' = \left(\frac{1}{R_0} + \frac{1}{R_s}\right)^{-1}$, and the transition property is described by $\Pr(R_0|0) = 1 - \epsilon$, $\Pr(R_0'|0) = \epsilon$, and $\Pr(R_1|1) = 1$. The additive Gaussian channel is with noise distribution $\eta \sim \mathcal{N}(0, \sigma^2)$, whose output $Y \in \mathcal{R}$ serves as the output of the (ϵ, σ)-channel.

For given input distribution $\Pr(X = 0) = 1 - q$, $\Pr(X = 1) = q$, the (ϵ, σ)-channel capacity can be derived as

$$
\begin{aligned}
C_q(\epsilon, \sigma) &= I(X; Y) \\
&= H(Y) - H(Y|X) \\
&= H(Y) - qH(Y|X = 1) - (1 - q)H(Y|X = 0) \\
&= -\int_{-\infty}^{+\infty} p_Y(y) \log_2 p_Y(y) dy - q \log_2 \sqrt{2\pi e \sigma^2} \\
&\quad + (1 - q) \int_{-\infty}^{+\infty} p_{Y|X=0}(y) \log_2 p_{Y|X=0}(y) dy
\end{aligned}
$$

where

$$
\begin{aligned}
p_Y(y) &= (1 - q)\left(\epsilon\phi(y, R_0') + (1 - \epsilon)\phi(y, R_0)\right) + q\phi(y, R_1) \\
p_{Y|X=0}(y) &= \epsilon\phi(y, R_0') + (1 - \epsilon)\phi(y, R_0) \\
\phi(y, m) &= 1/(\sqrt{2\pi}\sigma)e^{-\frac{(y-m)^2}{2\sigma^2}}.
\end{aligned}
$$

The $(0, \sigma)$- and $(1, \sigma)$-channels are asymmetrical binary-input AWGN channels, which are two special cases of an (ϵ, σ)-channel. Obviously, the (ϵ, σ)-channel capacity decreases as ϵ increases leading to $C_q(1, \sigma) < C_q(\epsilon, \sigma) < C_q(0, \sigma)$.

A T-block block-varying (ϵ, σ)-channel with parameters $\boldsymbol{\epsilon} = (\epsilon^1, \epsilon^2, ..., \epsilon^T)$ varies from data block to data block, while within the t-th data block, the channel is a symbol-wise i.i.d. (ϵ^t, σ)-channel, $t = 1, 2, ..., T$. The block-varying (ϵ, σ)-channel is a type of channel with block interference as proposed by [11].

The ReRAM channel over T memory arrays resembles the T-block block-varying (ϵ, σ)-channel. In particular, the sneak-path rate of the ReRAM channel varies from memory array to memory array resembles the block-varying property (the channel varies from block to block) of the block-varying (ϵ, σ)-channel. The channel parameter $\epsilon^t = \Pr(R_0'|X = 0)$ resembles the instantaneous sneak-path rate of the t-th memory array. The main difference is that in the ReRAM channel, the sneak-path interference is dependent on the input data and this

data-dependency leads to the channel correlation, while the parameter ϵ of block-varying (ϵ, σ)-channel is data-independent and the channel within each block is i.i.d. However, if we adopt TIN decoding where the decoder ignores this data-dependency and regards the sneak paths as the i.i.d. noise, an ReRAM channel is equivalent to a block-varying (ϵ, σ)-channel. The memory block length, denoted by M, of the block-varying (ϵ, σ)-channel should be identical to the data array size of the ReRAM channel, i.e., $M = mn$. The channel parameters $\epsilon^t, t = 1, 2, ..., T$ should be i.i.d. generated based on PMF of the sneak-path rate of the ReRAM channel. Therefore, the maximum achievable coding rate over the ReRAM channel under TIN decoding can be approximated by the block-varying (ϵ, σ)-channel capacity.

In preparation to give the capacity limit, we define an (ϵ, σ)-channel code:

Definition 1: An (ϵ, σ)-channel code includes a sequences of codes with rate $C_q(\epsilon, \sigma)$ and different code lengths n, which achieve asymptotical error free decoding over the i.i.d. (ϵ, σ)-channel as the code length approaches infinity.

The existence of the (ϵ, σ)-channel code ensemble is guaranteed by the conventional channel coding theorem.

Consider a block-varying (ϵ, σ)-channel with memory block size M. Parameter ϵ^t has the PMF $F(\epsilon^t), t = 1, 2, ..., T$, and mean value $\bar{\epsilon} = \sum_\epsilon \epsilon F(\epsilon)$, as defined in (3). Since the channel varies from block to block, we consider joint T-block encoding and decoding. The code length is therefore MT. Let R be the encoding rate, and we then have the following theorem:

Theorem 1: For fixed input distribution of Bernoulli (q), as $T \to \infty$, the maximum achievable code rate R with joint T-block encoding and decoding is bounded by

$$C_q(\bar{\epsilon}, \sigma) \le R \le \overline{C}_q(\sigma) \tag{8}$$

where $\overline{C}_q(\sigma) = \sum_\epsilon F(\epsilon) C_q(\epsilon, \sigma)$.

Proof: We first show that for a fixed input distribution of Bernoulli (q), $C_q(\bar{\epsilon}, \sigma)$ is achievable. Let $\boldsymbol{x} = (\boldsymbol{x}^1, ..., \boldsymbol{x}^T)$ be the joint T-block codeword, where t-th block $\boldsymbol{x}^t = (x_1^t, ..., x_M^t)$ experiences an (ϵ^t, σ)-channel, i.e., each symbol $x_j^t, j = 1, ..., M$, experiences an i.i.d. (ϵ^t, σ)-channel. We assume that the codeword is encoded in the way that the i-th bits, located at different data blocks, i.e., $(x_i^1, x_i^2, ..., x_i^T)$, belong to a codeword of a length-T $(\bar{\epsilon}, \sigma)$-channel code. In this way, the original length-MT codeword $\boldsymbol{x}$ can be considered as a vector consisting M length-T $(\bar{\epsilon}, \sigma)$-channel codewords. This is possible because we can always split the uncoded data vector into M equal-length sub-vectors, and encode each of them independently using an $(\bar{\epsilon}, \sigma)$-channel code. As the encoding rate of each $(\bar{\epsilon}, \sigma)$-channel code is $C_q(\bar{\epsilon}, \sigma)$, the overall code rate is $R = C_q(\bar{\epsilon}, \sigma)$.

During decoding, for each $i = 1, ..., M$, $(x_i^1, x_i^2, ..., x_i^T)$ is decoded based on its channel output $(y_i^1, y_i^2, ..., y_i^T)$, where y_i^t is a channel observation of x_i^t. Since coded bit x_i^t experiences an (ϵ^t, σ)-channel, where $\epsilon^t, t = 1, 2, ..., T$, are i.i.d. generated based on the PMF of $F(\epsilon)$, the overall codeword experiences an $(\bar{\epsilon}, \sigma)$-channel where $\bar{\epsilon} = \sum_\epsilon \epsilon F(\epsilon)$. Since $(x_i^1, x_i^2, ..., x_i^T)$ is an $(\bar{\epsilon}, \sigma)$-channel codeword, the decoding error probability approaches 0 as $T \to \infty$ according to Definition 1.

In [11], an upper bound of the block interference channel is proposed. That is, given the channel parameters ϵ, the

channel becomes memoryless, leading to $R \le \frac{1}{MT} I(\boldsymbol{x}; \boldsymbol{y}) \le \frac{1}{MT} I(\boldsymbol{x}; \boldsymbol{x}|\epsilon) \le I(x_1^1; y_1^1|\epsilon^1) = \sum_\epsilon F(\epsilon) C_q(\epsilon, \sigma) = \overline{C}_q(\sigma)$. $\square$

It was also shown in [11] that if the channel state is finite, i.e., ϵ is from a finite set, the upper bound is tight when the block size $M \to \infty$.

Based on our channel equivalence, $\max_q C_q(\epsilon_q, \sigma)$ is an approximate lower bound of the ReRAM channel capacity with TIN decoding, and $\max_q \sum_\epsilon F_q(\epsilon) C_q(\epsilon, \sigma)$ is an upper bound. Since we have the explicit formula (4) for ϵ_q, the lower bound is much easier to be calculated than the upper bound, which requires $F_q(\epsilon)$. Fortunately, we can show that when the array size N is large, the two bounds are numerically very close with each other, and hence the lower bound $\max_q C(\epsilon_q, \sigma)$ can be a good approximation of the ReRAM channel capacity. Fig. 6 illustrates the capacity upper bound $\sum_\epsilon F_q(\epsilon) C_q(\epsilon, \sigma)$ and lower bound $C(\epsilon_q, \sigma)$ as functions of q, for different memory array sizes $m \times n = 32 \times 32, 64 \times 64, 128 \times 128, 256 \times 256$ and different noise values of $\sigma = 30, 50, 100$. The resistance parameters are fixed with $R_1 = 100\ \Omega, R_0 = 1000\ \Omega$, and $R_s = 250\ \Omega$. Observe that when the memory array size N is large, the two bounds are very close with each other.

Fig. 6 also indicates that the ReRAM channel capacity bounds decrease as the data size increases due to the increase of the sneak-path rate, i.e., the larger the data array size the lower the average storage efficiency for each cell, and vice versa. For a very low noise level of $\sigma = 30$, the capacity bounds are maximized at about $q = 0.5$, and for $\sigma = 50, 100$, they are typically maximized when $q < 0.5$. The optimal value of q that maximizes the channel capacity bounds decreases as noise level increases. This is because noise amplifies the detrimental effect of sneak paths, while reducing q effectively reduces the sneak-path rate.

IV. Design of the Coding Scheme

In this section, we present the design of a coding scheme for the ReRAM channel. We also propose a real-time maximum likelihood channel estimation, based on which the message-passing decoding is performed. We utilize the state-of-the-art sparse-graph code and message-passing decoding theories to design the coding scheme, which is essentially an (ϵ_q, σ)-channel code design. A major difference between the (ϵ_q, σ)-channel code and the classical ECC is that since the former works over an asymmetrical channel, the coded data should follow the desired distribution to approach the channel capacity (Fig. 6). We propose a coding scheme, which is a serial concatenation between a classical ECC and a data shaper that shapes the desired data distribution. Bit error rate (BER) simulations and performance comparisons are also presented in this section for our proposed coding scheme.

A. Coding and Decoding Model

A system model for the proposed coding scheme is illustrated in Fig. 7. The encoder includes an ECC encoder and a data shaper. The ECC encoder encodes data vector $\boldsymbol{b} = (b_1, b_2, ..., b_{NR})$ into codeword $\boldsymbol{c} = (c_1, c_2, ..., c_N)$ whose entries are uniformly distributed on $\{0, 1\}$. The data shaper reforms the data distribution into Bernoulli (q). Its output is
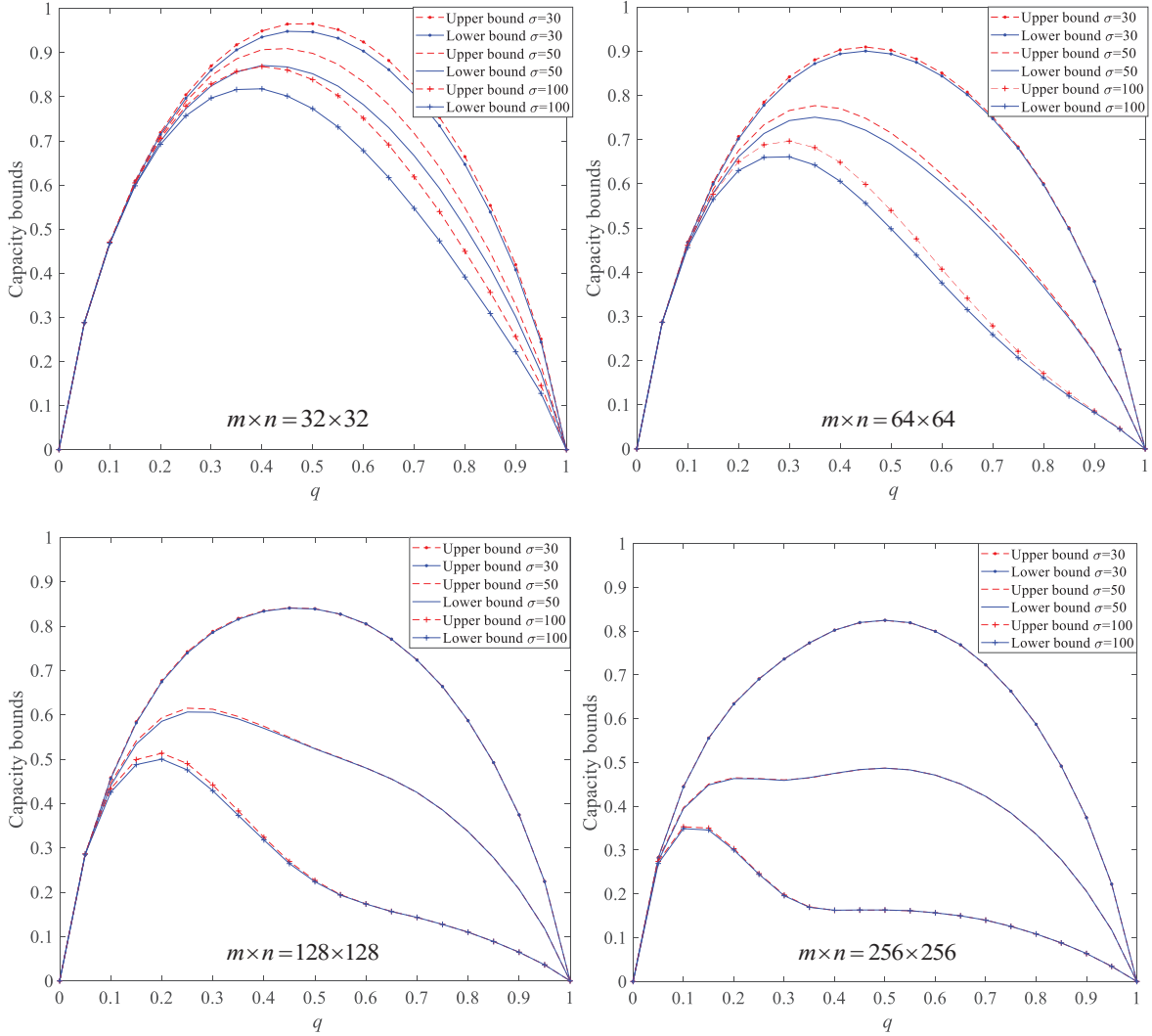
Fig. 6. Upper bound $\sum_\epsilon F_q(\epsilon)C_q(\epsilon,\sigma)$ and lower bound $C(\epsilon_q,\sigma)$ of the capacity as $T \to \infty$, with $R_1 = 100\ \Omega, R_0 = 1000\ \Omega, R_s = 250\ \Omega$, and $p_f = 10^{-3}$.

$\boldsymbol{x} = (x_1, x_2, ..., x_N)$. Here the data shaper in our system has rate-1, and therefore, the overall code rate is still R.

The decoder involves a real-time channel estimator, elementary signal estimator (ESE), a de-shaper, and an ECC decoder. Since the decoding is actually a block-varying (ϵ, σ)-channel decoding, the channel estimator first estimates the channel parameters $\boldsymbol{\epsilon} = (\epsilon^1, ..., \epsilon^T)$ over the T memory arrays based on readback signal $\boldsymbol{y} = (\boldsymbol{y}^1, ..., \boldsymbol{y}^T)$. Based on $\hat{\boldsymbol{\epsilon}}$ and $\boldsymbol{y}$, the ESE calculates a soft estimation $\{L(x_i^t|y_i^t, \hat{\epsilon}^t)\}$, i.e., the log-likelihood ratio (LLR), for each coded bit x_i^t that is used as the decoder input. The decoder consists of a de-shaper and an ECC decoder, both of which use soft-in soft-out (SISO) processings and perform iteratively to improve the decoding reliability. The corresponding decoding is standard message-passing decoding. Specifically, the de-shaper calculates soft LLR $\{L^e(c_i)\}$ for each ECC coded bit, based on which an ECC decoding refines the estimation and feds back an updated LLR $\{L^a(c_i)\}$ to the de-shaper for the next round of decoding iterations. After a fixed maximum number of iterations, a hard decision is performed at the ECC decoder to produce data

estimation $\hat{\boldsymbol{b}}$.

B. Data Shaper

The data shaper consists of a length-L repeater, a length-NL bit interleaver π, and an L-to-1 mapper (Fig. 8). The repeater duplicates each ECC coded bit L times. The bit interleaver permutes the repeater output to relocate the bits. The L-to-1 mapper maps every L bits to one bit, i.e., $\mathcal{M} : \{0, 1\}^L \to \{0, 1\}$. Therefore, the data shaper's overall rate is 1. For the L-to-1 mapper, since there are 2^L patterns for the mapping inputs, by mapping i of them to 1 and $2^L - i$ of them to 0, we obtain the data output with a distribution of Bernoulli $(\frac{i}{2^L})$. By choosing $i = 1, 2, ..., 2^L - 1$, we achieve data distributions of Bernoulli (q) with $q = \frac{1}{2^L}, \frac{2}{2^L}, ..., \frac{2^L-1}{2^L}$.

The interleaver inside the data shaper is crucial in our scheme. Rather than adopting random interleaving, we propose a structured interleaving scheme, as shown in the data shaper's factor graph in Fig. 8. The interleaver π consists of L sub-interleavers $\pi_i, i = 1, ..., L$, each of which can be random. The data-shaping process can be described using a factor graph.
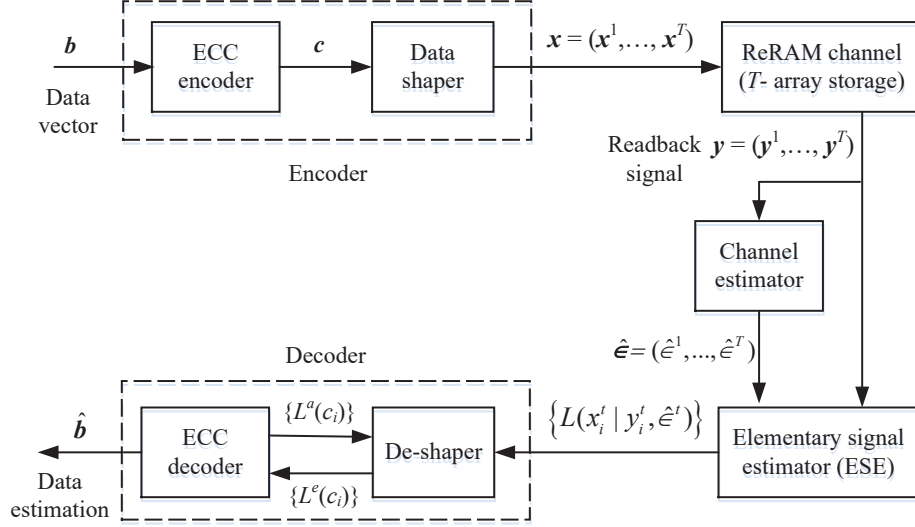
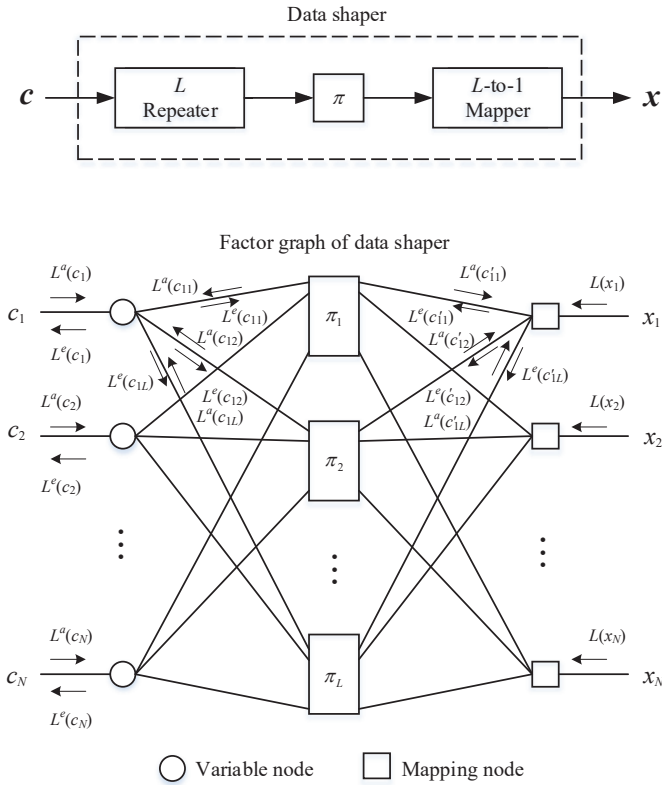Fig. 7. Proposed coding scheme for the ReRAM channel.

Fig. 8. Data shaper and its factor graph illustration.

Fig. 9. 3-to-1 mapping for output data distribution $\Pr(\tilde{x} = 0) = \frac{3}{4}$, $\Pr(\tilde{x} = 1) = \frac{1}{4}$.

Each variable node is associated with an ECC coded bit, where the i-th variable node is associated with c_i. There are L edges from a variable node to the interleavers corresponding to the L repetitions of the ECC coded bit. Each mapping node has L edges from the interleavers corresponding to the L mapping inputs. The i-th mapping node is associated with the mapping output x_i. By using our structured interleaver, the i-th repetitions of the ECC coded bits enter a sub-interleaver π_i whose outputs are used as the i-th inputs of the mapping nodes.
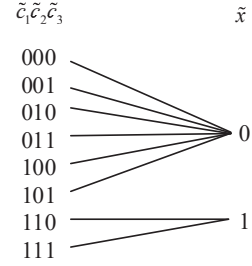
By doing so, each ECC coded bit has exactly one repetition that occupies the i-th input of a mapping node for $i = 1, ..., L$.

The advantage of employing this structured interleaver can be explained using an example. Consider a data shaper with an $(L = 3)$-repeater and a 3-to-1 mapper $\mathcal{M}(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3) = \tilde{x}$ (Fig. 9). There are eight patterns for three binary inputs $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$, where only two of them, 110 and 111, are mapped to 1, and the other six are mapped to 0. If $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$ are i.i.d. with $\Pr(\tilde{c}_i = 0) = \Pr(\tilde{c}_i = 1) = \frac{1}{2}, i = 1, 2, 3$, the mapping can realize output data distribution with $\Pr(\tilde{x} = 0) = \frac{3}{4}$, $\Pr(\tilde{x} = 1) = \frac{1}{4}$. Next we address the de-mapping. Mapping output $\tilde{x}$ actually contains a different quantity of information about the three input bits $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$. By formulating the mapping rule as $\tilde{x} = \mathcal{M}(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3) = \tilde{c}_1 \cdot \tilde{c}_2$, where $\cdot$ is a multiply operation, we evaluate the mutual information between $\tilde{x}$ and each input bit as $I(\tilde{x}; \tilde{c}_1) = I(\tilde{x}; \tilde{c}_2) = \frac{3}{4} \log_2 \frac{4}{3}$ and $I(\tilde{x}; \tilde{c}_3) = 0$. Therefore, if the de-mapper is sufficiently near-optimal, during de-mapping we can obtain information about the first and second bits $\tilde{c}_1, \tilde{c}_2$. Unfortunately, we cannot get any information about the third bit $\tilde{c}_3$ since $\tilde{x}$ does not contain any information about $\tilde{c}_3$. In other words, one-third of the bits are erased after de-mapping. If random interleaving is employed, with probability $\frac{1}{27}$, all the three repetitions of an ECC coded bit will be assigned as the third input of a mapping node and erased. In other words,

with probability $\frac{1}{27}$, an ECC coded bit will be erased after de-shaping, which leads to a poor ECC decoding performance. Our structured interleaving guarantees that all the ECC coded bits can obtain a positive and statistically equal quantity of information from the de-shaper to benefit the ECC decoding.

C. Channel Estimation and ESE

We propose a maximum likelihood channel estimation to obtain parameters $\epsilon = (\epsilon^1, ..., \epsilon^T)$. Since the decoder assumes the channel created by the t-th memory array as an i.i.d. (ϵ^t, σ)-channel, with the channel observation of $\mathbf{y}^t = (y_1^t, ..., y_{N/T}^t)$, the log-likelihood function of ϵ^t is written as:

$$\Lambda(\epsilon^t; \mathbf{y}^t) = \log \prod_{i=1}^{N/T} \Pr\left(y_i^t | \epsilon^t\right) \tag{9}$$

$$= \sum_{i=1}^{N/T} \log \Big[(1-q)\left(\epsilon^t \phi(y_i^t, R_0') + (1-\epsilon^t)\phi(y_i^t, R_0)\right)$$

$$+ q\phi(y_i^t, R_1) \Big]. \tag{10}$$

Next we consider the approximation for (10). Let $\bar{y}_i^t = \arg\min_{x \in \{R_0, R_0', R_1\}} |y_i^t - x|$ be the hard decision value of y_i^t. Since each term of (10) is in the form of $\log \sum_{x \in \{R_0, R_0', R_1\}} p_x \phi(y_i^t, x)$, when the channel noise level is low, it is dominated by the term of $x = \bar{y}_i^t$. We thereby apply

$$\log \sum_{x \in \{R_0, R_0', R_1\}} p_x \phi(y_i^t, x) \approx \log\left(p_{\bar{y}_i^t} \phi(y_i^t, \bar{y}_i^t)\right)$$

$$= \log p_{\bar{y}_i^t} - \frac{(y_i^t - \bar{y}_i^t)^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2),$$

and hence have the following approximation:

$$\Lambda(\epsilon^t; \mathbf{y}^t) \approx n_{R_0'} \log \epsilon^t + n_{R_0} \log(1-\epsilon^t) - \sum_{i=1}^{N/T} \frac{(y_i^t - \bar{y}_i^t)^2}{2\sigma^2} + c$$

where $n_x \triangleq \sum_{i=1}^{N/T} 1\{\bar{y}_i^t = x\}$ is the total number of y_i^t whose hard decision is x and c is a constant term independent of ϵ^t. By maximizing $\Lambda(\epsilon^t; \mathbf{y}^t)$ we obtain the estimation of ϵ^t:

$$\hat{\epsilon}^t = \arg\max_{\epsilon^t} \Lambda(\epsilon^t; \mathbf{y}^t) \approx \frac{n_{R_0'}}{n_{R_0'} + n_{R_0}}. \tag{11}$$

Next, the ESE calculates the LLR for each coded bit of $(\mathbf{x}^1, ..., \mathbf{x}^T)$ based on the readback signal $(\mathbf{y}^1, ..., \mathbf{y}^T)$ and the estimated channel parameters:

$$L(x_i^t | y_i^t, \hat{\epsilon}^t) = \log \frac{\Pr(y_i^t | x_i^t = 0, \hat{\epsilon}^t)}{\Pr(y_i^t | x_i^t = 1, \hat{\epsilon}^t)} \tag{12}$$

$$= \log \frac{\hat{\epsilon}^t \phi(y_i^t, R_0') + (1-\hat{\epsilon}^t)\phi(y_i^t, R_0)}{\phi(y_i^t, R_1, \sigma^2)} \tag{13}$$

for $i = 1, ..., N/T, t = 1, ..., T$. Note that the implementation of ESE (**??**) ignores the correlation between cells in a memory array and regards the sneak-path interference as the i.i.d. noise. A more sophisticated decoding scheme can be developed by utilizing the cell correlation and performing joint data and sneak path detection. It is left as our future work.

To demonstrate the accuracy of the proposed channel estimation, in Fig. 10, we illustrate the mean squared error
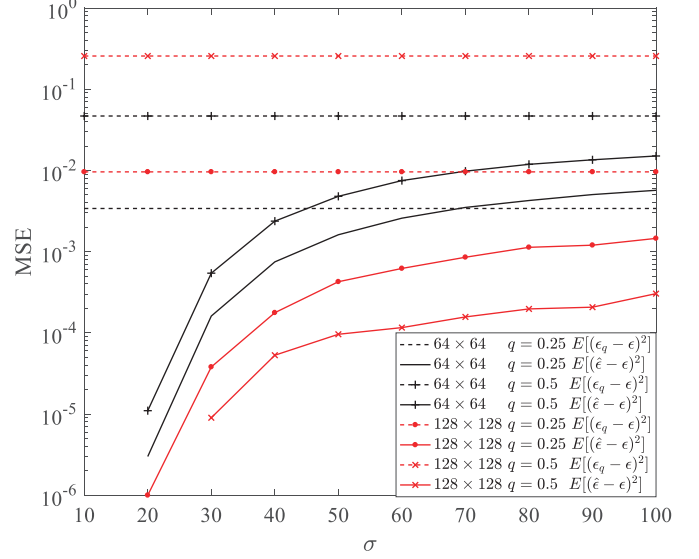

Fig. 10. MSE: $E[(\hat{\epsilon} - \epsilon)^2]$ of our proposed channel estimation (solid lines) and MSE: $E[(\epsilon_q - \epsilon)^2]$ of the average channel parameter (dashed lines).

(MSE) between the estimated and the actual sneak-path rates. We obtain the MSE: $E[(\hat{\epsilon} - \epsilon)^2]$ by simulation for $T = 1$ and memory array sizes $m \times n = 64 \times 64, 128 \times 128$, where ϵ is the actual sneak-path occurrence rate, and $\hat{\epsilon}$ is the estimated value obtained by (11). In general, the MSE is below 10^{-2} and decreases as the channel noise level decreases. For comparison, we also illustrate the MSE: $E[(\epsilon_q - \epsilon)^2]$ between the average and the actual sneak-path rates, where the average sneak-path rate is employed by the decoder when channel estimation is unavailable. Our proposed channel estimation is much more accurate to predict the channel than the average channel parameter, especially for large array size.

D. De-Shaper

The SISO de-shaper can also be realized by message-passing processing over the factor graph shown in Fig. 8. During the de-shaping, each node performs as a local processor and the edges pass LLR messages. The message passing on the edges is bi-directional. The overall processing is performed iteratively. In each iteration, each node in the factor graph acts once. A mapping node performs de-mapping processing based on the L priori LLRs from its neighboring variable nodes and the LLR from ESE and outputs an extrinsic LLR for each mapping input. The extrinsic LLR is used as an a priori LLR for variable node processings. A variable node combines the L priori LLRs from its neighboring mapping nodes and feeds back an extrinsic LLR to each of its neighboring mapping nodes. After a certain number of iterations the variable nodes output a more reliable LLR for each ECC coded bit as the de-shaper output.

1) Mapping Node Processing: A mapping node represents a mapping constraint, i.e., the L edges on the left side link to the L variable nodes that are the L mapping input, and the edge on the right side links to a mapping output. Therefore, the i-th mapping node represents mapping con-

straint $\mathcal{M}(c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}) = x_i$, where $c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}$ are the mapping inputs and x_i is the mapping output. Thus, the edges from the left of a mapping node should pass the LLRs for $c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}$ and the edge from the right side should pass LLR for x_i.

Let $L(x_i) = \log \frac{\Pr(y_i|x_i=0)}{\Pr(y_i|x_i=1)}$ be the LLR about $x_i, i = 1, ..., N$, obtained from the ESE. Let $L^a(c'_{i,j}) = \log \frac{\Pr(c'_{i,j}=0)}{\Pr(c'_{i,j}=1)}$ be an a priori LLR of $c'_{i,j}$ from a variable node. The i-th mapping node calculates an extrinsic LLR for $c'_{i,k}, k = 1, 2, ..., L$, given by

$$
\begin{aligned}
L^e(c'_{i,k}) &= \log \frac{\Pr(y_i|c'_{i,k}=0)}{\Pr(y_i|c'_{i,k}=1)} \\
&= \log \frac{\sum_{c'_{i,j}, j\neq k} \Pr(y_i|c'_{i,k}=0, c'_{i,j}, j\neq k) \prod_{j\neq k} \Pr(c'_{i,j})}{\sum_{c'_{i,j}, j\neq k} \Pr(y_i|c'_{i,k}=1, c'_{i,j}, j\neq k) \prod_{j\neq k} \Pr(c'_{i,j})} \\
&= \log \frac{\sum_{c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}} (1 - c'_{i,k}) \Pr(y_i|x_i) \prod_{j\neq k} \Pr(c'_{i,j})}{\sum_{c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}} c'_{i,k} \Pr(y_i|x_i) \prod_{j\neq k} \Pr(c'_{i,j})} \\
&= \log \frac{\sum_{c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}} (1 - c'_{i,k}) e^{(1-x_i)L(x_i) + \sum_{j\neq k}(1-c'_{i,j})L^a(c'_{i,j})}}{\sum_{c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L}} c'_{i,k} e^{(1-x_i)L(x_i) + \sum_{j\neq k}(1-c'_{i,j})L^a(c'_{i,j})}}
\end{aligned}
$$

where $x_i = \mathcal{M}(c'_{i,1}, c'_{i,2}, \cdots, c'_{i,L})$.

2) Variable Node Processing: Since a variable node is associated with an ECC coded bit, i.e., the i-th node is associated with c_i, the edges connected to it should pass LLR messages for the same bit, i.e., $c_{i,1} = c_{i,2} = \cdots = c_{i,L} = c_i$, where $c_{i,j}, j = 1, ..., L$, are L repetitions of c_i.

Consider the processing at the i-th variable node. Let $L^a(c_i)$ be the priori LLR about c_i from the ECC decoder and $L^a(c_{i,j})$ be the priori LLR about $c_{i,j}, j = 1, ..., L$, from the neighboring mapping nodes. Since $c_{i,1} = c_{i,2} = \cdots = c_{i,L} = c_i$, the variable node calculates an extrinsic LLR for each $c_{i,k}$, given by

$$L^e(c_{i,k}) = L^a(c_i) + \sum_{j\neq k} L^a(c_{i,k}). \tag{14}$$

After a certain number of processing iterations, the variable node outputs an extrinsic LLR about c_i to the ECC decoder

$$L^e(c_i) = \sum_{k=1}^{L} L^a(c_{i,k}). \tag{15}$$

E. ECC Optimization and BER Simulations

In this section, we present the ECC optimization and BER simulation results for ReRAM systems. With our proposed ECC, we achieve a high storage efficiency with a gap of less than 0.1 bit/cell from the ReRAM channel capacity.

For the ECC, we adopt an IRA code which is a type of irregular low-density parity-check (LDPC) code that is able to approach the i.i.d. channel capacity with low encoding and decoding complexity [12], [13]. We consider the code design for two ReRAM systems with a memory array sizes of $m \times n = 64 \times 64$ and 128×128, and at a noise level of $\sigma = 100$. We adopt the data shapers with a ($L = 4$)-repeater and 4-to-1 mappers with Mappings A and B (Fig. 11). Mappings A and B produce data distributions of $q = \frac{5}{16}$ and $q = \frac{3}{16}$, respectively, which approach the maximum storage efficiency for the considered ReRAM systems (Fig. 6). Here
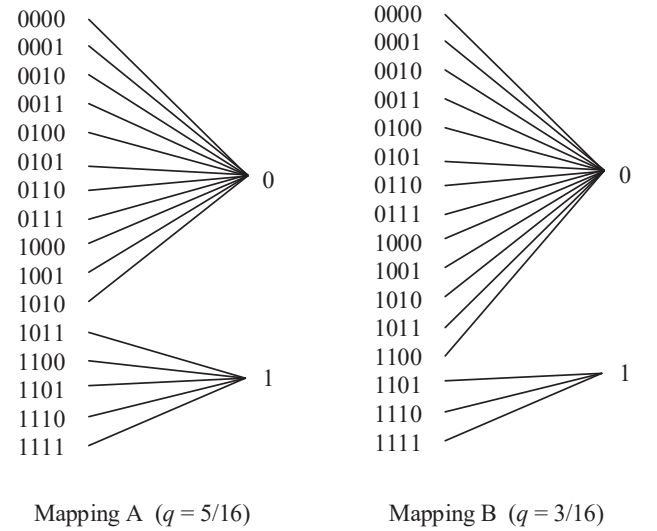

Fig. 11. Mappings A and B that achieved data distributions with $q = 5/16$ and $q = 3/16$.

TABLE I
CODE PARAMETERS FOR $m \times n = 64 \times 64$ AND 128×128 ReRAM SYSTEMS.
PARAMETERS OF IRA CODE INVOLVES ITS VARIABLE NODE DEGREE DISTRIBUTION: $\{\lambda_i\}$
AND COMBINER FACTOR: a [12], [13].

Array size $m \times n$	64×64	128×128
Data shaper	$L = 4$ Mapping A $q = 5/16$	$L = 4$ Mapping B $q = 3/16$
IRA code (ECC)	$\lambda_3 = 0.567736$ $\lambda_{50} = 0.432264$ Combiner: $a = 6$	$\lambda_3 = 0.501564$ $\lambda_{50} = 0.498436$ Combiner: $a = 4$
Code rate	$R = 0.542824$	$R = 0.414735$
$\max_q(C_q(\epsilon_q, \sigma = 100))$	0.660	0.494

the maximum storage efficiencies for $m \times n = 64 \times 64$ and 128×128 ReRAM systems are $\max_q C_q(\epsilon_q, \sigma = 100) = 0.660$ and 0.494 bit/cell, respectively. We optimize the IRA code for these two cases over an i.i.d. (ϵ_q, σ)-channel using a density evolution method [14]. The code parameters for these two ReRAM systems are listed in TABLE I. Our codes achieve $R = 0.542824$ and 0.414735 bit/cell, which are close to the capacity bound with gaps of about 0.12 and 0.08 bit/cell.

In Figs. 12 and 13, we simulate the BER of the two coded ReRAM systems in TABLE I with the across-array coding strategy over both the ReRAM channels (solid lines) and the equivalent block-varying (ϵ, σ)-channels (labeled by $\circ$). Message-passing decoding were employed for both the de-shaper and the ECC decoder, where the decoding of IRA codes can be found [13]. For the $m \times n = 64 \times 64$ system, we employ a code length $N = 64 \times 64 = 4096$ and for 128×128, we adopt $N = 128 \times 128 = 16384$. In both figures, the BERs over the ReRAM channels are almost the same as those over the block-varying (ϵ, σ)-channels, thus demonstrating the proposed channel equivalence. The BERs improve as T increases and approach the decoding performance over the i.i.d. (ϵ_q, σ)-channel, which is the performance limit for ReRAM channels
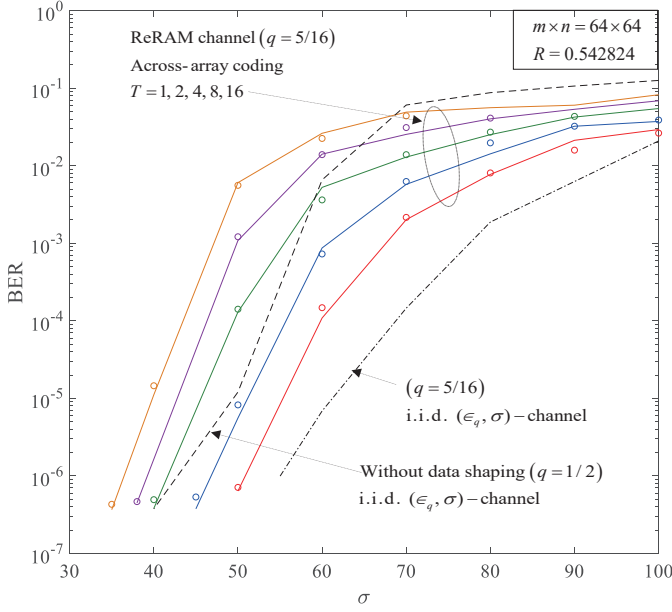
Fig. 12. BERs for IRA-coded ReRAM channel (solid line) and IRA-coded block-varying (ϵ, σ)-channel (labeled by ○) with across-array coding strategy where channel parameters are set as $m \times n = 64 \times 64$, $R_1 = 100$ Ω, $R_0 = 1000$ Ω, $R_s = 250$ Ω, and $p_f = 10^{-3}$.
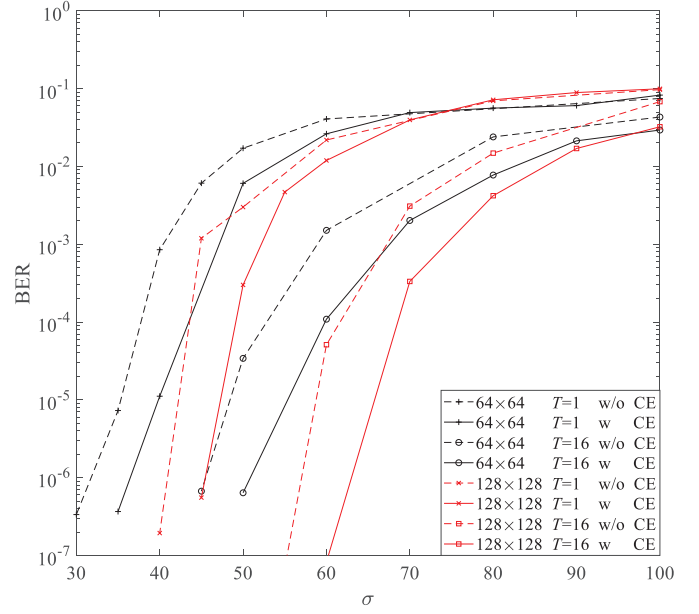

Fig. 14. BER comparison between the IRA-coded ReRAM channel with and without channel estimation (CE), where memory array size is $m \times n = 64 \times 64, 128 \times 128$ and joint-coding array number is $T = 1, 16$.
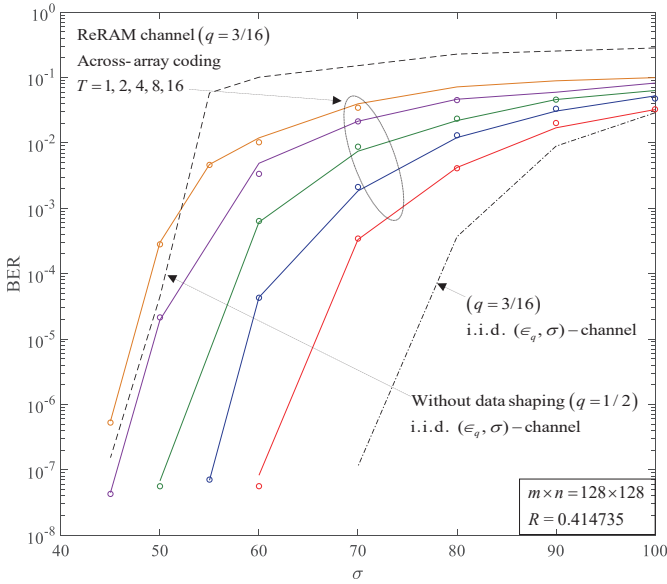

Fig. 13. BERs for IRA-coded ReRAM channel (solid line) and IRA-coded block-varying (ϵ, σ)-channel (labeled by ○) with across-array coding strategy where channel parameters are set as $m \times n = 64 \times 64$, $R_1 = 100$ Ω, $R_0 = 1000$ Ω, $R_s = 250$ Ω, and $p_f = 10^{-3}$.

Fig. 15. BER comparison between our proposed structured interleaving and random interleaving over the i.i.d. (ϵ_q, σ)-channel.

as $T \to \infty$. This performance improvement as T increases can be regarded as a "diversity" gain by assigning the codeword to multiple memory arrays. For comparison, we also illustrate the BER performances of the same IRA codes without data shaping ($q = 1/2$) over i.i.d. (ϵ_q, σ)-channel. The decoding performance deteriorates significantly due to the lack of data shaping.

To emphasize the importance of the proposed real-time channel estimation, in Fig. 14, we compared the BERs be-
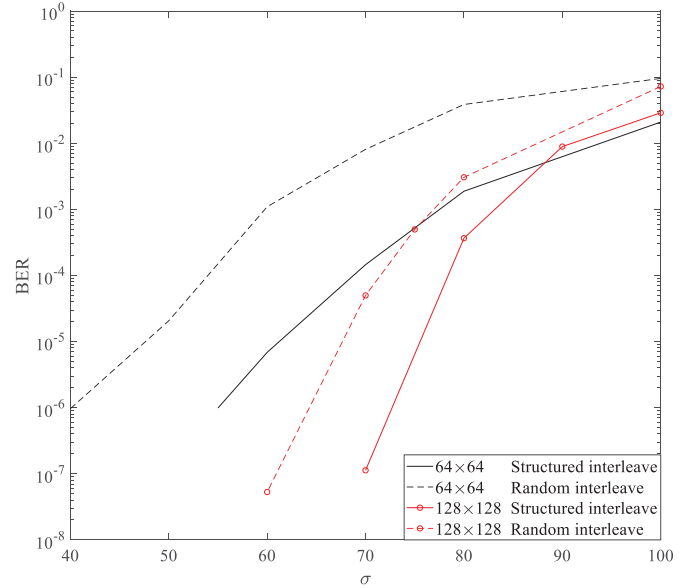
tween the IRA-coded ReRAM channels with and without channel estimation for memory array sizes $m \times n = 64 \times 64, 128 \times 128$ and joint-coding array numbers $T = 1, 16$. We employ the same code parameters listed in TABLE I for both cases. For the IRA-coded ReRAM channel without channel estimation, we employ the average channel parameter ϵ_q for the decoder. We observe that by applying the channel estimation, the BER improvement is obvious for each comparison pair.

Note that the codes in TABLE I are designed at $\sigma = 100$ for the i.i.d. (ϵ_q, σ)-channel, which means that theoretically the

code can be decoded without errors at $\sigma = 100$. However, the actual decoding performances shown in Figs. 12 and 13 are much worse. This is because with density evolution, the code is designed under an infinite code length assumption, while in our simulations finite-length codes were employed. In other words, the codes are asymptotically decodable at $\sigma = 100$ as the code length approaches infinity. For the same reason, the 128×128 ReRAM system achieves a lower BER performance than the 64×64 system since a much longer code is employed in the 128×128 system although the codes in both systems are designed at the same noise level of $\sigma = 100$.

In Fig. 15, we provide BER comparisons between our proposed structured interleaving scheme and the random interleaving for the codes in TABLE I over the i.i.d. (ϵ_q, σ)-channel. For both codes, our proposed structured interleaving scheme outperforms the random interleaving. The BER curves are steeper with structured interleaving than that with random interleaving. This verifies our analysis in Section IV-B. Note that similar performance gain can also be observed from the ReRAM channels, which are the block-varying forms of the (ϵ, σ)-channel.

V. Conclusion

We have considered the design of effective channel coding schemes to tackle both the sneak-path interference and the additive noise for the ReRAM channels. We have proposed an across-array coding strategy to mitigate the channel instability. It also enables a "diversity" gain during decoding. By employing TIN decoding, the ReRAM channel is equivalent to a block-varying channel whose status is not data-dependent, based on which, we proposed the capacity limit as well as a coding scheme. We have also proposed a real-time channel estimation scheme to obtain the sneak-path rates of the T arrays, based on which an ESE calculates the LLR for each coded bit for decoding. To deal with the channel asymmetry, we proposed an ECC concatenated with a data shaper, where the later forms the desired input data distribution to achieve the maximum information rate. With an optimal ECC design, the ReRAM system achieved a high storage efficiency with a gap of less than 0.1 bit/cell from the ReRAM channel capacity limit.

We would also like to point out some possible extensions that lead to our future work. Although we only considered the AWGN noise in this paper, our work can be easily extended to other types of noises, such as the lognormal noise, through reformulating the channel capacity and the LLR formula in the ESE. Moreover, to consider more general sneak-path models that involve multiple sneak paths affecting a read cell, the channel model in Fig. 5 should be modified as a binary input multi-level output channel, where the types of the output signal levels depend on the corresponding sneak-path combinations [4].

References

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

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

[3] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Trans. Inf. Theory.*, vol. 62, no. 9, pp. 4801–4813, Sep. 2016.

[4] Y. Ben-Hur and Y. Cassuto, "Detection and coding schemes for sneakpath interference in resistive memory arrays," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 3821–3833, Feb. 2019.

[5] Z. Chen, C. Schoeny, and L. Dolecek, "Pilot assisted adaptive thresholding for sneak-path mitigation in resistive memories with failed selection devices," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 66–81, Jan. 2020.

[6] P. Huang, P. H. Siegel, and E. Yaakobi, "Performance of multilevel flash memories with different binary labelings: a multi-user perspective," *IEEE JSAC*, vol. 34, no. 9, pp.2336–2353, Sep. 2016.

[7] R. Knopp and P. A. Humblet, "On coding for block fading channels," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp.189–205, Jan. 2000.

[8] 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, Oct. 2009.

[9] J. Zhou, K. Kim, and W. Lu, "Crossbar RRAM arrays: selector device requirements during read operation," *IEEE Trans. Electron Devices*, vol. 61, no. 5, pp. 1369–1376, May 2014.

[10] Q. Luo, X. Xu, T. Gong, and H. Lv, "8-layers 3D vertical RRAM with excellent scalability towards storage class memory applications," *in Proc. IEEE IEDM 2017*, pp. 2.7.1–2.7.4.

[11] R. J. Mceliece and W. E. Stark, "Channels with block interference," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 1, pp.44–53, Jan. 1984.

[12] D. Divsalar, H. Jin, And R. J. Mceliece. "Coding theorems for 'turbo-like' codes." in *Proc. 36Th Allerton Conf. On Communication, Control And Computing*, Allerton, Illinois, Sept. 1998, pp. 201–210.

[13] S. ten Brink and G. Kramer, "Design of repeat-accumulate codes for iterative detection and decoding," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2764–2772, Nov. 2003.

[14] T.J. Richardson and R.L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599 – 618, Feb. 2001.