Title
Coding Assisted Methods for Crossbar Resistive Memory

Author
Chen, Zehui

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Coding Assisted Methods for Crossbar Resistive Memory

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Zehui Chen

2021

ABSTRACT OF THE DISSERTATION

Coding Assisted Methods for Crossbar Resistive Memory

by

Zehui Chen
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2021
Professor Lara Dolecek, Chair

Living in the era of big-data, it is crucial to store vast amounts of data and process them quickly. Resistive random-access memory (ReRAM) with the crossbar structure is one promising candidate to be used as the next generation non-volatile memory device and is also one essential enabler for accelerators that can drastically increase data processing speed. In this work, we tackle problems in crossbar resistive memory and its accelerator application based on channel coding theory and estimation theory.

In Chapter 2, under the non-negligible device variability in practical resistive memory, we treat the problem of Hamming distance computation between two vectors, using low-level conductance measurement, based on a novel *Computation-in-Memory* architecture, which has shown great potential in reducing the burden of massive data processing by bypassing the communication and memory access bottleneck. We study the feasibility problem of Hamming distance computation in-memory under two distinct sources of memristor variability: resistance variation, and the non-deterministic write process. First, we introduce a technique for estimating the Hamming distance under resistance variation. Then, we propose error-detection and error-correction schemes to deal with the non-ideal write process. These

results are then combined to concurrently address both sources of variabilities. Lastly, we demonstrate the efficacy of our approaches on the k-nearest neighbors classifier, a machine learning algorithm that can be accelerated by computing Hamming distance in-memory.

In Chapter 3, considering unreliable selection devices, we study mitigation techniques for the re-occurred sneak-path problem. In a crossbar ReRAM, in which a memristor is positioned on each row-column intersection, the sneak-path problem is one of the main challenges for a reliable readout. The sneak-path problem can be solved with additional selection devices. When some selection devices fail short, the sneak-path problem re-occurs. The re-occurred sneak-path event can be described combinatorially and its adverse effect can be modeled as a parallel interference. Based on a simple pilot construction, we probabilistically characterize the inter-cell dependency of the re-occurred sneak-path events. Utilizing this dependency, we propose adaptive thresholding schemes for resistive memory readout using side information provided by pilot cells. This estimation theoretic approach effectively reduces the bit-error rate while maintaining low redundancy overhead and low complexity.

In Chapter 4, dealing with the increasing resistivity of wordline/bitline in crossbar resistive memory as a result of the scaled down technology node, we propose write/read communication channels under high line resistance and coding theoretic solutions tailored for this channel, targeting the storage class memory (SCM) application. By statistically relating the degraded write/read margins and the channel parameters, we propose binary asymmetric channel (BAC) models for the write/read operations. Method for optimizing the read threshold is proposed to reduce the raw bit-error rate (RBER). Observing a large non-uniformity of reliabilities in the memory array, we propose two schemes for efficient channel coding based on Bose-Chaudhuri-Hocquenghem (BCH) codes. An interleaved coding scheme is proposed to mitigate the non-uniformity of reliability and a location dependent coding framework is proposed to leverage this non-uniformity. Both of our proposed coding schemes effectively reduce the undetected bit-error rate (UBER).

The dissertation of Zehui Chen is approved.

Kang Wang

Lieven Vandenberghe

Jonathan Kao

Lara Dolecek, Committee Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Professor Lara Dolecek for her support of my research. Her guidance helped me in all the time of research. Without her lead, I could not have touched this research topic that is both practically meaningful and interesting to myself. She is also extremely patient with my mistakes in English writing. My writing skills are definitely improved with her help.

I would like to earnestly thank the rest of my dissertation committee: Professor Jonathan Kao, Professor Lieven Vandenberghe, and Professor Kang Lung Wang for their time and invaluable feedback.

My sincere thanks go to Professor Yuval Cassuto for his innovating prior works that attracted me into the study of resistive memory. I also thank Professor Yuval Cassuto for his insightful discussions on related topic during several conferences.

I would like to thank Clayton Schoeny for his enlightening ideas and discussions on my research questions during the starting years of my Ph.D. study. He is my go-to person for research and everything-about-UCLA.

Many thanks from the bottom of my heart go to my wife who wishes to stay anonymous and even not to be mentioned in this acknowledgment due to her modest nature. She is the anchor of my soul and harbor of my heart.

Chapter 1 is a version of [CSCD17] and [CSD18a]. Chapter 2 is a version of [CSD18b] and [CSD19]. Chapter 3 is a version of [CD20] and [CD21].

My Research is supported in part by a grant from UC MEXUS and an NSF-BSF grant no.1718389.

VITA

2016	B.S. (Electrical Engineering), Purdue University, West Lafayette in 2016
2018	M.A. (Electrical Engineering), UCLA, Los Angeles, California.
2018	Coding Theory Engineering Intern, Samsung Semiconductor.
2019	Machine Learning and Algorithm Design Intern, Samsung Semiconductor.
2020	Soc Design Engineer Graduate Intern, Intel.
2016–present	Graduate Research Assistant, Electrical and Computer Engineering, UCLA.

PUBLICATIONS

Z. Chen and L. Dolecek, "Write and Read Channel Models for 1S1R Crossbar Resistive Memory with High Line Resistance," in Proc. IEEE Global Communications Conference (GlobCom), Taipei, China, Dec. 2020.

Z. Chen, C. Schoeny, and L. Dolecek, "Pilot assisted adaptive thresholding for sneak-path mitigation in resistive memories with failed selection devices," IEEE Transactions on Communications, vol. 68, no. 1, pp. 66–81, 2019.

Z. Chen, C. Schoeny, and L. Dolecek, "Coding assisted adaptive thresholding for sneak-path mitigation in resistive memories," in Proc. IEEE Information Theory Workshop (ITW). IEEE, Guangzhou, China, Nov. 2018, pp. 1–5.

Z. Chen, C. Schoeny, and L. Dolecek, "Hamming distance computation in unreliable resistive

memory," IEEE Transactions on Communications, vol. 66, no. 11, pp. 5013–5027, 2018.

Z. Chen, C. Schoeny, Y. Cassuto, and L. Dolecek, "A coding scheme for reliable in-memory hamming distance computation," in Proc. Asilomar Conference on Signals, Systems, and Computers. IEEE, Pacific Grove, CA, Nov. 2017, pp. 1713–1717.

CHAPTER 1

Introduction

1.1 Motivation

In this era of big-data, traditional computer architecture, which is composed of three main components: storage, memory, and processor, faces performance challenges due to the increasing demand of data-intensive applications. On the storage—memory side, the *de facto* memory technology, DRAM, suffers from high leakage current and low density issues, where the main-stream technology for mass storage, NAND flash, incurs high latency and can only be written at a block level [BKS+08]. On the memory—processor side, typically, the processing speed of the processor is much faster than the read latency of the memory, creating a so-called memory-access-bottleneck. This memory-access-bottleneck is the main killer of performance in data-intensive applications as massive amount of data need to be moved from the memory to the processor [HXN+15].

Emerging crossbar resistive memory technology, which is based on a bipolar resistive switching device, referred as memristor [SSS+08], and the crossbar structure [VRK+09], are promising to overcome these aforementioned challenges. Crossbar resistive memory has low leakage current due to its resistive nature and has good scalability and high density due to its simple crossbar structure. It is therefore one of the most suitable candidate to be used as storage-class memory (SCM), a term that refers to memory technology that fills the latency and density gaps between DRAM and NAND flash memory [BKS+08]. Moreover, as data is stored as resistance/conductance, resistive memory enables many basic operations, such as

matrix-vector multiplication [NWY+16] and Hamming distance computation [CC15], to be performed in the memory itself, resulting in a Computation-in-Memory (CIM) architecture [HXN+15]. Accelerators that are built based on the CIM architecture can drastically speed up data intensive applications by bypassing the memory-access-bottleneck.

1.2 Challenges in Crossbar Resistive Memory

As an emerging technology that is still in its immature state, crossbar resistive memory faces many reliability issues that can potentially jeopardize the data integrity in its SCM application and the learning accuracy in its accelerator application. We briefly describe these reliability issues here and a detailed description/modeling of these issues can be found in the individual chapters.

1.2.1 Resistance Variation

In bipolar resistive memory, information is represented by the high resistance state (HRS) or the low resistance state (LRS) of the memory cells. In practical resistive memory, due to cycle-to-cycle variation and device-to-device variation, the state resistances/conductances are distributed in a large range [CL11]. Resistance variation jeopardizes memory reliability, for example, large resistance variation leads to overlapping resistance states which can potentially lead to read error. Using a higher write current or longer write pulse reduce the resistance variation but are costly in terms of energy and latency.

1.2.2 Non-deterministic Write Operation

The switching mechanism of memristors are non-deterministic. Literature shows that the switching time of memristor follows a log-normal distribution [MRPC+11]. The non-deterministic write process suggests that with finite switching time and programming voltage, unsuccessful

switching is inevitable, which results in a write error when the former state of the memristor is different from the data to be written. Increasing the write voltage or the write pulse increases the switching probability at the cost of higher energy consumption and longer write latency.

1.2.3 The Sneak-Path Problem

Under the crossbar structure, memory devices are placed on the intersection of horizontally and vertically placed nanowires. Without selection devices or with failed selection devices, when reading from cell, current can traverse through undesired cells and these paths are referred as the sneak-paths [ZFH+13]. This problem is especially severe when a cell is in HRS as the parallel low resistances in the sneak-paths, lower the sensed resistance and can potentially lead to a read error.

1.2.4 Line Resistance

With technology node scaling down to the single-nm regime, the interconnect resistances of the wordlines and bitlines in the crossbar structure scales up rapidly [LYWW13]. The increasing line resistances reduces the read and write margin of the crossbar resistive memory and therefore degrades the reliability of the write and read operations. The line resistance also limits the array size which jeopardizes the scalability of crossbar resistive memory.

1.3 Research Overview

As detailed in the two previous sections, crossbar resistive memory is promising in its SCM and in-memory accelerator applications while facing many challenges arising from device and structure nonideality. Research efforts from both a device perspective and a system perspective are therefore needed in order to overcome these challenges. In this dissertation,

from a system perspective, aiming to contribute toward the maturation of crossbar resistive memory, we focus on a single objective in the following three chapters: using system level approaches to improve the robustness of crossbar resistive memory in its application under these reliability issues. Based on communication theory, we first model these reliability issues. Then we propose estimation theoretic and coding theoretic approaches to improve the system level reliability metrics, i.e., the raw bit-error (RBER) and undetected bit-error rate (UBER) in the SCM application, and the learning accuracy in the in-memory accelerator applications. The reliability issues addressed in each chapter, the proposed schemes and the targeted application are summarized in the following table.

Table 1.1: A brief overview of chapters in the dissertation

Chapter #	Proposed Schemes	Reliability Issues	Application
2	Hamming Distance -Estimation In-Memory Error Detection/Correction	Resistance Variation Non-deterministic Write	Accelerator
3	Adaptive Thresholding	Sneak-Path Problem Resistance Variation	SCM
4	Read Threshold Optimization Interleaved Coding Location Dependent Coding	Line Resistance Resistance Variation Non-deterministic Write	SCM

CHAPTER 2

Hamming Distance Computation in Unreliable Resistive Memory

2.1 Introduction

With many emerging data-intensive applications, it has become imperative to have the means to store and quickly process vast amounts of high dimensional data. However, current computer architectures largely suffer from communication and memory access bottlenecks. Additionally, CMOS technology faces limited scalability issues [CZ14, HXN+15]. Resistive Random-Access Memory (ReRAM), also simply known as resistive memory, has been shown to have promising scalability with novel crossbar structure, and is thus a promising candidate for next generation none-volatile memories [SSS+08]. Enabled by this new technology, Computation-in-Memory (CIM) Architecture has been proposed, in which certain computations are performed in the physical memory itself [HXN+15]. This idea allows us to bypass the communication and memory access bottlenecks and can be used to speed up data-intensive applications.

Computing similarity metrics between vectors is a critical component in machine learning algorithms; image recognition and natural language processing are just some of many examples. It has already been shown that hashing higher dimensional data into binary space, and using Hamming distance as the distance metric is well suited for large-scale applications [KD09, NPF12]. Allowing for Hamming distance to be computed under the CIM Architecture is thus a promising technique to speed up many modern machine learning appli-

cations. Recently, a technique has been proposed to compute Hamming distance in resistive memory, assuming an ideal model for the memristors [CC15]. Following from their work, in this chapter, we use more sophisticated models to improve and refine their scheme (to which we refer as Hamming distance Computation-in-Memory (HD-CIM)). The feasibility of HD-CIM is studied under two main (and complementary) sources of memristor variability: resistance variation, and the non-deterministic switching mechanism during the memristor write process. In order to preserve the low-latency property of HD-CIM, we assume *limited accessibility to information*. First, only vector-level conductance measurement can be used, and second, only one measurement can be used per Hamming distance computation. Dealing with the two sources of memristor variability is thus more challenging due to these limited accessibility assumptions, in particular, traditional ECCs based on bit-level information do not apply. Simple yet effective solutions are proposed to deal with these sources of memristor variability when the information that can be read is limited. As suggested in [CC15], the optimal code that maps a message to a constant weight codeword while preserving the Hamming distance is used as the foundation for our solutions. The efficacy of our approaches under these sources of memristor variability are studied in detail for the k-nearest neighbors classifier, one promising application for HD-CIM.

The outline of Chapter 2 is as follows. Section 2.2 provides background on resistive memory, memristors, and the two sources of memristor variability investigated in this chapter. The two sources of memristor variability, variability due to resistance variation and variability due to non-deterministic switching mechanism, are elaborated in detail. We highlight the trade-offs between these sources of variability and performance metrics of resistive memory in order to emphasize the necessity of concerning these sources of variability. The mathematical models of the adverse effects these variability sources are established in this chapter. The solutions that deal with the adverse effects the variability sources are at first studied separately. Section 2.3 provides a solution to estimate the Hamming distance from a single conductance measurement in resistive memory where the resistances of memristors

are formulated as Gaussian random variables. To generalize our solution, inversion coding is used to provide *a priori* knowledge about vector weights. The estimation error probability and its bounds are also studied in Section 2.3 for the case with inversion coding and the case without inversion coding. In order to evaluate the effect of resistance variation on the end application, we introduce the average-case study framework. Using this framework, we present the performance of our estimation scheme on the k-nearest neighbors classifier as our main application. Section 2.4 provides error-detection and error-correction schemes as solutions to bit-errors caused by the non-ideal write process. Single error detection is first studied and then we generalize the same idea to multiple errors detection. Two different error correction schemes are explored in this chapter. The exact error correction scheme is provided based on the inherent error localization capability of inversion coding and an extra code that correct erasures. Motivated by the error tolerant nature of our application, we also provides an approximate error correction scheme that have low latency and low overhead. The performance of this approximate error correction scheme is evaluated under the average-case study framework for the k-nearest neighbors classifier. Section 2.5 combines results of Section 2.3 and Section 2.4 and provides a feasible scheme to estimate Hamming distance in resistive memory under the adverse effects of the two sources of memristor variability simultaneously. The joint effect of the two variability sources is first studied and the maximum *a posteriori* probability estimator is formulated. We then present a solution to the MAP estimator. Adapting the concept of approximate error correction scheme discussed in the last chapter, we provide a scheme that achieve estimation and error-detection/correction simultaneously.

2.2 Backgrounds

2.2.1 Memristors and Resistive Memory

In this chapter, we focus on resistive memory which uses a crossbar structure [VRK+09]. In crossbar resistive memory, a resistive switching device (also referred to as a memristor) is placed at the intersection of each row and column [VRK+09]. The logical states of "0" and "1" are represented by the internal resistance state of memristor, "High Resistance State (HRS)" and "Low Resistance State (LRS)," respectively. The state of the memristor can be programmed by applying different voltages to its terminals and can be sensed by measuring the corresponding current. In this chapter, under the same model used in [CC15], we are interested in inferring the Hamming distance between vectors from the conductance measurement between rows of memristors where the two vectors are stored. Figure 2.1 shows an example of the circuit representation that stores two vectors, $\boldsymbol{x} = (0, 0, 0, 1, 1)$ and $\boldsymbol{y} = (0, 1, 1, 0, 1)$, and the example conductance measurement between the two vectors, $G(\boldsymbol{x}, \boldsymbol{y})$. Using conductance in our calculations (instead of resistance) allows for a simple summation of each branch. We assume throughout this chapter that measurements between the two rows of memristors in resistive memory can be reliably performed.

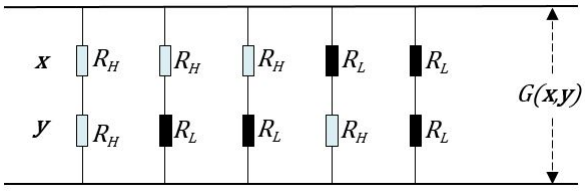


Figure 2.1: Example of an equivalent circuit for a measurement between two vectors.

2.2.2 Variability Due to Resistance Variation

While an on/off ratio from ~ 10 to above 1000 has been shown in many ReRAM papers as proof of a large memory operation window, the variations of HRS and LRS are both common and significant [CL11]. Previous work on HD-CIM assumed constant valued LRS and HRS resistances [CC15]. It is therefore of interest to take resistance variation into account and develop corresponding schemes in order to perform Hamming distance computation in practical resistive memory. In practice, the resistance variation is affected by many operational parameters. It is reported that LRS variability depends on two main parameters, the write current limit I_{limit}, and the write pulse width [CL11]. A smaller I_{limit} favors low-power operation but increases the variation of LRS. A shorter pulse width favors fast speed operation but it also increases the variation of LRS. As a result, studying HD-CIM solutions that tolerate resistance variation not only makes HD-CIM feasible in practice but also provides useful insight into the trade-off between performance and operation parameters, i.e., pulse width and I_{limit}.

Many papers have shown that the resistance distribution of LRS and HRS are Gaussian-like [BKL+05, CLG+11, WLW+10]. The reported resistances of LRS and HRS are both positive and large, i.e., on the order $k\Omega$ and $M\Omega$. It is therefore reasonable to assume that the conductance also follows a Gaussian distribution if the corresponding resistance is Gaussian-like. We assume that the process variability in each memristor is identical and independent. For a memristor i, let L_i and H_i be random variables denoting the state "0" conductance and the state "1" conductance, respectively. We assume L_i and H_i follow the following Gaussian distributions:

$$L_i \sim \mathcal{N}(\mu_L, \sigma_L^2),\ H_i \sim \mathcal{N}(\mu_H, \sigma_H^2). \tag{2.1}$$

In the above model, μ_L and μ_H are the mean of state "0" and state "1" conductance, and σ_L^2 and σ_H^2 are the variance of state "0" and state "1" conductance, respectively. We define $\epsilon = \mu_L/\mu_H$ which is also the "On/Off" resistance ratio, a key characteristic of memristor

devices [VRK+09].

2.2.3 Variability Due to Non-deterministic Switching Mechanism

Similar to the resistance of memristor, the switching mechanism of the memristor is also non-deterministic. It is reported that the switching time of some memristors, e.g., TiO_2 cells, follows a log-normal distribution with the median switching time exponentially dependent on the external voltage [MRPC+11]. Either increasing the programming voltage or increasing the switching time will increase the switching probability. Meanwhile, increasing the programming voltage will increase energy consumption and increasing the switching time will slow down the write progress and increase energy consumption [NXX12, YPQ+11]. With finite switching time and programming volatage, instances of unsuccessful memristor switching are inevitable; studying the effect of the non-deterministic switching mechanism in HD-CIM and the corresponding solution is thus necessary to make HD-CIM practical.

Unsuccessful write operations lead to bit errors when the former state of a memristor is different from the data to be written. We thus model this variability as a binary symmetric channel (BSC) when writing to resistive memory, i.e., a write BSC. The vectors to be stored are viewed as the input to this channel, while the vectors actually written to the resistive memory are the output.

2.3 Hamming Distance Estimation-In-Memory Under Resistance Variation

In 1.3.1, we provide a scheme (Hamming Distance Estimation-In-Memory) to estimate the Hamming distance between two vectors whose Hamming weights are *a priori* known, assuming the conductance of each memristor state is modeled as a random variable. We also provide a simplified scheme in the regime of small "On/Off" resistance ratios. In 1.3.2, an inversion coding technique is used to generalize Hamming distance Estimation-In-Memory

to be applicable to arbitrary vectors with unknown Hamming weights, using a single conductance measurement. We then prove the optimality of this code and present a modified estimation scheme. In 1.3.3, we provide analysis of the estimation error and show that inversion coding also provides benefits in terms of estimation accuracy. In 1.3.4, the average-case study framework is introduced to study the effect of resistance variation on the k-nearest neighbors classifier when our estimation scheme is used.

2.3.1 Hamming Distance Estimation-In-Memory (HD-EIM)

For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ stored in resistive memory where conductances of the two states of each memristor are modeled as Gaussian random variables, the conductance between the two rows of memristors is as follows:

$$G(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} \left[x_i y_i \frac{H_{i,\boldsymbol{x}} H_{i,\boldsymbol{y}}}{H_{i,\boldsymbol{x}} + H_{i,\boldsymbol{y}}} + x_i(1-y_i)\frac{H_{i,\boldsymbol{x}} L_{i,\boldsymbol{y}}}{H_{i,\boldsymbol{x}} + L_{i,\boldsymbol{y}}} + (1-x_i)y_i \frac{L_{i,\boldsymbol{x}} H_{i,\boldsymbol{y}}}{L_{i,\boldsymbol{x}} + H_{i,\boldsymbol{y}}} + (1-x_i)(1-y_i)\frac{L_{i,\boldsymbol{x}} L_{i,\boldsymbol{y}}}{L_{i,\boldsymbol{x}} + L_{i,\boldsymbol{y}}} \right]. \tag{2.2}$$

Here, x_i, y_i denote the i-th entry of vectors $\boldsymbol{x}, \boldsymbol{y}$ respectively. $H_{i,x}$ and $H_{i,y}$ are random variables denoting the "1" state conductances of memristors that store x_i and y_i respectively, and they are modeled as Gaussian random variable in (2.1). $L_{i,x}$ and $L_{i,y}$ are similarly defined for the "0" state conductances. Equation (2.2) follows by summing up the conductances of each branch i, which is composed of the serial conductance of memristors that store x_i and y_i. Note that Equation (2.2) is analogous to Equation (1) in [CC15] with the substitution of our new variability model.

The following approximation can be made when $\mu_H >> \sigma_H$ and $\mu_L >> \sigma_L$:

$$\begin{aligned} &\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \quad \frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8), \\ &\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right). \end{aligned} \tag{2.3}$$

These approximations are made in order to facilitate further calculation (see Appendix A for justification of approximations). We define $\epsilon = \mu_L/\mu_H$ and provide examples of ϵ in Appendix A, Table A.1.

For two vectors $\boldsymbol{x}$, $\boldsymbol{y} \in \{0,1\}^n$, we define N_{00} to be the number of element pairs that have $x_i = y_i = 0$ for $i \in \{1, ..., n\}$. Similarly we define N_{01} to be the number of element pairs that have $x_i = 0, y_i = 1$, N_{10} to be the number of element pairs that have $x_i = 1, y_i = 0$, and N_{11} to be the number of element pairs that have $x_i = y_i = 1$. Using our approximations, the conductance measurement can be expressed as follows,

$$G(\boldsymbol{x}, \boldsymbol{y}) \approx \bar{G}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N_{11}} F_i + \sum_{i=1}^{N_{01}+N_{10}} S_i + \sum_{i=1}^{N_{00}} T_i,$$

We normalize F_i, S_i, and T_i by $\mu_H/2$ and denote the normalized random variables by f_i, s_i, and t_i, (where $\epsilon = \mu_L/\mu_H$) yielding:

$$f_i \sim \mathcal{N}(1, \sigma_H^2/2\mu_H^2),\ s_i \sim \mathcal{N}\left(\frac{2\epsilon}{1+\epsilon}, \frac{4\sigma_L^2}{\mu_H^2(1+\epsilon)^4}\right),\ t_i \sim \mathcal{N}(\epsilon, \sigma_L^2/2\mu_H^2).$$

Similarly, we compute the normalized conductance measurement, $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$, as follows:

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\bar{G}(\boldsymbol{x}, \boldsymbol{y}) \times 2}{\mu_H} = \sum_{i=1}^{N_{11}} f_i + \sum_{i=1}^{N_{01}+N_{10}} s_i + \sum_{i=1}^{N_{00}} t_i.$$

Using elementary properties of independent Gaussian distributions, we have

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(N_{11} + (N_{01} + N_{10})\frac{2\epsilon}{1+\epsilon} + N_{00}\epsilon, \frac{N_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N_{01} + N_{10})8\sigma_L^2}{2\mu_H^2(1+\epsilon)^4} + \frac{N_{00}\sigma_L^2}{2\mu_H^2}\right). \quad (2.4)$$

We define w_x and w_y to be the *a priori* known Hamming weights of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. We define $D(\boldsymbol{x}, \boldsymbol{y})$ to be the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. Using the following facts,

$$N_{11} = [w_x + w_y - D(\boldsymbol{x}, \boldsymbol{y})]/2,\ N_{01} + N_{10} = D(\boldsymbol{x}, \boldsymbol{y}),\ N_{00} = n - N_{11} - N_{01} - N_{10}, \quad (2.5)$$

we can compute the following:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1+\epsilon}{(1-\epsilon)^2}[(1-\epsilon)(w_x + w_y) + 2n\epsilon - 2\tilde{G}(\boldsymbol{x}, \boldsymbol{y})]. \quad (2.6)$$

Note that Equation (2.6) is a substitution of the noisy $D(\boldsymbol{x}, \boldsymbol{y})$ into Equation (5) from [CC15]. Based on (2.4), (2.5) and (2.6), $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ has the following distribution:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\bigg(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2(1+\epsilon)^2 N_{11}\sigma_H^2}{\mu_H^2(1-\epsilon)^4} + \frac{(N_{01}+N_{10})16\sigma_L^2}{\mu_H^2(1+\epsilon)^2(1-\epsilon)^4} + \frac{2(1+\epsilon)^2 N_{00}\sigma_L^2}{\mu_H^2(1-\epsilon)^4} \bigg). \quad (2.7)$$

$\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is an intermediate random variable computed using the conductance measurement from which we can estimate $D(\boldsymbol{x}, \boldsymbol{y})$. Note that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ will be redefined per section for the appropriate context. We can now view the problem of estimating Hamming distance from a conductance measurement as a classic communication problem. The transmitter sends $D(\boldsymbol{x}, \boldsymbol{y})$ and the channel is Gaussian with zero mean and variance $\sigma^2(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, as specified in (2.7). The receiver sees $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and estimates $D(\boldsymbol{x}, \boldsymbol{y})$ from the observation.

We note that $D(\boldsymbol{x}, \boldsymbol{y})$ takes only integer values from 0 to n. We also observe that for any two nearby distributions which center at D_H and $D_H + 1$, their variances only differ by a small amount when σ_H and σ_L are relatively close. Based on these observations, we propose an estimator $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$. We refer to this estimator as the *nearest integer estimator*. The nearest integer estimator completes the last step of HD-EIM, which estimates $D(\boldsymbol{x}, \boldsymbol{y})$ from a single measurement $G(\boldsymbol{x}, \boldsymbol{y})$.

From (2.7), we observe that $\sigma^2(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$ does not have a clear relationship to the message $D(\boldsymbol{x}, \boldsymbol{y})$ which makes the analysis of the estimation error intractable. The approximation in (2.3), which leads to (2.7), is suitable for a variety of "On/Off" resistance ratios. However, smaller "On/Off" resistance ratios are usually reported in literature, e.g., $\epsilon = 0.01$ [CL11]. Therefore, we seek to further simplify our equations in order provide a single parameter characterization for the estimation error. When ϵ is small, the simplified approximations readily follow:

$$\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \quad \frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8),$$
$$\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}(\mu_L, \sigma_L^2). \quad (2.8)$$

With the simplified approximations, the distribution of $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ and the calculation of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ change accordingly:

$$\tilde{G}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\bigg(N_{11} + (N_{01} + N_{10})2\epsilon + N_{00}\epsilon, \frac{N_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N_{01} + N_{10})8\sigma_L^2}{2\mu_H^2} + \frac{N_{00}\sigma_L^2}{2\mu_H^2} \bigg). \quad (2.9)$$

We then compute $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ as:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1 - 3\epsilon}[(1 - \epsilon)(w_x + w_y) + 2n\epsilon - 2\tilde{G}(\boldsymbol{x}, \boldsymbol{y})], \quad (2.10)$$

where

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\bigg(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2N_{11}\sigma_H^2 + 2N_{00}\sigma_L^2 + (N_{01} + N_{10})16\sigma_L^2}{\mu_H^2(1 - 3\epsilon)^2} \bigg). \quad (2.11)$$

The estimation of $D(\boldsymbol{x}, \boldsymbol{y})$ is performed using the nearest integer estimator. This simplification is used in the remaining parts of this section, Section 2.4, and Section 2.5.

2.3.2 Inversion Coding

In the previous discussion, we proposed the HD-EIM scheme which estimates the Hamming distance between two vectors from a single conductance measurement, with the assumption that the weights of both vectors are *a priori* known. However, this assumption may not be valid for many applications. Therefore, in order to generalize our HD-EIM scheme to applications where vectors with arbitrary weights are of interest, we require a method to gain knowledge of vector weights. In this subsection we discuss two approaches to get vector weights before HD-EIM is performed: weight estimation and inversion coding. We briefly describe the idea of weight estimation and elaborate on the inversion coding technique.

In [CC15], where the ideal two-state conductance model of a memristor is considered, the weight of a vector can be computed from the measurement between memristor that store the vector itself and some preset vector, e.g., the all-1 vector. We can use this idea, in conjunction with the techniques described in 1.3.1, to estimate the weight of a vector in the presence of variability due to resistance variation. After the weights of two vectors are estimated, the Hamming distance can be then estimated using the HD-EIM scheme. This approach

of estimating the weight of both vectors first, and then estimating the Hamming distance requires a total of three conductance measurements in order to complete one Hamming distance estimation. The extra two measurements introduce extra latency which is not favored by frequent read applications. The overall estimation accuracy then also suffers from additional estimation errors when estimating the weights of vectors.

Alternatively, in applications where latency is the primary concern, we use the following coding technique to force every vector to have the same Hamming weight prior to the data being written to resistive memory, thus enabling Hamming distance Estimation-In-Memory with only one conductance measurement.

Auxiliary Code 1. *(cf. [CC15]). We define an inversion encoding of the vector* $\boldsymbol{x}$ *to be* $\boldsymbol{x}^{(c)} = [\boldsymbol{x}|\neg\boldsymbol{x}]$*, where* $\neg\boldsymbol{x}$ *is the bitwise complement of* $\boldsymbol{x}$ *and* $|$ *denotes concatenation.*

We refer to Auxiliary Code 1 as inversion coding throughout this chapter. Let us define the weight of a vector $\boldsymbol{x}$ as $w(\boldsymbol{x})$. With inversion coding, we have $w(\boldsymbol{x}^{(c)}) = n, \forall \boldsymbol{x} \in \{0,1\}^n$. With inversion coded vectors stored in resistive memory, the weights are known to be n, thus HD-EIM can be readily used. By the nature of inversion coding, we have $D(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = 2D(\boldsymbol{x}, \boldsymbol{y})$. This relationship can thus be used to estimate $D(\boldsymbol{x}, \boldsymbol{y})$ from the $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ using the following equation of the redefined $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}), \tag{2.12}$$

where

$$\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = \frac{1}{1-3\epsilon}[2n(1-\epsilon) + 4n\epsilon - 2\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})]. \tag{2.13}$$

The random variables $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ denote the normalized conductance measurements between two rows that store the coded vectors $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$; $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ will be redefined per section for the appropriate context. Adapting (2.9), we have the distribution of $\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ as follows:

$$\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) \sim \mathcal{N}\left(N'_{11} + (N'_{01} + N'_{10})2\epsilon + N'_{00}\epsilon, \frac{N'_{11}\sigma_H^2}{2\mu_H^2} + \frac{(N'_{01} + N'_{10})8\sigma_L^2}{2\mu_H^2} + \frac{N'_{00}\sigma_L^2}{2\mu_H^2}\right). \tag{2.14}$$

Here N'_{gh} is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$, for $g, h \in \{0, 1\}$. We therefore have:

$$\tilde{D}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) \sim \mathcal{N}\left(D(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}), \frac{2N'_{11}\sigma_H^2 + 2N'_{00}\sigma_L^2 + (N'_{01} + N'_{10})16\sigma_L^2}{\mu_H^2(1-3\epsilon)^2} \right), \tag{2.15}$$

and

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(D(\boldsymbol{x}, \boldsymbol{y}), \frac{2N'_{11}\sigma_H^2 + 2N'_{00}\sigma_L^2 + (N'_{01} + N'_{10})16\sigma_L^2}{4\mu_H^2(1-3\epsilon)^2} \right). \tag{2.16}$$

We can thus estimate $D(\boldsymbol{x}, \boldsymbol{y})$ from $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ using the nearest integer estimator $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, thus adapting the HD-CIM scheme to the case where an inversion coding is used. We now show the optimality of inversion coding in terms of its redundancy among all constant weight codes.

Lemma 1. *Define a code to be a injective mapping that maps a message $\boldsymbol{x} \in \{0, 1\}^a$ to a codeword $\bar{\boldsymbol{x}} \in \{0, 1\}^b$. Let $D(\boldsymbol{x}, \boldsymbol{y})$ denote the Hamming distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. We define a code to be Hamming distance preserving if and only if $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) = f\left(D(\boldsymbol{x}, \boldsymbol{y})\right)$ for some bijective function f. We also define a constant weight code to be a code that satisfies the property $w(\bar{\boldsymbol{x}}) = w_0, \forall \bar{\boldsymbol{x}} \in \{0, 1\}^b$ and some constant w_0. The necessary conditions for a constant weight code to be Hamming distance preserving are:*

$$w_0 \geq a, \ b \geq 2a.$$

Proof. Define $\mathcal{D}_1$ to be the range of $D(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1\}^a$. We have $|\mathcal{D}_1| = a + 1$. Define $\mathcal{C}_1$ to be the set of length-b binary vectors that have weight w_0. Also define $\mathcal{C}_2$ to be the set of length-b codewords generated by any constant weight code with weight w_0. Due to the nature of codes, $\mathcal{C}_2 \subseteq \mathcal{C}_1$. Define $\mathcal{D}_2$ to be the range of $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}} \in \mathcal{C}_1$, define $\mathcal{D}_3$ to be the range of $D(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}} \in \mathcal{C}_2$. The following relationship can be easily verified, $|\mathcal{D}_2| = \min(w_0 + 1, b - w_0 + 1)$. In order for the code to be Hamming distance preserving, i.e., f to be a bijective function, we need $|\mathcal{D}_3| = |\mathcal{D}_1| = a + 1$. Since $\mathcal{C}_2 \subseteq \mathcal{C}_1$, we have $|\mathcal{D}_3| = a + 1 \leq |\mathcal{D}_2| = \min(w_0 + 1, b - w_0 + 1)$, which proves our necessary conditions. ■

Due to the limited read accessibility to resistive memory, there is no direct access to $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$. For any constant weight code we could use, we can only estimate the Hamming distance between between pairs of codewords using HD-EIM scheme. It is therefore necessary for our constant weight code to be Hamming distance preserving so that we can directly recover the Hamming distance between the corresponding pairs of messages. Since the rate of this code is a/b, Lemma 1 implies that the maximum rate of a code of this type is $1/2$. The inversion coding indeed has rate $1/2$ and is thus optimal in term of redundancy.

2.3.3 Estimation Error Probability and Bounds

In previous subsections, we have introduced two methods for HD-EIM: one in which vectors with known weights are stored and the other in which inversion coded vectors are stored. Due to resistance variation, estimation errors can occur when determining the Hamming distance between vectors. In this subsection, the estimation error probability is studied for the two cases and the results are compared.

Lemma 2. *When two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ with known weight are stored in resistive memory, relying on approximation (2.8), we can estimate $D(\boldsymbol{x}, \boldsymbol{y})$ using $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$, with $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ computed from (2.10), and the normalized conductance measurement $\tilde{G}(\boldsymbol{x}, \boldsymbol{y})$ from (2.9). Then, the conditional estimation error probability has the following bound:*

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y})) \leq 2Q\left(\frac{1}{2\sqrt{\beta(n + 7D(\boldsymbol{x}, \boldsymbol{y}))}}\right), \tag{2.17}$$

where $\beta = \frac{2\max(\sigma_L^2, \sigma_H^2)}{\mu_H^2(1-3\epsilon)^2}$ and $Q(\cdot)$ is the Q-function, i.e., $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{u^2}{2}) du$.

Proof. From $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))$ and (2.11), the probability of erroneous estimation can be calculated as:

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y})) = 2Q\left(\frac{1}{2\sigma(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}))}\right),$$

Define $\sigma_{max}^2 = \max(\sigma_H^2, \sigma_L^2)$. The standard deviation of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ can be upper bounded:

$$\sigma(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) = \sqrt{\frac{2N_{11}\sigma_H^2 + 2N_{00}\sigma_L^2 + (N_{01} + N_{10})16\sigma_L^2}{\mu_H^2(1-3\epsilon)^2}}$$
$$\leq \sqrt{\frac{2N_{11}\sigma_{max}^2 + 2N_{00}\sigma_{max}^2 + (N_{01} + N_{10})16\sigma_{max}^2}{\mu_H^2(1-3\epsilon)^2}} = \sqrt{\beta(n + 7D(\boldsymbol{x}, \boldsymbol{y}))}.$$

Using elementary properties of the Q-function, Lemma 2 is proved. ■

Lemma 2 provides us with a single parameter relation between β and the estimation error probability. The parameter β serves as a measure of device reliability and examples of β are provided in Appendix A Table A.1 It is also observed that the estimation error probability is largely dependent on the Hamming distance. A smaller Hamming distance is associated with a smaller estimation error probability. This property is well suited for classification problems in which objects with smaller distances are of interest.

We also provide a bound of the unconditional estimation error probability of HD-EIM when vectors with known weights are stored.

Corollary 2.1. *The unconditional estimation error probability of HD-EIM, when vectors with known weights are stored, is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})) \leq 2Q\left(\frac{1}{4\sqrt{2\beta n}}\right).$$

Proof. Follows immediately from $D(\boldsymbol{x}, \boldsymbol{y}) \leq n, \forall\, \boldsymbol{x}, \boldsymbol{y}$ and Lemma 2. ■

This simple bound gives insight into the trade-off between β, the device reliability and n, the vector length (scalability). For instance, we can achieve the same level of Hamming distance calculation accuracy with larger resistance variation by shortening the vectors.

We next adapt Lemma 2 and Corollary 2.1 to the case where inversion coding is used.

Lemma 3. *When two inversion coded vectors, $\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)} \in \{0,1\}^{2n}$, corresponding to two original vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, are stored in resistive memory, relying on approximation*

(2.8), we can estimate $D(\boldsymbol{x},\boldsymbol{y})$ *using* $\hat{D}(\boldsymbol{x},\boldsymbol{y}) = nint(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))$*, with* $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$ *computed from (2.12), and the normalized conductance measurement* $\tilde{G}(\boldsymbol{x},\boldsymbol{y})$ *from (2.14). Then, the conditional estimation error probability is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x},\boldsymbol{y}) \neq D(\boldsymbol{x},\boldsymbol{y}) \mid D(\boldsymbol{x},\boldsymbol{y})) \leq 2Q\left(\frac{1}{\sqrt{2\beta(n+7D(\boldsymbol{x},\boldsymbol{y}))}}\right), \tag{2.18}$$

where $\beta = \frac{2\max(\sigma_L^2,\sigma_H^2)}{\mu_H^2(1-3\epsilon)^2}$.

Proof. The proof of this lemma is similar to proof of Lemma 2 with $\sigma^2(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))$ following from (2.16) in conjunction with the following properties of inversion coding:

$$N'_{11} = N'_{00} = N_{11} + N_{00}, \;\; N'_{01} = N'_{10} = N_{01} + N_{10}.$$

■

We also provide a simple bound of the estimation error probability of HD-EIM when inversion coded vectors are stored.

Corollary 3.1. *The unconditional estimation error probability of HD-EIM, when inversion coded vectors are stored, is upper bounded as:*

$$P(\hat{D}(\boldsymbol{x},\boldsymbol{y}) \neq D(\boldsymbol{x},\boldsymbol{y})) \leq 2Q\left(\frac{1}{4\sqrt{\beta n}}\right)$$

Proof. Follows immediately from $D(\boldsymbol{x},\boldsymbol{y}) \leq n, \forall\, \boldsymbol{x},\boldsymbol{y}$ and Lemma 3. ■

The bounds in Lemmas 2 and 3 are tight when $\sigma_H^2 = \sigma_L^2$. We observe that when these bounds are tight, the estimation error probability using coded vectors is smaller than the one using uncoded vectors. This observation shows that inversion coding also improves the estimation accuracy, in addition to its ability to generalize HD-EIM to vectors with unknown weights. Note that additional coding, e.g., encoding $\boldsymbol{x}$ to be $[\boldsymbol{x}|\neg\boldsymbol{x}|\boldsymbol{x}|\neg\boldsymbol{x}]$, further improves estimation accuracy but may not be favorable in terms of redundancy. The single inversion coding technique is the maximum rate constant weight code that preserves Hamming

distance, thus allowing HD-EIM to be efficiently performed with a single conductance measurement. However, additional coding only improves estimation accuracy with diminishing returns.

2.3.4 An Average-Case Study on the k-NN Classifier Under Computation Noise Using HD-EIM

In the previous discussion we analyzed the estimation error of the Hamming distance Estimation-in-Memory scheme under device variability due to resistance variation. Due to resistance variation, the Hamming distance between vectors has to be estimated and could lead to estimation error. From the application point of view, the estimation error can be viewed as computation noise when Hamming distance computation is performed. We have provided a bound on the conditional error probability. Meanwhile, erroneous Hamming distance computations may not necessarily lead to erroneous results due to the error-tolerant nature of some applications, e.g., the k-nearest neighbors classifier. In this subsection, we focus on the k-nearest neighbors classifier as our main application and provide analysis on how this computation noise affects the classification accuracy. Throughout this whole subsection, we assume binary attributes (vectors) are inversion coded and stored in resistive memory. We next introduce the framework we use to analyze the k-nearest neighbors classifier under computation noise.

The average-case analysis framework, introduced by Pazzani and Sarrett [PS92], is a useful theoretical framework to understand the behavior of learning algorithms. This framework is based on the formal computation of the expected accuracy of a learning algorithm for a certain fixed class of concepts [OY97]. We use this framework to formally compute the expected accuracy of the k-NN classifier to learn the m-of-n/ℓ concept, a boolean threshold function, under computation noise. The m-of-n/ℓ concept is defined by the number of relevant attributes (n), the number of irrelevant attributes (ℓ), and the threshold (m). Under this concept, an instance is positive if m or more relevant attributes exist and is negative if

fewer than m relevant attributes exist. Note that n is redefined in this subsection in order to be consistent with [OY97]. We assume that relevant and irrelevant attributes occur with probabilities p and q, respectively. The k-NN classifier classifies an instance as positive if more than half of its k nearest neighbors are positive. When a tie occurs, the classifier randomly decides the class. We compute the expected accuracy when the bound is tight, i.e., $\sigma_H^2 = \sigma_L^2$ and the computation noise is parameterized by β, as stated in Lemma 3. The expected classification accuracy is a function of $n, m, p, q, k, N,$ and β, where N is the size of the training space. We use the next example to illustrate the idea of the k-nearest neighbors classifier and how an erroneous Hamming distance computation may not lead to erroneous classification.

Example 1. *In this example, we are interested in the classification of the testing instance based on the 5 training instances using a k-nearest neighbors classifier. The 1 testing instance and 5 training instances are listed in the next Table. We set k to be 3 for the k-NN classifier. The training instances are generated according to the 4-of-8/0 concept and we use 1/0 to mark positive/negative class labels.*

Table 2.1: An example on k-NN.

Testing Instance					
Attribute	11111000				
Training Instances					
Instance #	1	2	3	4	5
Class Label	1	0	1	0	0
Attribute	11111100	11100000	11110100	11000000	11000100
Hamming Distance to Testing Case	1	2	2	3	4

In the error-free case, i.e., all Hamming distances are computed correctly, the k-NN classifier classifies the testing instance as positive because its 3 nearest neighbors (instance #1, #2, and #3) have class labels 1, 0, and 1 respectively. Positive is the correct class label

for the testing instance based on the 4-of-8/0 concept.

Now we consider the case where Hamming distance computation is prone to estimation error due to resistance variation. We take a special case where the Hamming distance between the testing instance (11111000) and training instance #2 (11100000) is erroneously computed to be 3. Note that there are two training instances (#2 and #4) at distance 3 w.r.t. the testing instance; the k-NN classifier will randomly choose one to be the nearest neighbor of the testing instance. Observe that although the Hamming distance computation is erroneous, the classification result is still positive because the 2 closest training instance (#1 and #3) are both positive. This example illustrate how erroneous Hamming distance computation may not lead to erroneous classification.

2.3.4.1 Formal Calculation of Expected Accuracy

In [OY97], Okamoto and Nobuhiro used the average-case study framework to analyze noisy attributes and class labels. We adapt some of their equations to the scenario involving computation noise. For the calculations of expected accuracy that have already been done by Okamoto and Nobuhiro, we simply state their results here (we refer readers to [OY97] for further details). We modify the equations to incorporate the computation noise and to reflect the fact that in our model, the attributes and class labels are noise-free.

The probability that an instance consists of x relevant attributes and y irrelevant attributes can be calculated as [OY97]:

$$P_{oc}(x,y) = \binom{n}{x}\binom{l}{y}p^x(1-p)^{n-x}q^y(1-q)^{l-y}.$$

Then the expected classification accuracy can be calculated as [OY97]:

$$A(k) = \sum_{y=0}^{l}\left[\sum_{x=0}^{m-1} P_{oc}(x,y)(1-P_{pos}(k,x,y)) + \sum_{x=m}^{n} P_{oc}(x,y)P_{pos}(k,x,y)\right],$$

where $P_{pos}(k,x,y)$ represents the probability that the k-NN classifier classifies an arbitrary test instance with x relevant attributes and y irrelevant attributes as positive.

To calculate $P_{pos}(k,x,y)$, we first calculate $P_{dp}(x,y,e)$ ($P_{dn}(x,y,e)$, resp.) which is the probability that an arbitrary positive (negative, resp.) training instance has Hamming distance e from the arbitrary testing instance $t(x,y) \in I(x,y)$. The Hamming distance e is assumed to be estimated by HD-EIM scheme from an observation $\tilde{e}$. $I(x,y)$ represents the set of instances in which x relevant attributes and y irrelevant attributes simultaneously occur. $P_{dp}(x,y,e)$ and $P_{dn}(x,y,e)$ can be represented as:

$$P_{dp}(x,y,e) = \sum_{\hat{e}=0}^{n} \sum_{\hat{y}=0}^{l} \sum_{\hat{x}=m}^{n} P_{oc}(\hat{x},\hat{y}) P_{dis}(x,y,\hat{x},\hat{y},\hat{e}) P_H(e,\hat{e}), \tag{2.19}$$

and

$$P_{dn}(x,y,e) = \sum_{\hat{e}=0}^{n} \sum_{\hat{y}=0}^{l} \sum_{\hat{x}=0}^{m-1} P_{oc}(\hat{x},\hat{y}) P_{dis}(x,y,\hat{x},\hat{y},\hat{e}) P_H(e,\hat{e}). \tag{2.20}$$

$P_{dis}(x,y,\hat{x},\hat{y},e)$ is the probability that an arbitrary instance in $I(\hat{x},\hat{y})$ has Hamming distance e from an arbitrary instance in $I(x,y)$. $P_H(e,\hat{e})$ is the probability that the original Hamming distance $\hat{e}$ is estimated to be e.

In order to compute $P_H(e,\hat{e})$, we consider the following scenario where the observation $\tilde{e}$ is a random variable with distribution $\mathcal{N}(\hat{e}, \frac{1}{2}\beta(n+7\hat{e}))$. This corresponds to the case that two attribute vectors with Hamming distance $\hat{e}$ are inversion coded and then stored in resistive memory. For $e = \text{nint}(\tilde{e})$ if $0 \le \tilde{e} \le n$, $e = 0$ if $\tilde{e} < 0$, and $e = n$ if $\tilde{e} > n$, using elementary properties of Gaussian distribution, $P_H(e,\hat{e})$ is calculated as follows:

$$P_H(e,\hat{e}) = \begin{cases} 1 - Q\left(\frac{\frac{1}{2}-\hat{e}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{if} \quad e = 0, \\ Q\left(\frac{n-\frac{1}{2}-\hat{e}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{if} \quad e = n, \\ Q\left(\frac{e-\hat{e}-\frac{1}{2}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) - Q\left(\frac{e-\hat{e}+\frac{1}{2}}{\sqrt{\frac{1}{2}\beta(n+7\hat{e})}}\right) & \text{otherwise.} \end{cases}$$

The remaining equations in this subsection are stated directly from [OY97]; we include them here for completeness. $P_{dis}(x,y,\hat{x},\hat{y},e)$ can be calculated as follows [OY97]:

$$P_{dis}(x,y,\hat{x},\hat{y},e) = \sum_{(z_r,z_i)\in S} \frac{\binom{x}{z_r}\binom{n-x}{\hat{x}-z_r}}{\binom{n}{\hat{x}}} \frac{\binom{y}{z_i}\binom{l-y}{\hat{y}-z_i}}{\binom{l}{\hat{y}}},$$

where S is a set of all pairs of z_r and z_i that satisfy the following conditions:

$$\max(0, x+\hat{x}-n) \leq z_r \leq \min(x, \hat{x}), \max(0, y+\hat{y}-l) \leq z_i \leq \min(y, \hat{y}), z_r+z_i = \frac{x+y+\hat{x}+\hat{y}-e}{2}.$$

$P_{pos}(k, x, y)$ can be then calculated.

$$P_{pos}(k, x, y) = \sum_{d=0}^{n+l} \sum_{a=0}^{k-1} \sum_{b=k-a}^{N-a} P_{num}(x, y, d, a, b) P_{sp}(x, y, d, a, b).$$

where

$$P_{num}(x, y, d, a, b) = \binom{N}{a} \binom{N-a}{b} P_l(x, y, d)^a \times P_d(x, y, d)^b (1 - P_l(x, y, d) - P_d(x, y, d)^{N-a-b},$$

$$P_l(x, y, d) = \sum_{e=0}^{d-1} (P_{dp}(x, y, e) + P_{dn}(x, y, e)),$$

$$P_d(x, y, d) = P_{dp}(x, y, d) + P_{dn}(x, y, d),$$

and

$$P_{sp}(k, x, y, d, a, b)$$
$$= \sum_{u=0}^{a} \sum_{v=0}^{b} \left\{ P_{lp}^{(u)}(a, u) P_{dp}^{(v)}(b, v) \sum_{w=\lceil \frac{k+1}{2} \rceil - u}^{v} \left\{ P_{dp}^{(w)}(k, a, b, v, w) + \frac{1}{2} P_{dp}^{(w)}(k, a, b, v, \frac{k}{2} - u) \right\} \right\},$$

where

$$P_{lp}^{(u)}(a, u) = \binom{a}{u} (\frac{\sum_{e=0}^{d-1} P_{dp}(x, y, e)}{P_l(x, y, d)})^u (\frac{\sum_{e=0}^{d-1} P_{dn}(x, y, e)}{P_l(x, y, d)})^{a-u},$$

$$P_{dp}^{(v)}(b, v) = \binom{b}{v} (\frac{P_{dp}(x, y, d)}{P_d(x, y, d)})^v (\frac{P_{dn}(x, y, d)}{P_d(x, y, d)})^{b-v},$$

$$P_{dp}^{(w)}(k, a, b, v, w) = \frac{\binom{v}{w} \binom{b-v}{k-a-w}}{\binom{b}{k-a}}.$$

This ends our derivation for the expected accuracy.

2.3.4.2 Average-Case Study Results Using HD-EIM

First we study the effects of different levels of computation noise on the 3-of-5/2 concept. In Figure 2.2, we fix N to be 32 and we report two device reliability (noise-level), $\beta = 0.01$ and $\beta = 0.1$, for k spanning from 1 to 16.

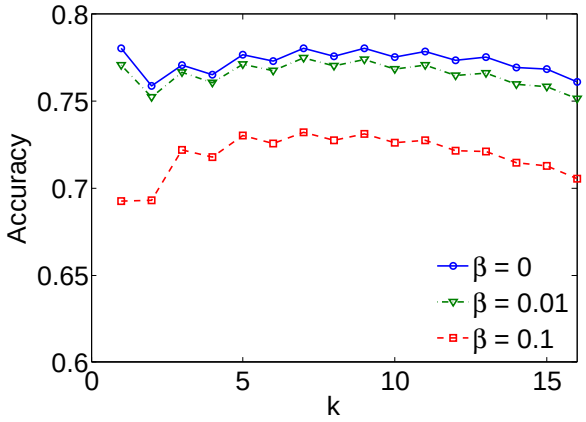


Figure 2.2: The performance of 3-of-5/2 concept under computation noise.

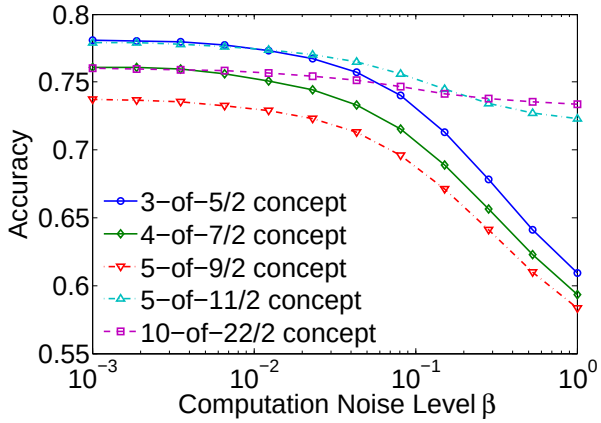


Figure 2.3: Learning accuracy with various levels of computation noise.

In Figure 2.2, the upper line is the theoretical result for the noise-free classification accuracy, which is consistent with the results in [OY97]. The two lower lines are the theoretical accuracy with different device parameter β. When k is an even number, it is observed that the classification accuracy of the k-NN classifier drops remarkably due to the randomness when a tie occurs. Figure 2.2 shows that the computation noise decreases the classification accuracy for all k and the amount of decrement largely depends on the noise level β.

We next show the effects of computation noise for a wide range of noise levels on a variety of different concepts in Figure 2.3. The classification accuracy of k-NN at each noise level

for each concept is chosen to be the maximum accuracy out of the range $2 \leq k \leq 16$.

In Figure 2.3, β ranges from 10^{-3} to 1 in order to see a clear trend. However, from a realistic point of view, $\beta = \frac{2\max(\sigma_L^2,\sigma_H^2)}{\mu_H^2(1-3\epsilon)^2} = 1$ is unlikely to occur in a real device. We observe that for small β, i.e., $\beta < 10^{-2}$, the influence of the estimation error is negligible. It is interesting to note that the impact of the estimation error decreases as the dimension of concepts increases, which could be beneficial for applications with many attributes, e.g., $n = 22$.

2.4 Error-Detection and Error-Correction Scheme Under Write BSC

In this section we consider memristor variability due to unreliable write operations. The adverse effect of unreliable write operation can be modeled as a write BSC and therefore can be viewed as attribute noise in learning problems. In [OY97], the authors provide an average-case study and observed that attribute noise decreases classification accuracy. In order to keep the low latency property of HD-CIM, we are restricted to the conductance measurements between rows of memristors. Therefore, traditional ECCs based on entry-wise access can not be used to deal with the write BSC and this fact motivates us to provide error-detection and error-correction schemes based on conductance measurements only. In this chapter, error-detection schemes are explored in 2.4.1 and 2.4.2. Two different error-correction schemes are explored in 2.4.3 and 2.4.4. In 2.4.5, we present the average-case study on k-nearest neighbor classifier using one of our proposed error-correction scheme in 2.4.4. Throughout this section, we assume the memristors have no resistance variation, i.e., $\sigma_H^2 = \sigma_L^2 = 0$. We again assume inversion coded vectors are stored in the resistive memory.

2.4.1 Single Error Detection

Define $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ to be two inversion coded vectors corresponding to two length-n vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Let $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ be the noisy vectors actually stored due to the write noise (BSC). We first establish necessary equations to compute the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ from a measurement between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$. Define the normalized conductance measurement to be $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$. As resistance variability is not considered in this section, $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ is a constant rather than a random variable. We thus have:

$$\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \tilde{N}'_{11} + (\tilde{N}'_{10} + \tilde{N}'_{01})\frac{2\epsilon}{1+\epsilon} + \tilde{N}'_{00}\epsilon. \tag{2.21}$$

Here $\tilde{N}'_{gh}$ is defined to be the number of coordinates having bit g in $\tilde{\boldsymbol{x}}^{(c)}$ and bit h in $\tilde{\boldsymbol{y}}^{(c)}$, for $g, h \in \{0, 1\}$.

A variable denoting the normalized conductance measurement between rows of memristors that store $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ can be defined as:

$$\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}) = N'_{11} + (N'_{10} + N'_{01})\frac{2\epsilon}{1+\epsilon} + N'_{00}\epsilon. \tag{2.22}$$

Here N'_{gh} is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$, for $g, h \in \{0, 1\}$.

We again use an intermediate variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ to denote the result calculated from the measurement as follows:

$$\tilde{D}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \frac{1+\epsilon}{(1-\epsilon)^2}[2n(1-\epsilon) + 4n\epsilon - 2\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \tag{2.23}$$

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\tilde{D}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \frac{1+\epsilon}{(1-\epsilon)^2}[n + n\epsilon - \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \tag{2.24}$$

Note that when no error occurs, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y})$. Equation (2.23) is adapted from Equation (2.6) when inversion coding is used and write BSC is considered.

It is useful to denote the difference in conductance measurements between the noisy and the noise-free case as:

$$\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) - \tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}).$$

Similarly, let us define $\Delta D(\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, which is later used to characterize bit errors, as follows:

$$\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) - D(\boldsymbol{x}, \boldsymbol{y}) = -\frac{(1+\epsilon)\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})}{(1-\epsilon)^2}. \tag{2.25}$$

We now calculate how $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ are affected by a single bit error, i.e., $t = 1$. Due to the symmetry of this problem, there are only four different fundamental types of errors for a pair of elements $x_i^{(c)}$ and $y_i^{(c)}$, $(0,0) \to (0,1)$, $(0,1) \to (0,0)$, $(0,1) \to (1,1)$, and $(1,1) \to (0,1)$, which we denote as error types A, B, C, and D, respectively. All other error types are expressed in terms of these four error types. Each of these error types affects the relationship between $\{N'_{00}, N'_{01}, N'_{10}, N'_{11}\}$ and $\{\tilde{N}'_{00}, \tilde{N}'_{01}, \tilde{N}'_{10}, \tilde{N}'_{11}\}$. For example, an error that changes $(0,0)$ into $(0,1)$ will result in $\tilde{N}'_{01} = N'_{01} + 1$, $\tilde{N}'_{00} = N'_{00} - 1$, $\tilde{N}'_{11} = N'_{11}$ and $\tilde{N}'_{10} = N'_{10}$. Table 2.2 lists the resulting values of $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ for each error type. The above two variables are abbreviated as $\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ and $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ since the relative relationship is clear.

We use the following lemma to detect a single bit error.

Lemma 4. *If exactly one of $\tilde{\boldsymbol{x}}^{(c)}$ or $\tilde{\boldsymbol{y}}^{(c)}$ contains a single bit error, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = 1$ and $0 < \epsilon < \frac{1}{2}$, then we can detect that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$.*

Proof. If no error is present, we have $D(\boldsymbol{x}, \boldsymbol{y}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y})$, i.e., $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = 0$. For $0 < \epsilon < \frac{1}{2}$, each error type will cause the value of $\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ to be a non-integer and in turn lead to non-integer $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ (since $D(\boldsymbol{x}, \boldsymbol{y})$ is always an integer and $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y}) + \Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$). A single bit error can thus be detected by first computing $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and checking whether it is an integer or not. ■

Table 2.2: The four types of errors.

Error Type	$(x_i^{(c)}, y_i^{(c)}) \to (\tilde{x}_i^{(c)}, \tilde{y}_i^{(c)})$	$\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$	$\Delta\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$
A	(0,0) $\to$ (0,1)	$\frac{2\epsilon}{1+\epsilon} - \epsilon$	$\frac{-\epsilon}{1-\epsilon}$
B	(0,1) $\to$ (0,0)	$-\left(\frac{2\epsilon}{1+\epsilon} - \epsilon\right)$	$\frac{\epsilon}{1-\epsilon}$
C	(0,1) $\to$ (1,1)	$-\left(\frac{2\epsilon}{1+\epsilon} - 1\right)$	$\frac{-\epsilon}{1-\epsilon} - 1$
D	(1,1) $\to$ (0,1)	$\frac{2\epsilon}{1+\epsilon} - 1$	$\frac{\epsilon}{1-\epsilon} + 1$

This process of error detection is later referred to as an *integer check*. When error-free, $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = D(\boldsymbol{x}, \boldsymbol{y})$ where $\hat{D}(\boldsymbol{x}, \boldsymbol{y})$ is the estimated result from the intermediate variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$.

Lemma 4 suggests that a single bit error is detectable under certain reasonable constraints on ϵ. However, the error type can not be uniquely determined from $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. For example, a vector with $D(\boldsymbol{x}, \boldsymbol{y}) = D_H$ incurring error type B would result in an identical value for $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ as a vector with $D(\boldsymbol{x}, \boldsymbol{y}) = D_H - 1$ incurring error type D. Note that this multiple solution result is due to the nature of the underlying problem and is thus unavoidable if only vector-level information is used.

2.4.2 Multiple Errors Detection

So far we have shown that a single bit error in a pair of inversion coded vectors can be detected by checking whether $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is an integer or not. Next, we generalize the idea of an *integer check* to multiple errors.

Lemma 5. *If together $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ contain an odd number of bit errors t, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = t$ (odd), then we conclude that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$ for the following constraint on ϵ:*

$$0 < \epsilon < \frac{1}{t+1}.$$

Proof. Define $\boldsymbol{e}_{i,p}, i \in \{1, ..., t\}, p \in \{\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}\}$ to be the error vectors corresponding to the i-th error in the vector indexed by p. For clarity, the superscript in p is omitted. For example, if the i-th bit flip is at the j-th position of $\boldsymbol{x}^{(c)}$, $\boldsymbol{e}_{i,\boldsymbol{x}}$ is the all-0 vector with 1 at the j-th position and $\boldsymbol{e}_{i,\boldsymbol{y}}$ is an all-0 vector. (We permit error at the j-th position in $\boldsymbol{y}_c$ but there will be another pair of error vectors corresponding to it.) We further define $\Delta\tilde{D}(\boldsymbol{e}_i)$ to be the change in output induced by $\boldsymbol{e}_{i,\boldsymbol{x}}$ and $\boldsymbol{e}_{i,\boldsymbol{y}}$, i.e.,

$$\Delta\tilde{D}(\boldsymbol{e}_i) = -\frac{(1+\epsilon)\Delta\tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)}, \boldsymbol{x}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{x}}, \boldsymbol{y}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{y}})}{(1-\epsilon)^2}.$$

The following relation is observed:

$$\begin{aligned}
&\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \\
&= \sum_{i=1}^{t} \{-\frac{(1+\epsilon)[\tilde{G}(\boldsymbol{x}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{x}}, \boldsymbol{y}^{(c)} + \boldsymbol{e}_{i,\boldsymbol{y}}) - \tilde{G}(\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})]}{(1-\epsilon)^2}\} \\
&= \sum_{i=1}^{t} \Delta D(\boldsymbol{e}_i).
\end{aligned}$$

Each $\Delta D(\boldsymbol{e}_i)$ can be viewed as the change of the output for a single bit error, thus assuming the same value as the last column in Table 2.2. For $0 < \epsilon < \frac{1}{t+1}$, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ has a non-integer value because any odd t choices from those values of $D(\boldsymbol{e}_i)$ will be summed to $\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$, where $0 < |\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 < 1$. Here and elsewhere, the operation $x \pmod 1$ is defined as the fractional part of x. As a result, any odd number of bit errors can be detected by an *integer check*.

■

We next present the result for an even number of errors, t.

Lemma 6. *We denote the fraction of errors that are detectable for even t as $r_d(t)$. If together $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ contain an even number of bit errors t, i.e., $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = t$ (even), and the channel parameter satisfies $0 < \epsilon < \dfrac{1}{t+1}$, then*

$$r_d(t) = 1 - \frac{\binom{t}{t/2}}{2^t}.$$

Proof. For an even number of errors, some error patterns are undetectable. We define $t_{A,C}$ to be the number of errors of either type A or C and similarly for $t_{B,D}$. Notice that error types A and C (and error types B and D) have the same integer parts in $\Delta D(\boldsymbol{e}_i)$, thus contributing equally to $|\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1$. The analysis on $t_{A,C}$ and $t_{B,D}$ thus covers all the possible error combinations. The error pattern is undetectable when $t_{A,C} = t_{B,D} = t/2$. By viewing this error detection problem as a fair-coin flipping problem in which $t/2$ heads and $t/2$ tails occur in t trials, we calculate the probability of an undetectable pattern occurring to be $\binom{t}{t/2}/2^t$. When $t_{A,C} \neq t_{B,D}$, let $t' = |t_{A,C} - t_{B,D}|$. We have $|\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 = \frac{\epsilon t'}{1-\epsilon}, 0 < t' \leq t$. With $0 < \epsilon < \dfrac{1}{t+1}$, we have $0 < |\Delta D(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})| \pmod 1 < 1$ which directly allows for error detection. ■

Lemma 5 and Lemma 6 show that multiple errors are sometimes detectable by an *integer check*. Note that by an *integer check* alone, we are unable to differentiate the detected multiple errors and the detected single error.

2.4.3 Exact Error Correction with Additional Coding

From previous discussion, we have error detection schemes to detect erroneous computation of $D(\boldsymbol{x}, \boldsymbol{y})$ by an *integer check* of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. Our ultimate goal is to recover the correct Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. In this subsection, we provide an error correction scheme based on acquiring the full error profile, i.e., error location and error value, in vectors $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ and we call this scheme the Exact Hamming scheme. Error localization is done by comparing with preset vectors while utilizing property of inversion coding. $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$

can thus be successfully recovered relying on extra error correction code.

2.4.3.1 Error Localization

Claim 1. *Define L to be the set of vectors $\boldsymbol{l}_i \in \{0,1\}^{2n}, 1 \leq i \leq n$ whose all bits are one except the the i-th and the $(i+n)$-th bits. Also define $\mathbf{1}$ to be the all-1s vector with length $2n$. Assume error is detected in a pair of vectors $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$, with $n+1$ pairwise measurements between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and vectors in $L \cup \mathbf{1}$, we can narrow the location of each error (if any) to two positions, i and $i+n$ except rare cases. Same applies to $\tilde{\boldsymbol{y}}^{(c)}$.*

Without loss of generality, we assume there are bit errors in $\tilde{\boldsymbol{x}}^{(c)}$. With the measurement of $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1})$, we compute $\tilde{D}(\tilde{\boldsymbol{x}}, \mathbf{1})$ using Equation 2.24. We then perform n measurements between rows of memristor that store $\tilde{\boldsymbol{x}}^{(c)}$ and each of the vectors in L to compute $\tilde{D}(\tilde{\boldsymbol{x}}, \boldsymbol{l}_i)$ using the same equation. Error localization is achieved by computing $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \tilde{D}(\tilde{\boldsymbol{x}}, \boldsymbol{l}_i) - \tilde{D}(\tilde{\boldsymbol{x}}, \mathbf{1})$ for all $1 \leq i \leq n$. Let $I_{error} = \{i \in I | \Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 2\} \cup \{i \in I | \Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon}\}$. There is an error either at position i or at position $i+n$ for each $i \in I_{error}$.

By Auxiliary Code 1, for a given $\boldsymbol{l}_i$, if both the i-th and the $(i+n)$-th positions of $\tilde{\boldsymbol{x}}^{(c)}$ are error-free, the error patterns are (1,1) $\rightarrow$ (0,1) and (0,1) $\rightarrow$ (0,0) which result in $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 1$. If a bit error occurs at either the i-th or the $(i+n)$-th position of $\hat{\boldsymbol{x}}_c$, then the corresponding error patterns are two (1,1) $\rightarrow$ (0,0) which lead to $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon} + 2$ for bit flip from 1 to 0 or two (0,1) $\rightarrow$ (0,0) with $\Delta D_i(\tilde{\boldsymbol{x}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{x}}^{(c)}, \boldsymbol{l}_i) = \frac{2\epsilon}{1-\epsilon}$ for bit flip from 0 to 1. If errors occur at both the i-th or the $(i+n)$-th positions of $\hat{\boldsymbol{x}}_c$, we are unable to localize those two errors.

2.4.3.2 Bit Value Reading

Using the same idea of comparing with preset vectors, we are able to read bit-value at arbitrary location using conductance measurements between rows of memristor.

Claim 2. *Define B to be the set of vectors $\boldsymbol{b}_i \in \{0,1\}^{2n}, 1 \leq i \leq 2n$ whose bits are all one except the i-th bit. The i-th bit value of a given vector, say $\tilde{\boldsymbol{y}}^{(c)}$, can be inferred from two pairwise measurements between, $\tilde{\boldsymbol{y}}^{(c)}$ and $\boldsymbol{b}_i$; $\tilde{\boldsymbol{y}}^{(c)}$ and $\mathbf{1}$.*

Two measurements, $\tilde{G}(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1})$ and $\tilde{G}(\tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i)$, are taken and the corresponding $\tilde{D}(\tilde{\boldsymbol{y}}, \mathbf{1})$ and $\tilde{D}(\tilde{\boldsymbol{y}}, \boldsymbol{b}_i)$ are computed using equation 2.24. The inference of the i-th bit value in $\hat{\boldsymbol{y}}_c$ is based on $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \tilde{D}(\tilde{\boldsymbol{y}}, \boldsymbol{b}_i) - \tilde{D}(\tilde{\boldsymbol{y}}, \mathbf{1})$. For $y_i^{(c)} = 1$, the error pattern is (1,1) $\rightarrow$ (1,0) with $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \dfrac{\epsilon}{1-\epsilon} + 1$ and for $y_i^{(c)} = 0$, the error pattern is (0,1) $\rightarrow$ (0,0) with $\Delta D_i(\tilde{\boldsymbol{y}}^{(c)}, \mathbf{1}, \tilde{\boldsymbol{y}}^{(c)}, \boldsymbol{b}_i) = \dfrac{\epsilon}{1-\epsilon}$.

2.4.3.3 Error Correction with Additional Coding

For each error we localized using technique described in 2.4.3.1, we have ambiguity among two locations. One of these locations is in the first half of the erroneous vector and the other is in the second half. Based on the underlying inversion coding, the values at those two positions are complement of each other when error-free. We can thus focus on recovering the value of the first position by viewing it as an erasure. Here we propose adding extra redundancy by encoding the first half of $\boldsymbol{x}^{(c)}$ using a standard erasure correcting code. After error is detected, these parity bits can be accessed using technique described in 2.4.3.2. Because of this additional coding, we are able to correct every erasure that is conveyed by the error localization step. With the bit-value information of the corresponding vector (acquired using technique in 2.4.3.2), we are able to recover the Hamming distance between the uncorrupted vectors. The error correction capability of this Exact Hamming Scheme depends on the erasure correction capability of this exact code.

The caveat here is that for this erasure correcting code, the Hamming distance may not be preserved. As a result, error correction capability requires the ability to measure the conductance between sub-rows of memristor that store two sub-vectors in the resistive memory.

2.4.4 Soft Hamming Scheme for Single Error Correction

The Exact Hamming scheme can correctly find the exact error location and error value thus achieving the goal of recovering the original Hamming distance. However, it suffers from many aspects. First, the error correction capability of Exact Hamming scheme requires extra erasure correction code which adds redundancy and also requires sub-vector reading. Second, the error localization procedure incurs many costly read operations between rows of memristors. Lastly, the error localization procedure may fail if rare error patterns occur and the erasure correction procedure may fail if there is bits error in the corresponding parity bit. These drawbacks of the Exact Hamming schemes make it not favorable in terms of its cost in space and time.

We notice that some of the potential application of HD-CIM, e.g., the k-nearest neighbor classifier, have certain error tolerant capability. This observation motivate us to provide an approximate error correction scheme focused on correcting the most frequent error, the single bit error. This approximate error correction scheme will improve the performance of our application under the write BSC while keep a small error-detection/error-correction latency. This scheme is based on the following corollary.

For single bit error that occurs, although the error type can not be uniquely determined by an *integer check*, with further constrains on ϵ, we are able to determine whether the error is of type {B,D} or {A,C} .

Corollary 6.1. *If* $D(\boldsymbol{x}^{(c)}, \tilde{\boldsymbol{x}}^{(c)}) + D(\boldsymbol{y}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) = 1$ *and* $0 < \epsilon < \frac{1}{3}$*, then we conclude that* $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \neq D(\boldsymbol{x}, \boldsymbol{y})$ *and we determine whether the error is of type* $\{B,E\}$ *or* $\{A,C\}$*.*

Proof. The ϵ range here, $0 < \epsilon < \frac{1}{3}$, falls within the range of ϵ in Lemma 4, thus a single bit error is detectable. Since $| \frac{\epsilon}{1-\epsilon} | < \frac{1}{2}$, if the error belongs to type {B,D}, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) < 1/2$. Similarly, if the error belongs to type {A,C}, we have $\text{nint}(\tilde{D}(\boldsymbol{x}, \boldsymbol{y})) - \tilde{D}(\boldsymbol{x}, \boldsymbol{y}) < 1/2$. This relationship between $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ and its nearest integer uniquely determine whether the error is of type {A,C} or {B,D}. ■

We say $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$ lies on the right-hand-side (or left-hand-side) of its nearest integer if $\tilde{D}(\boldsymbol{x},\boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) < 1/2$ (or $\text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - \tilde{D}(\boldsymbol{x},\boldsymbol{y}) < 1/2$).

As a result of Corollary 6.1, if $0 < \epsilon < \frac{1}{3}$, for the detected single bit error, we have two possible Hamming distances that could be the original one. We summarize the candidate original Hamming distance as follows:

$$\begin{aligned}
&\text{If } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) < 1/2,\\
&\quad \text{then } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) \text{ or } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - 1;\\
&\text{if } \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - \tilde{D}(\boldsymbol{x},\boldsymbol{y}) < 1/2,\\
&\quad \text{then } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) \text{ or } D(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) + 1.
\end{aligned}$$

We now seek to provide a correction scheme to the detected single bit error. We first realize that choosing randomly between the two candidate solutions is functionally the same as always choosing $\hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))$. Therefore, choosing randomly would not be wise because it does not take advantage of the extra information we can get from Corollary 6.1.

We propose the following scheme for error-detection and error-correction.

$$\begin{aligned}
&\text{If} \quad \tilde{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})), \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y}));\\
&\text{if} \quad \tilde{D}(\boldsymbol{x},\boldsymbol{y}) - \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) < 1/2, \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - 1/2;\\
&\text{if} \quad \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) - \tilde{D}(\boldsymbol{x},\boldsymbol{y}) < 1/2, \text{then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = \text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) + 1/2.
\end{aligned} \tag{2.26}$$

The previous scheme is derived by taking the average of the two candidate Hamming distances after an error is classified as either type {A,C} or {B,D}. As this error-detection and correction scheme has the possibility to give non-integer Hamming distance as the result, we refer it as the Soft Hamming scheme. Analysis of this Soft Hamming scheme on the k-nearest neighbors classifier will be provided in the next subsection.

2.4.5 An Average-Case Study on the k-NN Classifier Under Attribute Noise Using the Soft Hamming Scheme

In order to incorporate the Soft Hamming scheme into the average-case study framework, we assume all error patterns, except the undetectable errors, are detectable and can be separated into the two classes—{A,C} or {B,D}—. We also assume ϵ is sufficiently small and the Soft Hamming scheme is used to correct the detected error.

The next example helps to illustrate how the Soft Hamming Scheme can be used to improve classification accuracy under the write BSC.

Example 2. *In this example we use the same setup as Example 1. Consider the case that under the adverse effect of write BSC, the attribute of training instance #3 (11110100) is changed to 01110100. Without the Soft Hamming scheme, the Hamming distance between training instance #3 and the testing instance is 3. Both instance #3 (class: positive) and instance #4 (class: negative) now have distance 3 w.r.t. the testing instance. Choosing randomly among instance #3 and #4 to be the third nearest neighbor of the testing instance, the k-NN classifier have 50% chance to have erroneous classification.*

If we implement the Soft Hamming scheme, the error pattern $(1,1) \to (0,1)$ *can be detected when computing the Hamming distance between the testing instance and training instance #3. The Soft Hamming scheme will therefore set the Hamming distance between them to be 2.5 based on previous discussions. As a result, the k-NN classifier can correctly classify the testing instance as positive.*

2.4.5.1 Formal Calculation of Expected Accuracy

We define an augmented variable $e^{(s)} = 2\hat{D}(\boldsymbol{x}, \boldsymbol{y})$, where $\hat{D}(\boldsymbol{x}, \boldsymbol{y})$ is the result of the Soft Hamming scheme. For example, $e^{(s)} = 3$ corresponds to the case that either $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the left-hand-side of 1 or $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the right-hand-side of 2. As another example, $e^{(s)} = 4$ corresponds to the case that $\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \hat{D}(\boldsymbol{x}, \boldsymbol{y}) = 2$. Most calculations from subsection 2.3.4

are applicable in this context by setting $\beta = 0$ since resistance variation is not considered. We recalculate $P_{dp}^{(s)}(x, y, e^{(s)})$ ($P_{dn}^{(s)}(x, y, e^{(s)})$, resp.) which is the probability of an arbitrary positive (negative, resp.) training instance having distance $e^{(s)}$ from the arbitrary testing instance $t(x, y) \in I(x, y)$. This calculation is separated into two cases. We assume, in this subsection, an arbitrary attribute is flipped with probability p.

In the first case, we consider an even value for $e^{(s)}$ where no error is detected. From previous discussion, no error is detected when the number of bit flips from 0 to 1, s, is equals to the number of bit flips from 1 to 0, r. When $e^{(s)}$is even, $P_{dp}^{(s)}(x, y, e^{(s)})$ and $P_{dn}^{(s)}(x, y, e^{(s)})$ can be expressed as:

$$P_{dp}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} P_{dp}(x, y, \hat{e}) P_{dif}^{s=r}(\hat{e}, e^{(s)}/2),\ P_{dn}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} P_{dn}(x, y, \hat{e}) P_{dif}^{s=r}(\hat{e}, e^{(s)}/2), \tag{2.27}$$

where $P_{dp}(x, y, e)$ and $P_{dn}(x, y, e)$ are the probabilities calculated from subsection 2.3.4 and $P_{dif}^{s=r}(\hat{e}, \hat{e}')$ is the probability that $\hat{D}(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}'$ given that $D(\boldsymbol{x}, \boldsymbol{y}) = \hat{e}$ and $s = r$. We can calculate this probability by examining Table 2.2. We observe that each bit flip from 0 to 1, neglecting the fraction parts, will either decrease $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ by 1 or keep it the same. Also, each bit flip from 1 to 0, neglecting the fraction parts, will either increase $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ by 1 or keep it the same. $P_{dif}^{s=r}(\hat{e}, \hat{e}')$ can therefore be calculated as:

$$P_{dif}^{s=r}(\hat{e}, \hat{e}') = \sum_{s=|\hat{e}'-\hat{e}|}^{n+l} \sum_{w=|\hat{e}'-\hat{e}|}^{\min(s+|\hat{e}'-\hat{e}|, s)} \left[\binom{s}{w} (\frac{1}{2})^s \binom{s}{w- \mid \hat{e}' - \hat{e} \mid} (\frac{1}{2})^s \binom{n+l}{s} \binom{n+l}{s} p^{2s} (1-p)^{2n+2l-2s} \right]. \tag{2.28}$$

In the case that $e^{(s)}$ is an odd number, $e^{(s)}$ can arise from two cases: either $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the left-hand-side of $(e^{(s)} - 1)/2$, i.e., $r < s$, or $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ is on the right-hand-side of $(e^{(s)} + 1)/2$, i.e., $r > s$. When $e^{(s)}$is odd, $P_{dp}^{(s)}(x, y, e)$ and $P_{dn}^{(s)}(x, y, e)$ can be expressed as:

$$P_{dp}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} \{P_{dp}(x, y, \hat{e}) [P_{dif}^{s>r}(\hat{e}, (e^{(s)} - 1)/2) + P_{dif}^{s<r}(\hat{e}, (e^{(s)} + 1)/2)]\},$$

$$P_{dn}^{(s)}(x, y, e^{(s)}) = \sum_{\hat{e}=0}^{n+l} \{P_{dn}(x, y, \hat{e}) [P_{dif}^{s>r}(\hat{e}, (e^{(s)} - 1)/2) + P_{dif}^{s<r}(\hat{e}, (e^{(s)} + 1)/2)]\}.$$

$P_{dif}^{s>r}(\hat{e},\hat{e}')$ is defined to be the probability that $\text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) = \hat{e}'$ given $D(\boldsymbol{x},\boldsymbol{y}) = \hat{e}$ and $s > r$. $P_{dif}^{s<r}(\hat{e},\hat{e}')$ is defined to be the probability that $\text{nint}(\tilde{D}(\boldsymbol{x},\boldsymbol{y})) = \hat{e}'$ given $D(\boldsymbol{x},\boldsymbol{y}) = \hat{e}$ and $s < r$.

From Table 2.2, we can express $P_{dif}^{s>r}(\hat{e},\hat{e}')$ and $P_{dif}^{s>r}(\hat{e},\hat{e}')$ as following:

$$
\begin{aligned}
&P_{dif}^{s>r}(\hat{e},\hat{e}') \\
&= \mathbb{1}(\hat{e} > \hat{e}') \sum_{s=\hat{e}-\hat{e}'}^{n+l} \sum_{r=0}^{s-1} \sum_{w=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{w} (\tfrac{1}{2})^{s+r} \binom{r}{w-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}, \\
&+ \mathbb{1}(\hat{e} \le \hat{e}') \sum_{r=\hat{e}-\hat{e}'}^{n+l} \sum_{s=r+1}^{n+l} \sum_{h=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{h} (\tfrac{1}{2})^{s+r} \binom{r}{h-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}
\end{aligned}
$$

$$
\begin{aligned}
&P_{dif}^{s<r}(\hat{e},\hat{e}') \\
&= \mathbb{1}(\hat{e} \ge \hat{e}') \sum_{s=\hat{e}-\hat{e}'}^{n+l} \sum_{r=s+1}^{n+l} \sum_{w=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{w} (\tfrac{1}{2})^{s+r} \binom{r}{w-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}, \\
&+ \mathbb{1}(\hat{e} < \hat{e}') \sum_{r=\hat{e}-\hat{e}'}^{n+l} \sum_{s=0}^{s-1} \sum_{h=\hat{e}-\hat{e}'}^{\min(r+\hat{e}-\hat{e}',s)} \left\{ \binom{s}{h} (\tfrac{1}{2})^{s+r} \binom{r}{h-\hat{e}+\hat{e}'} \binom{n+l}{s} \binom{n+l}{r} p^{s+r}(1-p)^{2n+2l-s-r} \right\}
\end{aligned}
$$

where $\mathbb{1}$ denotes the indicator function.

This ends our derivation for $P_{dp}^{(s)}(x,y,e^{(s)})$ and $P_{dn}^{(s)}(x,y,e^{(s)})$. These two probabilities can then be used in place of $P_{dp}(x,y,e)$ and $P_{dn}(x,y,e)$ from 1.3.4. We proceed with the rest of the calculation in 1.4.5.1 using $e^{(s)}$ instead of e and calculate the classification accuracy accordingly.

2.4.5.2 Average-Case Study Results Using Soft Hamming Scheme

We use the above equations to calculate the expected accuracy of the k-NN classifier where the Hamming distance is computed using the Soft Hamming scheme on the 3-of-5/2 concept. We set $N = 32$ and the accuracy is selected to be the maximum accuracy for $2 \le k \le 16$. The attribute noise levels are characterized by the cross over probabilities of the write BSC, P_{bsc}.

We also plot the expected classification accuracy of the noise-free case and the expected classification accuracy under attribute noise without error-detection/error-correction. We

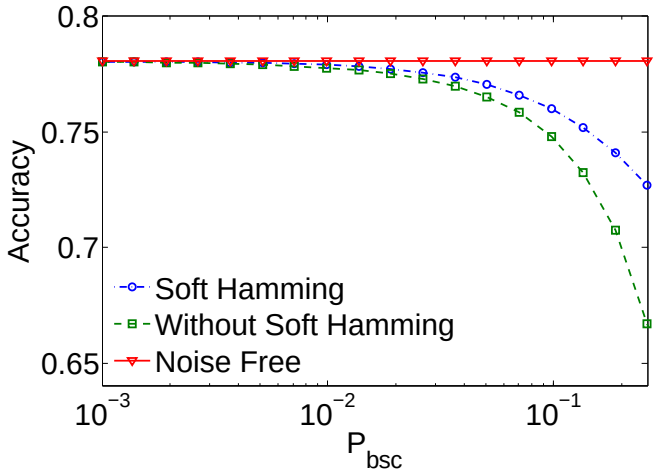


Figure 2.4: Learning accuracy with various levels of attribute noise.

observe that in the region where attribute noise is small, e.g., $p_{bsc} < 10^{-2}$, the attribute noise has little effect on classification accuracy. In the region where attribute noise is large, e.g., $p_{bsc} > 10^{-2}$, our Soft Hamming scheme can successfully improve the classification accuracy under attribute noise.

2.5 Error-Detection and Error-Correction under Resistance Variation and Write BSC

In previous sections, we proposed HD-EIM to feasibly compute Hamming distance in resistive memory under the adverse effect of memristor variability due to resistance variation. We also provided the Soft Hamming scheme to deal with single bit error introduced by memristor variability due to nondeterministic switching mechanism. In this section, we seek to combine the two approaches in order to deal with memristor variability due to both sources. Again, we suppose $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$ are inversion coded vectors to be stored. And we define $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$ to be the noisy vectors actually stored due to write BSC. In order to make the math tractable, we provide a solution to a simpler problem, where $\sigma_H^2 = \sigma_L^2 = \sigma_0^2$.

With resistance variation taken into account, the normalized conductance measurement

is a random variable that is additionally affected by the write BSC. Combining Equations (2.14) and (2.21), we have the distribution of the normalized conductance measurement as follows:

$$\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)}) \sim \mathcal{N}\left(\tilde{N}'_{11} + (\tilde{N}'_{01} + \tilde{N}'_{10})2\epsilon + \tilde{N}'_{00}\epsilon, \frac{\tilde{N}'_{11}\sigma_0^2}{2\mu_H^2} + \frac{(\tilde{N}'_{01} + \tilde{N}'_{10})8\sigma_0^2}{2\mu_H^2} + \frac{\tilde{N}'_{00}\sigma_0^2}{2\mu_H^2}\right). \quad (2.29)$$

Here $\tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})$ is a random variable denoting the normalized conductance measurement between rows of memristors that store $\tilde{\boldsymbol{x}}^{(c)}$ and $\tilde{\boldsymbol{y}}^{(c)}$. $\tilde{N}'_{gh}$ is defined to be the number of coordinates having bit g in $\boldsymbol{x}^{(c)}$ and bit h in $\boldsymbol{y}^{(c)}$, for $g, h \in \{0, 1\}$. We use the following equation to calculate an intermediate random variable $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$:

$$\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1 - 3\epsilon}[n + n\epsilon - \tilde{G}(\tilde{\boldsymbol{x}}^{(c)}, \tilde{\boldsymbol{y}}^{(c)})]. \quad (2.30)$$

Once again, there are four different fundamental types of errors for a pair of elements $x_i^{(c)}$ and $y_i^{(c)}$, $(0,0) \to (0,1)$, $(0,1) \to (0,0)$, $(0,1) \to (1,1)$ and $(1,1) \to (0,1)$, which we define them as error types A, B, C, and D, respectively.

Let us define a random variable E that represents the error type of a single bit error and whether the error occurs or not.

$$P(E) = \begin{cases} p_e, & \text{if} \quad E = A, B, C, D \\ 1 - 4p_e & \text{if} \quad E = 0, \end{cases}$$

where $E = A, B, C, D$ stands for the case that a single type A, B, C or D error occurs, respectively; $E = 0$ stands for the error-free case. We define p_e to be the probability that a single bit error occurs in a pair of coded vector $\boldsymbol{x}^{(c)}$ and $\boldsymbol{y}^{(c)}$, which can be calculated from the BSC parameter.

We then study the conditional distribution of $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$. Given $D(\boldsymbol{x}, \boldsymbol{y})$, for the error-free case, i.e., E = 0, $\tilde{D}(\boldsymbol{x}, \boldsymbol{y})$ assumes the following distribution:

$$\mathcal{N}\left(D(\boldsymbol{x}, \boldsymbol{y}), \frac{(n + 7D(\boldsymbol{x}, \boldsymbol{y}))\sigma_0^2}{\mu_H^2(1 - 3\epsilon)^2}\right). \quad (2.31)$$

Table 2.3: Fundamental error types under resistance variation.

Error Type	Mean	Variance
A	$D(\boldsymbol{x},\boldsymbol{y})-\gamma$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})+7/2)$
B	$D(\boldsymbol{x},\boldsymbol{y})+\gamma$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})-7/2)$
C	$D(\boldsymbol{x},\boldsymbol{y})-\gamma-1$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})-7/2)$
D	$D(\boldsymbol{x},\boldsymbol{y})+\gamma+1$	$\lambda(n+7D(\boldsymbol{x},\boldsymbol{y})+7/2)$

In the Table 2.3, we summarize the distribution of $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$ for a single error of each type, in terms of its mean and variance. To simplify notation, we define $\gamma=\frac{\epsilon}{1-3\epsilon}$ and $\lambda=\frac{\sigma_0^2}{\mu_H^2(1-3\epsilon)^2}$. Table 2.3 is derived by examining (2.29) and Equation (2.30) for each error type.

From Table 2.3 and Equation (2.31), we have the conditional pdf, $f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})\mid D(\boldsymbol{x},\boldsymbol{y}),E)$ for each pair of $D(\boldsymbol{x},\boldsymbol{y})\in\{0,...,n\}$ and $E\in\{0,A,B,C,D\}$. We can therefore estimate E and $D(\boldsymbol{x},\boldsymbol{y})$ given the observation $\tilde{D}(\boldsymbol{x},\boldsymbol{y})$. Denoting our estimation of the pair $\{E,D(\boldsymbol{x},\boldsymbol{y})\}$ to be the pair $\{\hat{E},\hat{D}(\boldsymbol{x},\boldsymbol{y})\}$, respectively, we have the following MAP (maximum *a posteriori* probability) estimator:

$$\begin{aligned}
\{\hat{E},\hat{D}(\boldsymbol{x},\boldsymbol{y})\} &= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(D(\boldsymbol{x},\boldsymbol{y}),E\mid\tilde{D}(\boldsymbol{x},\boldsymbol{y}))\\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad \frac{f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})\mid D(\boldsymbol{x},\boldsymbol{y}),E)f(D(\boldsymbol{x},\boldsymbol{y}),E)}{f(\tilde{D}(\boldsymbol{x},\boldsymbol{y}))}\\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})\mid D(\boldsymbol{x},\boldsymbol{y}),E)f(D(\boldsymbol{x},\boldsymbol{y}),E) \qquad (2.32)\\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})\mid D(\boldsymbol{x},\boldsymbol{y}),E)P(E)P(D(\boldsymbol{x},\boldsymbol{y}))\\
&= \underset{\{E,D(\boldsymbol{x},\boldsymbol{y})\}}{\arg\max} \quad f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})\mid D(\boldsymbol{x},\boldsymbol{y}),E)P(E).
\end{aligned}$$

Next, we present a solution to the above MAP estimator. Define $s_m(D_H)$ for $D_H\in$

$\{0, ..., n-1\}$ to be a solution of equation:

$$f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B) = f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = A), \tag{2.33}$$

such that $D_H + \gamma < s_m(D_H) < D_H + 1$.

Note that for Equation (2.33), there always exists a solution that satisfies the constraint of $s_m(D_H)$ therefore $s_m(D_H)$ always exist.

We define $s_0^{(1)}$ and $s_0^{(2)}$ to be solutions of the following equation:

$$f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = A) = f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = C), \tag{2.34}$$

such that:

$$-1 - \gamma < s_0^{(1)} < -\gamma, \text{and } s_0^{(2)} < -1 - \gamma.$$

We also define s_n to be a solution of the equation:

$$f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = B) = f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = D), \tag{2.35}$$

such that:

$$n + \gamma < s_n < n + 1 + \gamma.$$

We define $s_+(D_H)$ for $D_H \in \{0, ..., n\}$ to be a solution, if exist, of the equation:

$$(1 - 4p_e) f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = 0) = p_e f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B), \tag{2.36}$$

such that:

$$D_H < s_+(D_H) < s_m(D_H) \quad \text{for} \quad D_H \in \{0, ..., n-1\},$$
$$n < s_+(D_H) < s_n \quad \text{for} \quad D_H = n.$$

Similarly, we define $s_-(D_H), D_H \in \{0, ..., n\}$ to be a solution, if exist, of the equation:

$$(1-4p_e) f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = 0) = p_e f(\tilde{D}(\boldsymbol{x}, \boldsymbol{y}) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = A), \tag{2.37}$$

such that:

$$s_m(D_H) < s_-(D_H) < D_H \quad \text{for} \quad D_H \in \{1, ..., n\},$$
$$s_0^{(1)} < s_-(D_H) < 0 \quad \text{for} \quad D_H = 0.$$

Note that there exist cases where none of the solutions of Equation (2.36) or Equation (2.37) satisfies the constraint of $s_+(D_H)$ or $s_-(D_H)$. Hence $s_+(D_H)$ or $s_-(D_H)$ may not exist for certain device parameters.

If $s_+(D_H)$ and $s_-(D_H)$ exist for all D_H, and the following conditions are true:

$$\begin{aligned} &p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B) = p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = A) \\ \geq &(1 - 4p_e) f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = 0), \quad \text{for} \quad D_H \in \{0, ..., n-1\}; \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H, E = B) = p_e f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = A) \\ \geq &(1 - 4p_e) f(s_m(D_H) \mid D(\boldsymbol{x}, \boldsymbol{y}) = D_H + 1, E = 0), \quad \text{for} \quad D_H \in \{0, ..., n-1\}; \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = A) = p_e f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = C) \\ \geq &(1 - 4p_e) f(s_0^{(1)} \mid D(\boldsymbol{x}, \boldsymbol{y}) = 0, E = 0); \end{aligned}$$

and

$$\begin{aligned} &p_e f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = B) = p_e f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = D) \\ \geq &(1 - 4p_e) f(s_n \mid D(\boldsymbol{x}, \boldsymbol{y}) = n, E = 0); \end{aligned}$$

then the MAP estimator has a solution as follows:

$$
\begin{aligned}
&\text{If } s_-(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_+(D_H) \quad \text{for some } D_H, \text{ then } \hat{E}=0, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H;\\
&\text{if } s_+(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_m(D_H) \quad \text{for some } D_H,\\
&\quad \text{then } \hat{E}=B, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H, \text{ or } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H - 1;\\
&\text{if } s_m(D_H-1) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(D_H) \quad \text{for some } D_H,\\
&\quad \text{then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H, \text{ or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H + 1;\\
&\text{if } s_0^{(1)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(0) \text{ then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0, \text{or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1;\\
&\text{if } s_+(n) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_n \text{ then } \hat{E}=B, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n, \text{or } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n-1;\\
&\text{if } s_0^{(2)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(1)}, \text{ then } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) > s_n, \text{ then } \hat{E}=D, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(2)}, \text{ then } \hat{E}=A, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0, \text{or } \hat{E}=C, \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1.
\end{aligned}
\tag{2.38}
$$

Note that the MAP estimator has multiple solutions when an error is detected, i.e., $E \neq 0$. This is similar to what we observed in Section 2.4 and it is due to the underlying problem, e.g., $f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})|D(\boldsymbol{x},\boldsymbol{y}) = D_H, E = B) = f(\tilde{D}(\boldsymbol{x},\boldsymbol{y})|D(\boldsymbol{x},\boldsymbol{y}) = D_H - 1, E = D)$. As a solution, we adapt the same idea of the Soft Hamming scheme and propose the following estimator:

$$
\begin{aligned}
&\text{If } s_-(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_+(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H;\\
&\text{if } s_+(D_H) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_m(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H - 1/2;\\
&\text{if } s_m(D_H-1) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(D_H) \quad \text{for some } D_H, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = D_H + 1/2;\\
&\text{if } s_0^{(1)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_-(0), \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 1/2;\\
&\text{if } s_+(n) < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_n, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n - 1/2;\\
&\text{if } s_0^{(2)} < \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(1)}, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = 0;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) > s_n, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n;\\
&\text{if } \tilde{D}(\boldsymbol{x},\boldsymbol{y}) \le s_0^{(2)}, \text{ then } \hat{D}(\boldsymbol{x},\boldsymbol{y}) = n - \frac{1}{2}.
\end{aligned}
\tag{2.39}
$$

Note that we do not specify the error type in the above estimator. $\hat{D}(\boldsymbol{x}, \boldsymbol{y})$ is estimated by taking the average between two candidate estimation in Estimator (2.38). This estimator therefore can successfully estimate the Hamming distance between vectors in resistive memory under the adverse effect of resistance variation while also have the capability to detect and correct a single bit error caused by the write BSC. Also note that the threshold values, e.g., $s_{\pm}(D_H)$, can be precomputed and stored in table to improve efficiency.

CHAPTER 3

Pilot Assisted Adaptive Thresholding for Sneak-Path Mitigation in Resistive Memories with Failed Selection Devices

3.1 Introduction

Crossbar Resistive random-access memory (ReRAM), in which a memristor is positioned on each row-column intersection of the crossbar structure, is considered to be a promising candidate to be used as a next generation non-volatile memory device because of its many unique advantages including a simple structure and high density [SSS+08]. One fundamental problem in purely passive crossbar ReRAM that demands detailed attention is the *sneak-path* problem [ZFH+13]. When a cell in a crossbar array is read, a voltage is applied to the memristor and the resistance is measured to determine whether it is in Low-Resistance State (logic 1) or High-Resistance State (logic 0). Sneak paths are undesirable paths in parallel to the selected cell; they traverse through unselected cells. This problem is especially severe when a cell in High-Resistance State (logic 0) is read because parallel low resistances, due to sneak-paths, lower the resistance measured from the cell at High-Resistance State, thus causing difficulties in distinguishing between the High-Resistance State and the Low-Resistance State. The sneak-path problem is commonly solved by adding a selection device in series of each memory cell [DHC+13]. Adding diodes results in the 1D1R structure [ZDL+13, TGK+11]; adding transistors results in the 1T1R structure [CGC+13]; and adding selectors results in the 1S1R structure [HTL+11].

Although introducing selection devices can eliminate the sneak-path problem, selection devices are also prone to failure [AN97, Han88]. As modern storage devices have ultra high reliability requirements, it is of interest to study crossbar resistive memories with failed selection devices. A study on crossbar resistive memories with failed selection devices can also aid a memory architecture designer in understanding how selection device reliability affects the memory reliability. In this work, we first focus on 1D1R structured arrays with shorted diodes, and then we generalize the results to 1S1R structured arrays with shorted selectors [BHC17]. The case that diodes/selectors fail open is not considered in this work. In this case, the information stored in the cells to which the open diodes/selectors are connected, is lost, and this case is best treated with erasure correcting codes.

In a 1D1R or 1S1R structured array, the sneak-path problem re-occurs when some diodes or selectors fail short, motivating the study on the re-occurred sneak-path problem and approaches to mitigate it. Similar to what is observed in [CKY16], the events that cells on the same row/column are affected by the re-occurred sneak path problem are highly dependent. This dependency causes natural difficulties for standard coding theoretic solutions which assume independent bit-errors. Investigation of alternative techniques which can utilize this dependency is therefore of interest.

In Chapter 3, our goal is to study the re-occurred sneak-path problem and to provide simple yet effective approaches to mitigate it. When viewing the effect of the re-occurred sneak-path problem as a *parallel interference*, the dependency between cells can be utilized to mitigate the re-occurred sneak-path problem. From an estimation theoretic point of view, we utilize the dependency between cells to provide improved estimation schemes based on our probabilistic characterization of the inter-cell dependency. We propose adaptive thresholding schemes that adaptively change the threshold using side information provided by pilot cells. With a simple pilot construction and our proposed adaptive thresholding schemes, we demonstrate up to 50% reduction in bit-error rate (BER) comparing to the fixed threshold scheme. In addition to ReRAM that is built out of memristors, our approaches

are also applicable to other memory with crossbar structure, such as the Phase Changing Memory (PCM).

The content of this chapter is organized as follows. Section 3.2 provides modeling of the re-occurred sneak-path event and its adverse effect in two types of memory structure. The pilot construction used in this chapter is also introduced in Section 3.2. Section 3.3 formally characterizes the dependency of the re-occurred sneak-path event between the pilot cells and the information cells by calculating joint probabilities of the re-occurred sneak-path events among cells. In Section 3.4, the adaptive thresholding idea is introduced and the thresholds for different cases are derived. Section 3.5 provides bit-error rate analysis and present results comparing our proposed scheme with the fixed threshold scheme for various parameters. We present a case study comparing our proposed schemes with a comparable coding theoretic solution in Section 3.6.

3.2 Sneak path Modeling and the Diagonal-0 Coding

In this chapter, since it is clear that we are addressing the "re-occurred" sneak-path problem, we drop the repeated attribute "re-occurred" for simplified language. As an initial step, we use the sneak-path definition in [CKY16] with modification to model the sneak-path event. In this work, we restrict ourselves to the sneak-path of length 3 because a sneak-path of length 3 is more likely to occur than sneak-paths of other lengths and the adverse effect of a sneak-path of length 3 is more prominent than the adverse effects of a sneak-path of other lengths. Other factors that affect the occurrence of the sneak-path problem, such as wire resistance, are not considered in this chapter and are left for future work. In this section, we first propose the model of a sneak-path event for crossbar resistive memory with the 1D1R structure and then extend the model to the 1S1R structured array. Our models assume that the selection devices fails short i.i.d. with probability p_f due to reliability issues. Note that these models can be easily extended to crossbar resistive memory without any selection device

by setting $p_f = 1$. We define $A \in \{0,1\}^{m \times n}$ to be the data matrix representing data stored in a crossbar resistive memory of size $m \times n$, and let A_{ij} denote the bit value at cell (i,j). For simplicity, we assume $m \leq n$ and that m divides n with $n/m = r$. The case when m does not divide n will be briefly discussed at the end of subsection 3.2.4. After presenting the modeling of the sneak-path event for the two different crossbar resistive memory structures, we describe a modeling of the adverse effect of a sneak-path event. At the end of this section, we provide the pilot construction that we use throughout this chapter.

3.2.1 Modeling of the Sneak-path Event for the 1D1R Structure

By our definition, a sneak-path event occurs at cell (i,j) for an array with a 1D1R structure if the following three conditions are met:

1. The bit value stored is 0.
2. There exists at least one combination of $c \in [1, \cdots, m], s \in [1, \cdots, n], c \neq j, r \neq i$ that induces a sneak-path, defined by
$$A_{ic} = A_{sc} = A_{sj} = 1. \tag{3.1}$$
3. The diode at cell location (s,c) fails short.

We define e_{ij} to be a boolean random variable denoting the occurrence of the sneak-path event at location (i,j), conditioned on the bit value stored at cell (i,j) being 0. That is, $e_{ij} = 1$ if and only if the cell storing a 0 at position (i,j) incurs a sneak-path event. We also refer to the realization of e_{ij} as the sneak-path state of cell (i,j). Note that the sneak-path event is defined only for those cells that store 0's because the adverse effect of a sneak-path event on a cell that stores 1 is not detrimental to the read process. We have the third condition because in a 1D1R structured array, the diode in series prevents current from flowing in the reverse direction. The failure of the diode at cell location (s,c) to short circuit is therefore necessary to allow current flowing along the sneak-path.

3.2.2 Modeling of the Sneak-path Event for the 1S1R Structure

The only difference between our definition for the sneak-path event in a 1S1R structured array and the definition in a 1D1R structured array is the third condition. The third condition for a sneak-path event to occur at cell (i, j) in a 1S1R structured array is:

- The selectors at cell location (i, c), (s, c) and (s, j) fail short.

A cell selector is a device that has a high resistance when the voltage across it is below a certain threshold, e.g., when the cell is on the sneak-path. Therefore, all three selectors in series of the three cells on the sneak-path, i.e., cells that store A_{ic}, A_{sc} and A_{sj}, need to fail short for the sneak-path event to occur. Modeling for the 1D1R and 1S1R structures share the same notation. In this chapter, when it is necessary to differentiate these two set-ups, it will be clear from the context which one is being considered.

3.2.3 Adverse Effect of a Sneak-Path Event

As this is an initial study, and with limited prior work, we focus on a simplified model described as follows. Our modeling of the adverse effect of a sneak-path event is adapted from [BHC17] as a parallel interference. We first define the 0 state resistance of the memristor to be R_0, and the 1 state resistance of the memristor to be R_1. The adverse effect of a sneak-path event is modeled as a parasitic resistor with value R_s that is parallel to the read cell. We then denote r_{ij} to be the measured resistance value of cell (i, j), with an additive noise η. This additive noise captures two noise sources: the noise introduced by the resistance variation of resistive memory cells [CL11], and the measurement noise introduced by the sensing circuit [BHC17]. To consider the noise introduced by the resistance variation, we first model the noise caused by the variation of HRS resistance and LRS resistance as Gaussian [CLG+11, WLW+10]. As shown in [CSD18a], parallel combination of two Gaussian random variables can be approximated with another Gaussian random variable. As a result, we also

model the noise caused by the variation of the sneak-path resistance as a Gaussian additive noise added to the parallel combination of the sneak-path resistance and HRS resistance. Collectively, and for mathematical simplicity, we assume η to be Gaussian with mean 0 and variance σ^2, i.e., the same noise is added to the LRS resistance, the HRS resistance, and the parallel combination of the sneak-path resistance and HRS resistance. Adaptation to different noise models, including Gaussians with different variances are discussed at the end of subsection 3.4.2.

Together, we have the following model:

$$r_{ij} = \begin{cases} \left(\frac{1}{R_0} + \frac{e_{ij}}{R_s}\right)^{-1} + \eta & \text{when 0 is stored,} \\ R_1 + \eta & \text{when 1 is stored.} \end{cases} \tag{3.2}$$

In this chapter, we assume $R_1 < (1/R_0 + 1/R_s)^{-1}$ to avoid analyzing degenerate scenarios.

3.2.4 Pilot Construction

In a crossbar resistive memory, and as noted in [CKY16], it is not hard to observe that the occurrence of a sneak-path event at one cell is *not* independent of the occurrence of a sneak-path event at another cell. For example, knowing that $e_{ij} = 1$ increases the probability of $e_{ij^*} = 1, j^* \in [1, \cdots, n], j^* \neq j$, as well as $e_{i^*j} = 1, i^* \in [1, \cdots, m], i^* \neq i$. This special behavior of resistive memory presents natural difficulties to standard coding theoretic solutions when viewing the occurrence of a sneak-path event as a bit error. However, when viewing the effect of a sneak-path event as a parallel interference, one can utilize this inter-cell dependency to develop better estimation schemes based on side information provided by cells with known bit values.

We note that two cells are correlated the most when they are on the same row or column. It is also observed in [ZFH+13] that the location of the cell (i and j) does not affect the probability of e_{ij}. Moreover, the event that two cells in the same row (column) simultaneously

incur sneak-path events is also independent of the relative position of cells, i.e., the (joint) probability that these two cells simultaneously incur sneak-path events is independent of their positions, given that we do not consider wire resistance in this study. As a result, knowledge of the occurrence of a sneak-path event at a cell provides the same information as for all other cells on the same row (column). Based on this observation, we wish to explore robustness to selection device failure when a cell is allowed to access a few other cells with known states by utilizing their inter-cell dependency. The simplest yet effective case is that each information cell (a cell that can store either 0 or 1), has at least one pilot cell (a cell with known state) on its row and at least one pilot cell on its column. We therefore propose the following pilot construction.

Pilot Construction 1. *Let $A \in \{0,1\}^{m\times n}$ denote the data matrix representing data stored in a crossbar resistive memory array of size $m\times n$, and let A_{ij} denote the bit value at cell (i,j). We assume $m \leq n$ (A is fat) and m divides n with $n/m = r$. We preset all cells (i,j) such that $i \equiv j \pmod m$ to store 0, i.e., $A_{ij} = 0$, if $\quad i \equiv j \pmod m, \forall i \in \{1,...,m\}, j \in \{1,...,n\}$. These preset cells are pilot cells and the rest of the cells are information cells.*

For an array of size $m \times n$, to have at least one pilot cell on the row of each information cell, we need at least m pilot cells. To have at least one pilot cell on the column of each information cell, we need at least n pilot cells. Together, to have at least one pilot cell on both row and column of each information cell, we need at least n pilot cells. Therefore the rate is at most $(m-1)/m$. It is easy to check that Pilot construction 1 achieves this upper bound and satisfies our pilot cell requirements.

The general term "pilot cells" can be thought of as the set of some pre-selected cells that are accessed first in order to infer certain characteristics of the array. In lithium battery, pilot cells are tested to determine the charge and discharge current for the entire battery. In [**?**], pilot cells are used to estimate the sneak current in resistive memory. In our work, pilot cells are used to estimate the sneak-path states of other cells on their rows and columns. Specifically, we use the resistance of pilot cells with known resistance states to infer the

probabilities of other cells incurring the sneak-path events. We provide details on the usage of pilot cells in both Section 3.3 and Section 3.4.

With this simple pilot construction, each cell at location (i, j), with $i \not\equiv j \pmod{m}$, has 1 pilot cell that stores a 0 in its column and r pilot cells that store 0's in its row. We will use these pilot cells for our improved estimation schemes in latter section. Although there are r pilot cells in the row of an information cell, we only use one of them. Here and elsewhere, we define $j' = j \mod m + m\mathbb{1}(j \mod m = 0), i' = m\lfloor j/m \rfloor + i - m\mathbb{1}(j \mod m = 0)$ where $\mathbb{1}(\cdot)$ is the indicator function. For an information cell that stores A_{ij}, with $i \not\equiv j \pmod{m}$, we refer to the cells that store $A_{j'j}$ and $A_{ii'}$ to be its two reference cells. This construction generalizes the construction in [CSD18b] that is only applicable to a square array, i.e., $m = n$. For an array where m does not divide n, Pilot Construction 1 is still applicable. In this case, some of the information cells at the lower right side of the array do not have reference cells in their rows as defined before. Instead, they can use other pilot cells on their rows as the reference cells. The probabilistic characterization for this case could be done using the same techniques that will be discussed in the following section. We omit the explicit discussion of it because, while considerably more tedious, it does not provide further insight.

3.3 Probabilities and Joint Probabilities of the Sneak path Event

In the previous section, we proposed a pilot construction for the resistive memory array. In order to utilize the known 0's stored in pilot cells for more informed estimation schemes, several important probabilities need to be calculated analytically. First, in order to determine the sneak-path state of a pilot cell, we need $P_0(e_{ij})$ for $i \equiv j \pmod{m}$. Second, in order to use the sneak-path state of pilot cells for improved estimation schemes, we need $P(e_{ij}|e_{ii'}, A_{ij} = 0)$, $P(e_{ij}|e_{j'j}, A_{ij} = 0)$ and $P(e_{ij}|e_{ii'}, e_{j'j}, A_{ij} = 0)$ for $i \not\equiv j \pmod{m}$. The three sets of conditional probabilities are used in different estimation schemes in latter sections. To make a comparison with the scheme that does not use any side information, we also need

$P(e_{ij}|A_{ij} = 0)$. Although similar probabilistic characterizations of sneak-path event(s) for a single cell and for two cells can be found in [CKY16], modifications need to be made to consider the pilot construction. In addition, characterizing dependency between three cells is necessary to allow for a more accurate estimation scheme.

To keep the discussion clear, in this section, we simply state the probabilities that are used to obtain the necessary probabilities mentioned above, and we refer readers to Appendix B for the lemmas that calculate these probabilities. All probabilities that we stated are conditioned on the usage of our pilot construction, and, for clarity, we omit explicitly specifying this condition. Apart from $P_0(e_{ij})$, the probabilities stated in this section also conditioned on the event $A_{ij} = 0$. We also omit specifying this condition explicitly, and instead use the subscript 0 to highlight that $P_0(e_{ij})$ is not conditioned on a known information cell. We assume that the bit values to be stored in information cells are chosen i.i.d. Bernoulli with parameter q representing the prior probability of a 1 being stored. Note that although we include the indexes i and j in the argument of probabilities, these probabilities are independent of i and j. We only require i and j to satisfy certain conditions based on the context.

All probabilities have two versions, one for the 1D1R structure and one for the 1S1R structure. In this section and the following sections, we do not differentiate them in most cases unless specified. In Appendix B, we provide lemmas that calculate the stated probabilities for the 1D1R structure. The corresponding probabilities for the 1S1R structure can be calculated using the following claim, which readily follows from the definitions of the sneak-path event in subsection 3.2.1 and subsection 3.2.2.

Claim 3. *For a 1S1R structured array, let $p_f^{(1S1R)}$ be the selector failure probability, and $q^{(1S1R)}$ be the prior probability of a 1 being stored. In order to calculate a certain probability for the 1S1R structured array for which the corresponding equation for the 1D1R structure is already available, it suffices to use that equation for the 1D1R structure with substitution: $p_f = 1$ and $q = q^{(1S1R)} \times p_f^{(1S1R)}$.*

To obtain the set of probabilities $P_0(e_{ij})$, we calculate $P_0(e_{ij} = 0)$ (Lemma 8 in Appendix B). To obtain the set of probabilities $P(e_{ij})$, we calculate $P(e_{ij} = 0)$ (Lemma 9 in Appendix B). There are two reference cells for each information cell. We first consider the case of using the sneak-path state of only one reference cell. To obtain probabilities of form $P(e_{ij}|e_{ii'})$ and $P(e_{ij}|e_{j'j})$, we first need $P(e_{ij} = 0, e_{ii'} = 0)$ (Lemma 10 in Appendix B) and $P(e_{ij} = 0, e_{j'j} = 0)$ (Lemma 11 in Appendix B). We also need $P(e_{ii'} = 0)$ (Lemma 12 in Appendix B) and $P(e_{j'j} = 0)$ (Lemma 13 in Appendix B). When using the sneak-path states of two reference cells, we need $P(e_{ij}|e_{ii'}, e_{j'j})$. To obtain the set of probabilities $P(e_{ij}|e_{ii'}, e_{j'j})$, we first need $P(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0)$ (Lemma 14 in Appendix B). We also need the set of probabilities $P(e_{ii'}, e_{j'j})$ obtained by calculating $P(e_{ii'} = 0, e_{j'j} = 0)$ (Lemma 15 in Appendix B).

3.4 Adaptive Thresholding Schemes

With all necessary conditional probabilities calculated, we propose our adaptive thresholding schemes. We propose two schemes, the *Single Reference Scheme* based on a single reference cell, and the *Double Reference Scheme* based on two reference cells. The *Single Reference Scheme* is further separated into two sub-schemes, the *Single Reference (Row) Scheme* that uses the reference cell on the row of an information cell, and the *Single Reference (Column) Scheme* that uses the reference cell on the column of an information cell. We need to consider both the *Single Reference (Row) Scheme* and the *Single Reference (Column) Scheme* because in a strictly non-square array, due to asymmetry, the two reference cells do not provide the same side information. For comparison, we also state the *No Reference Scheme* which uses no side information. For the two adaptive thresholding schemes, first we determine the sneak-path states of the reference cells for a targeted information cell, then, based on these sneak-path states, we choose appropriate thresholds to decide the states of the information cells to be read. All decisions are made through threshold estimators for implementation simplicity. In this section and the next section, the probability density function of the

Gaussian noise η in Equation (3.2) is defined to be $f_\eta(\cdot)$.

In the remainder of this section, we first calculate the threshold of the threshold estimator that determines the sneak-path states of pilot cells. Then, we calculate the thresholds of the threshold estimators that determine the states of information cells, assuming known sneak-path states. Afterwards, we further discuss the proposed adaptive thresholding schemes that are based on the derived threshold estimators.

3.4.1 Optimal Threshold Estimation for Pilot Cells

We define τ_0 to be the threshold of the threshold estimator used to determine whether or not a sneak-path event occurs at a pilot cell. For a given resistance measurement r_{ij}, taken for a pilot cell (i, j) where $i \equiv j \pmod{m}$, the output of this threshold estimator is

$$\hat{e}_{ij} = \begin{cases} 1 & \text{if} \quad 0 \leq r_{ij} \leq \tau_0, \\ 0 & \text{if} \quad \tau_0 \leq r_{ij} \leq \infty. \end{cases} \tag{3.3}$$

Next, we derive the optimal threshold for this threshold estimator. For an arbitrary pilot cell that stores $A_{ij} = 0$ with $i \equiv j \pmod{m}$, there are two hypotheses, $e_{ij} = 0$ and $e_{ij} = 1$. Based on our modeling in Equation (3.2), the posterior functions of the two hypotheses can be expressed as,

$$\Lambda_{e_{ij}=0}(r_{ij}) = f_\eta\left(r_{ij} - R_0\right) P_0(e_{ij} = 0), \tag{3.4}$$

and

$$\Lambda_{e_{ij}=1}(r_{ij}) = f_\eta\left(r_{ij} - \left(\frac{1}{R_0} + \frac{1}{R_s}\right)^{-1}\right) P_0(e_{ij} = 1), \tag{3.5}$$

where $P_0(e_{ij} = 0)$ and $P_0(e_{ij} = 1)$ can be calculated using Lemma 8 in Appendix B.

Minimizing the average error probability of this threshold estimator by Bayes Criterion gives the condition:

$$\Lambda_{e_{ij}=1}(\tau_0) = \Lambda_{e_{ij}=0}(\tau_0). \tag{3.6}$$

Solving (3.6) gives the following optimal threshold:

$$\tau_0 = \frac{1}{2}\frac{R_0^2 - \left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-2} + 2\sigma^2 \log\left(\frac{P_0(e_{ij}=1)}{P_0(e_{ij}=0)}\right)}{R_0 - \left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}}. \tag{3.7}$$

3.4.2 Optimal Threshold Estimation for Information Cells

In this subsection, we calculate the optimal thresholds for the three thresholding schemes, assuming the sneak-path states of the pilot cells are known. The three thresholding schemes are alike and only differ in the side-information used for each scheme. It is thus convenient to use c to denote the side-information, i.e., sneak-path state(s), used in each scheme to describe the three schemes collectively. Depending on the corresponding scheme, c may be nothing, a scalar or a vector. For an arbitrary information cell with $i \not\equiv j \pmod m$, we let $c = none$ for the *No Reference Scheme*, $c = e_{ii'}$ for the *Single Reference (Row) Scheme*, $c = e_{j'j}$ for the *Single Reference (Column) Scheme*, and $c = [e_{ii'}, e_{j'j}]$ for the *Double Reference Scheme*.

There are two hypotheses for an information cell (i, j), $A_{ij} = 0$ and $A_{ij} = 1$. We are interested in the threshold estimators that decide between these two hypotheses, while considering different realizations of c. Let $\tau(c)$ be the threshold used for a particular thresholding scheme under a certain realization of the corresponding c. For example, $\tau(none)$ is the threshold to be used when no sneak-path state of reference cells is used; $\tau(e_{ii'} = 1, e_{j'j} = 1)$ is the threshold to be used when the *Double Threshold Scheme* is used and both reference cells of this information cell incur sneak-path events. Therefore, when reading cell (i, j), the output of the threshold estimator, using the selected threshold $\tau(c)$ where c is known sneak-path state(s) of reference cell(s), is:

$$\hat{A}_{ij} = \begin{cases} 1 & \text{if} \quad 0 \le r_{ij} \le \tau(c), \\ 0 & \text{if} \quad \tau(c) \le r_{ij} \le \infty. \end{cases} \tag{3.8}$$

Based on the selected thresholding scheme and the realization of c, the posterior functions

of each hypothesis as a function of r_{ij} are

$$\Lambda_{A_{ij}=0}(r_{ij}) = (1-q)\left[f_\eta(r_{ij}-R_0)P(e_{ij}=0|c) + f_\eta\left(r_{ij} - \left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}\right)P(e_{ij}=1|c)\right], \tag{3.9}$$

and

$$\Lambda_{A_{ij}=1}(r_{ij}) = q f_\eta(r_{ij}-R_1), \tag{3.10}$$

where $P(e_{ij}=0|c)$ and $P(e_{ij}=1|c)$ can be calculated using Lemmas 9-15 in Appendix B.

Minimizing the average error probability of the threshold estimator by Bayes Criterion gives the following condition for the optimal threshold:

$$\Lambda_{A_{ij}=1}(\tau(c)) = \Lambda_{A_{ij}=0}(\tau(c)). \tag{3.11}$$

Note that unlike in Equation (3.6) where both sides have simple Gaussian density functions, Equation (3.11) has the right hand side expressed as an addition of two Gaussian density functions. As a result, Equation (3.11) has two solutions, and the solution lying in the middle of the two distributions (3.9) and (3.10) is the desired threshold that minimizes the error probability. We can solve $\tau(c)$ easily using available numerical methods, e.g., using *vpasolve*($\Lambda_{A_{ij}=1}(\tau(c)) = \Lambda_{A_{ij}=0}(\tau(c))$,$[R_1, R_0]$) in MATLAB.

We quickly remark that in the case that a closed form solution of $\tau(c)$ is needed, one can use the following approximation:

$$\tau(c) \approx \frac{1}{2}\frac{\left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-2} - R_1^2 + 2\sigma^2 \log\left(\frac{q}{(1-q)P(e_{ij}=1|c)}\right)}{\left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1} - R_1}. \tag{3.12}$$

This approximation follows by replacing the two Gaussian density functions in (3.9) with a single Gaussian density function, i.e., $(1-q)f_\eta\left(r_{ij} - \left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}\right)P(e_{ij}=1|c)$, which is closest to the Gaussian density function in (3.10). As a result, this approximation is valid

when R_0 and $\left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}$ are sufficiently apart, and when $f_\eta\left(r_{ij}-\left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}\right)P(e_{ij}=1|c) \gg f_\eta(r_{ij}-R_0)P(e_{ij}=0|c)$ around $\left(\frac{1}{R_0}+\frac{1}{R_s}\right)^{-1}$. While being closed-form, this approximation loses the optimality of the threshold estimator. For the BER results in latter section, we use the thresholds that are solved numerically.

Note that although Gaussian additive noise with the same variance and zero mean is used for mathematical simplicity, other models of the additive noise, such as log-normal distribution [GYW$^+$12], or Gaussian additive noise with different variances for the HRS, LRS, and sneak-path resistances, can be easily adopted with appropriate changes in the thresholds calculations, e.g., by using different η in (3.2) and replacing $f_\eta(\cdot)$ with an appropriate density function(s). The case that the diode is partially shorted (ohmic) can be considered as a bias term among the noise modeling for the sneak-path resistances. If the noise distribution is unimodal, the validity of the approximation in (3.12) follows analogously to the Gaussian case.

Also note that, for mathematical tractability, only one sneak-path of length 3 is considered in this chapter. As a result of this simplification, the sneak-path resistance R_s is the dominant term when the sneak-path event occurs. In a more precise model, one storage cell can be affected by multiple sneak-paths with different structures, [BHC19]. Probabilistic characterization of how a single cell is affected by multiple sneak-paths is provided in [BHC19]. Meanwhile, characterization of how multiple cells are affected by differently structured multiple sneak-paths is still an open problem. To adopt the adaptive thresholding schemes, one would need to consider sneak-path(s) with different structures as separate events. For example, the content within the bracket of Equation (3.9) would be replaced with the summation of many terms. Each term would be the product between the pdf of the measured resistance given a sneak-path event with a particular length and structure, and the probability of this sneak-path event given the sneak-path state(s) of the reference cell(s). The sneak-path state of a reference cell would also have multiple states (not just 0 and 1)

when multiple sneak-paths with different structures are considered. In this chapter, given the limited space, we only provide an analysis for the to the elementary case to illustrate the basic idea of adaptive thresholding.

3.4.3 Estimation Using Adaptive Thresholding Schemes

Because the failures of selection devices are hard and are typically found during the writing process, we seek to re-estimate the sneak-path states of pilot cells after a write operation:

- Whenever new data is written to the memory, except for the case when the *No Reference Scheme* is used, measure r_{ij} for each pilot cell (i, j) with $i \equiv j \pmod{m}$. Then use τ_0 to decide $\hat{e}_{ij}$, and store these estimated sneak-path states of pilot cells.

To read an information cell (i, j), i.e., the cell that stores A_{ij} with $i \not\equiv j \pmod{m}$, the following procedure is performed depending on the selected adaptive thresholding scheme:

- If the *No Reference Scheme* is used, measure r_{ij} and use $\tau(none)$ to decide $\hat{A}_{ij}$.
- If the *Single Reference (Row or Column) Scheme* is used, retrieve the previously stored $\hat{e}_{ii'}$ or $\hat{e}_{j'j}$. Then measure r_{ij} and use $\tau(e_{ii'} = \hat{e}_{ii'})$ or $\tau(e_{j'j} = \hat{e}_{j'j})$ to decide $\hat{A}_{ij}$.
- If the *Double Reference Scheme* is used, retrieve the previously stored $\hat{e}_{ii'}$ and $\hat{e}_{j'j}$. Then measure r_{ij} and use $\tau(e_{ii'} = \hat{e}_{ii'}, e_{j'j} = \hat{e}_{j'j})$ to decide $\hat{A}_{ij}$.

From the above statements, we notice that the adaptive thresholding schemes are particularly efficient for applications in which memory cells are read more frequently than they are written. In addition, the *Single Reference (Column) Scheme*, in a fat array, possesses a special property amenable for simple implementation: all memory cells on the same column use the same threshold, which depends on the estimated sneak-path state of the pilot cell in this column. Therefore, this threshold could be hardware configured to be used for the entire column.

3.5 BER Analysis and Results

In Section 3.4, we proposed adaptive thresholding schemes and the associated threshold estimators. Based on these proposed methods, in this section, we derive an analytical result for the bit-error rate (BER), and present a comparison between different schemes.

To read an information cell (i, j), we assume resistance measurements r_{ij}, $r_{ii'}$ and $r_{j'j}$ are available. Which r is being used depends on which thresholding scheme is being selected. Let $c = [e_{ii'}, e_{j'j}]$ denote the true sneak-path states for the two reference cells, and let C denote the support of c. Let $\hat{c}$ denote the estimated sneak-path state(s) for zero ($\hat{c} = none$), one ($\hat{c} = \hat{e}_{ii'}$ or $\hat{c} = \hat{e}_{j'j}$) or two ($\hat{c} = [\hat{e}_{ii'}, \hat{e}_{j'j}]$) reference cells, depending on the selected thresholding scheme, and let $\hat{C}$ be the support of the corresponding $\hat{c}$. Since the sneak-path events involving the two reference cells are only weakly dependent, for the sake of simplified analysis of $P_r(c)$ in Equation (3.13), and without sacrificing qualitative findings, we shall assume that they are independent. Specifically, since $P_r(c)$ is only a weight term in Equation (3.13), unlike in the threshold calculations, this calculation is insensitive to the small error introduced by the independence assumption. For the exact evaluation of $P_r(e_{ii'}, e_{j'j})$, one can use $P_r(e_{ii'} = 0, e_{j'j} = 0) = (1-q)P(e_{ii'} = 0, e_{j'j} = 0|A_{ij} = 0)+qP(e_{ii'} = 0, e_{j'j} = 0|A_{ij} = 1)$,where $P(e_{ii'} = 0, e_{j'j} = 0|A_{ij} = 0)$ is calculated in Lemma 15 and $P(e_{ii'} = 0, e_{j'j} = 0|A_{ij} = 1)$ can be obtained using an argument similar to Lemma 15.

Let $P(\hat{A}_{ij} \neq A_{ij})$ be the average probability of a decision error. We have the following expression:

$$P(\hat{A}_{ij} \neq A_{ij}) = \sum_{c \in C} P_r(c) \sum_{\hat{c} \in \hat{C}} P(\hat{c}|c) P(\hat{A}_{ij} \neq A_{ij}|c, \hat{c}). \tag{3.13}$$

In Equation (3.13), $P_r(c)$ is the probability of certain realizations of c, and can be calculated using $P_r(c) = P_0(e_{ij} = e_{ii'})P_0(e_{ij} = e_{j'j})$; $P(\hat{c}|c)$ is the probability of deciding $\hat{c}$ given c; $P(\hat{A}_{ij} \neq A_{ij}|c, \hat{c})$ is the decision error probability given certain realizations of c and $\hat{c}$, where the latter governs the choice of the threshold. Probabilities $P(\hat{c}|c)$ for the *No Reference*

Scheme and the *Single Reference (Row/Column) Scheme* are summarized in Table 3.1.

Table 3.1: List of $P(\hat{c}|c)$.

No Reference Scheme				
$P(none\|c)$	1			
Single Reference (Row) Scheme				
case	$\hat{e}_{ii'}=0, e_{ii'}=0$	$\hat{e}_{ii'}=1, e_{ii'}=1$	$\hat{e}_{ii'}=1, e_{ii'}=0$	$\hat{e}_{ii'}=0, e_{ii'}=1$
$P(\hat{e}_{ii'}\|c)$	$Q\left(\frac{\tau_0-R_0}{\sigma}\right)$	$Q\left(\frac{\frac{R_0R_s}{R_0+R_s}-\tau_0}{\sigma}\right)$	$Q\left(\frac{R_0-\tau_0}{\sigma}\right)$	$Q\left(\frac{\tau_0-\frac{R_0R_s}{R_0+R_s}}{\sigma}\right)$
Single Reference (Column) Scheme				
case	$\hat{e}_{ii'}=0, e_{ii'}=0$	$\hat{e}_{ii'}=1, e_{ii'}=1$	$\hat{e}_{ii'}=1, e_{ii'}=0$	$\hat{e}_{ii'}=0, e_{ii'}=1$
$P(\hat{e}_{ii'}\|c)$	$Q\left(\frac{\tau_0-R_0}{\sigma}\right)$	$Q\left(\frac{\frac{R_0R_s}{R_0+R_s}-\tau_0}{\sigma}\right)$	$Q\left(\frac{R_0-\tau_0}{\sigma}\right)$	$Q\left(\frac{\tau_0-\frac{R_0R_s}{R_0+R_s}}{\sigma}\right)$

Probability $P(\hat{c}|c)$ for the *Double Reference Scheme* is calculated as follows using the probabilities in Table 3.1:

$$P([\hat{e}_{ii'},\hat{e}_{j'j'}]|c)=P(\hat{e}_{ii'}|c)P(\hat{e}_{j'j'}|c). \tag{3.14}$$

The decision error probability $P(\hat{A}_{ij}\neq A_{ij}|c,\hat{c})$ is given by the following equation:

$$\begin{aligned}P(\hat{A}_{ij}\neq A_{ij}|c,\hat{c}) &= qQ\left(\frac{\tau(\hat{c})-R_1}{\sigma}\right)+(1-q)\Bigg[P(e_{ij}=1|c)\\ &\times Q\left(\frac{\frac{R_0R_s}{R_0+R_s}-\tau(\hat{c})}{\sigma}\right)+P(e_{ij}=0|c)Q\left(\frac{R_0-\tau(\hat{c})}{\sigma}\right)\Bigg],\end{aligned} \tag{3.15}$$

where q is the prior probability of 1 being stored. In the above equations, $Q(\cdot)$ is the Q-function, i.e., $Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^\infty \exp(-\frac{u^2}{2})du$. These probabilities are obtained using our modeling, the estimators introduced in Section 3.4, and simple estimation theory.

With the theoretical expression for the bit-error rate derived, we evaluate the results for different schemes. In the following evaluations, we use prior probability $q = 0.5$ and the following resistance values, which are the same as in [BHC17]: $R_1 = 100\Omega$, $R_0 = 1000\Omega$, $R_s = 250$ (the value of the parameter R_s will be changed for the upcoming Figure 3.4) and $q = 0.5$. We vary the parameters σ, n and p_f to test their influence on the performance of our proposed schemes.

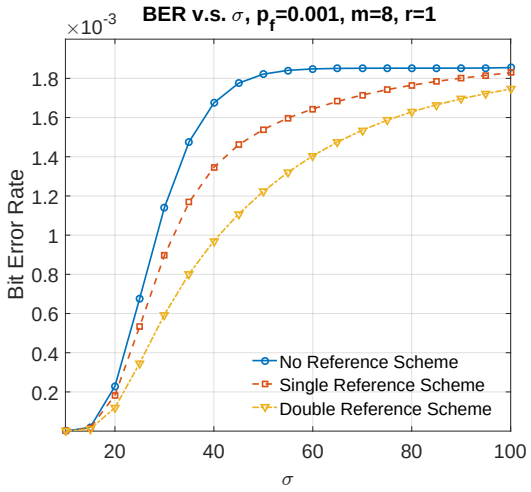


Figure 3.1: BER for a 1D1R structured square array with various noise magnitude.

We first evaluate results for a square array, i.e., $r = 1$. In this case, there is no need to differentiate the *Single Reference (Row) Scheme* and the *Single Reference (Column) Scheme*. We first set $m = 8$, $p_f = 0.001$, and vary σ from 10 to 100. The results are shown in Figure 3.1. We observe that both the *Single Reference Scheme* and the *Double Reference Scheme* offer noticeable improvements in terms of BER compared with the *No Reference Scheme*. Using the *Double Reference Scheme* provides better BER than using the *Single Reference Scheme*. We observe that under moderate noise (σ from 20 to 40), we can get consistently over 40% reduction in BER by using the *Double Reference Scheme*, and over 20% reduction in BER by using the *Single Reference Scheme*. In the high noise regime ($\sigma > 40$), all three schemes start to saturate and the improvement offered by our proposed schemes becomes smaller. However, by using our proposed schemes, the saturation of BER becomes slower.

For example, for a targeted BER of 1.6×10^{-3}, in terms of the noise standard deviation σ, 50% more noise can be tolerated by using the *Single Reference Scheme* and almost 100% more noise can be tolerated by using the *Double Reference Scheme*.

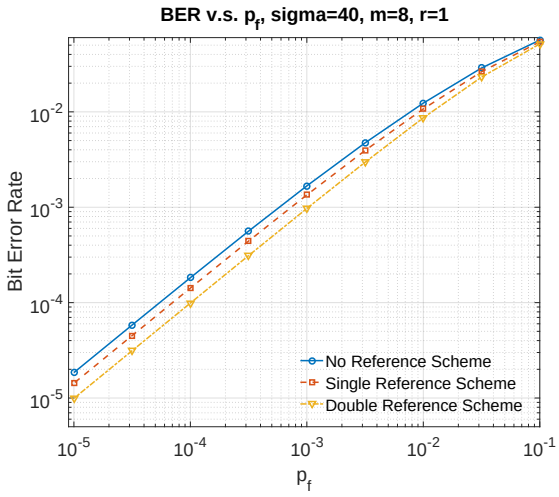


Figure 3.2: BER for a 1D1R structured square array with various p_f.

Next, we set $\sigma = 40$, $m = 8$, and vary p_f from 1×10^{-1} to 1×10^{-5}. The results are shown in Figure 3.2. We observe that as the diode becomes more reliable, i.e. as p_f becomes smaller, the improvement resulting from our proposed schemes increases. We conjecture that as the diodes become more reliable, the sneak-path events become more rare so that the information provided by the reference cells impacts the threshold more profoundly. For example, for $p_f = 1 \times 10^{-1}$, knowing $e_{ii'} = 1$ only changes $P(e_{ij} = 1|none) = 0.3017$ to $P(e_{ij} = 1|e_{ii'} = 1) = 0.5609$. In contrast, for $p_f = 1 \times 10^{-4}$, knowing $e_{ii'} = 1$ changes $P(e_{ij} = 1|none) = 0.0003749$ to $P(e_{ij} = 1|e_{ii'} = 1) = 0.4168$. The relative larger change of $P(e_{ij} = 1|c)$ in the latter case shifts the threshold more towards the left thus reducing the estimation error more effectively.

Next, we set $\sigma = 40$, $p_f = 0.001$ and vary m from 8 to 16 to test the performance of our proposed schemes in arrays with different sizes. The results are shown in Figure 3.3. We observe consistent improvement in using the proposed schemes in the range of tested

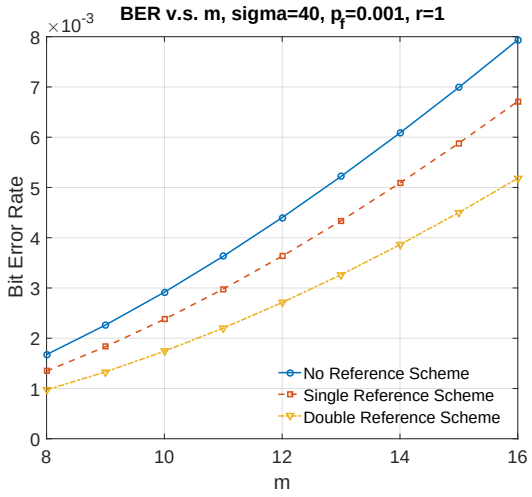


Figure 3.3: BER for a 1D1R structured square array with various array dimension.

array size. We also observe that as the array size gets larger, the relative improvement using our proposed scheme decreases by a small amount, akin to the reasoning for decrease in performance improvement in the case of increased p_f. For a large array, one can divide the large array into small sub-arrays to keep the relative larger improvement. However, this approach induces a larger circuit overhead and a larger storage overhead, resulting in the familiar trade-off between reliability and overhead observed in e.g., channel coding. Also, for a large array, i.e., large m, the probabilities in Appendix B are hard to compute due to the combinatorial nature of these calculations. This difficulty can potentially limit the usage of the proposed schemes but it can be mitigated in two ways. First, these probabilities are only computed once for an array of a given size so these probabilities along with the induced thresholds can be precomputed at the design stage. Second, if the exact probabilities are prohibitively difficult to compute, one can use the approximations mentioned at the end of the Appendix B to compute the approximated probabilities and then the induced suboptimal thresholds. This simplified approach still maintains most of the relative gain of the adaptive thresholding schemes while reducing the burden of computing those probabilities exactly.

We also test the sensitivity of our proposed schemes under different values of the sneak-

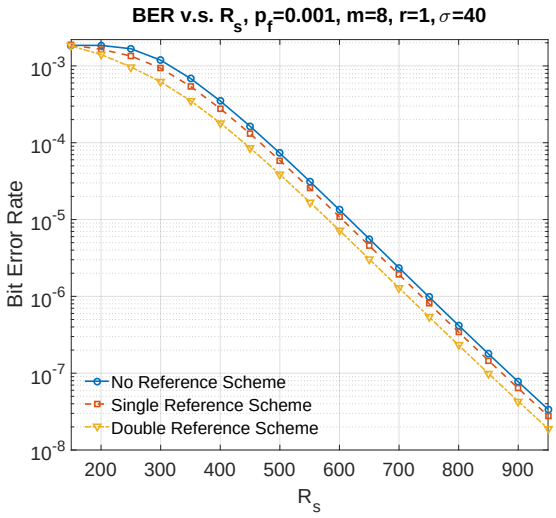


Figure 3.4: BER for a 1D1R structured square array with various sneak-path resistance.

path resistance R_s, which is governed by cell nonlinearity [JYZP+12]. We set $\sigma = 40$, $p_f = 0.001$ and vary R_s from 150 to 950. As shown in Figure 3.4, the improvement using our proposed schemes is consistent throughout this range, except when R_s is very small, i.e., in the case when $\frac{R_0 R_s}{R_0+R_s}$ is very close to R_1. When $\frac{R_0 R_s}{R_0+R_s}$ is very close to R_1, knowledge of the sneak-path state(s) of reference cell(s) does not change threshold significantly. As a result, the adaptive thresholding schemes are ineffective. From this result, we can also infer the performance of our proposed scheme for other values of the R_0, R_1 parameters, as the Q-function in the BER calculation is invariant to the same scaling of R_0, R_1, R_s, and σ.

Next, we turn our attention to the performance of the proposed schemes for an array with non-unity aspect ratio, i.e., for $r \neq 1$. We first set $m = 8$, $r = 2$, $p_f = 0.001$ and vary σ from 10 to 100. The results are shown in Figure 3.5. From the results, we note that the *Single Reference (Row) Scheme* outperforms *Single Reference (Column) Scheme*. This observation suggests that the reference cell in the row of an information cell provides more information than the reference cell in the column of this information cell. Intuitively, with more available information cells in the row than in the column, an information cell is more likely to incur

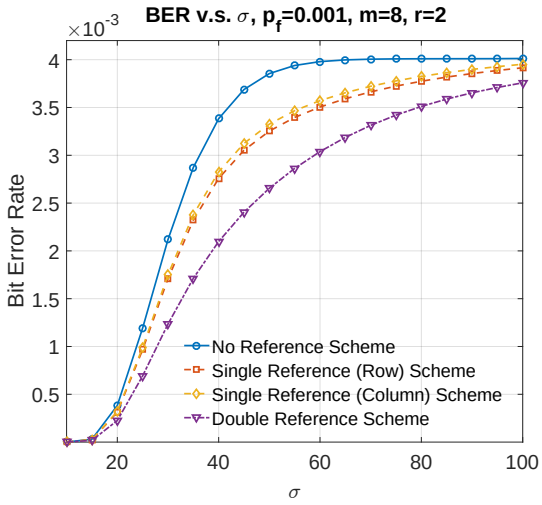
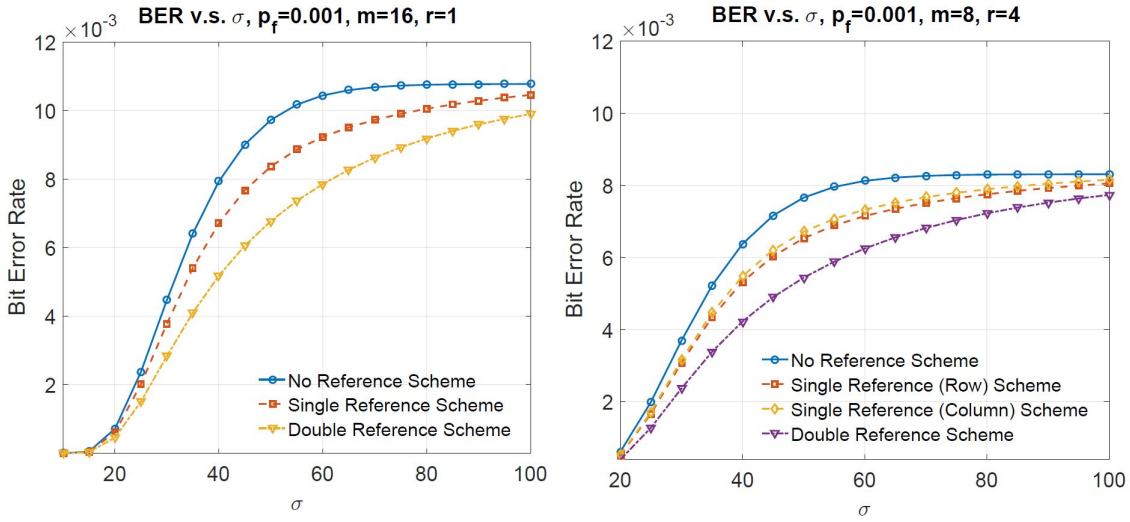


Figure 3.5: BER for a 1D1R structured 8×16 array with various noise magnitude.

a sneak-path event given that we know a sneak-path event occurs at the reference cell in its row than given that we know a sneak-path event occurs at the reference cell in its column. For example, for the parameters in Figure 3.5, we have $P(e_{ij} = 0, e_{ii'} = 0) = 0.9959$ and $P(e_{ij} = 0, e_{j'j} = 0) = 0.9955$.

Next, we present the results for two arrays with the same total number of cells, one array being a square array and the other array being a non-square array. We set $\sigma = 40$, $p_f = 0.001$. The square array is 16×16 with 240 information cells, and the non-square array is 8×32 with 224 information cells. The results are reported in Figure 3.6. We observe that although the non-square array has fewer information cells relative to the square array, the non-square array is less susceptible to bit-errors relative to the square array under the same amount of noise and the same thresholding scheme. Intuitively, this can be partially explained by the observation that a decrease in one dimension (here from 16 to 8), which decreases the probability of a sneak-path event, has a stronger effect on the occurrence of sneak-path event than the same simultaneous multiplicative increase in the other dimension (here from 16 to 32), which in contrast increases the probability of a sneak-path event. For

example, the extreme case of the array of size 1×256 is indeed free of sneak-path events.



(a) 16×16 array.

(b) 8×32 array.

Figure 3.6: BER for two 1D1R structured arrays with same number of cells.

Finally, we report the performance for an array with a 1S1R structure. We set $m = 8$, $p_f = 0.1$, and vary σ from 10 to 100. We choose $p_f = 0.1$ because the 1S1R structure is less susceptible to the sneak-path event compared to the 1D1R structure. 1S1R structure with $p_f = 0.1$ gives $P(e_{ij} = 0)$, comparable to $P(e_{ij} = 0)$ for 1D1R structure with $p_f = 0.001$. The result is reported in Figure 3.7.

We observe that although the *Single Reference Scheme* and the *Double Reference Scheme* still offer improvements in terms of BER, the improvements are incremental. This observations suggests that, in the 1S1R structured array, with the stronger isolation provided by the cell selectors, the dependency of the sneak-path event between the reference cells and the information cell is weaker than this dependency is in the 1D1R structured array. For example, in a 1S1R structured array, with $m = 8$, $r = 1$ and $p_f = 10^{-0.5}$, we get $P(e_{ij} = 0) = 0.8896$, $P(e_{ij} = 0|e_{ii'} = 0) = 0.8961$, $P(e_{ij} = 0|e_{ii'} = 0, e_{j'j} = 0) = 0.8325$, and $P(e_{ij} = 0|e_{ii'} = 1, e_{j'j} = 1) = 0.3899$; in a 1D1R structured array, with $m = 8$, $r = 1$ and $p_f = 10^{-1.5}$, we get $P(e_{ij} = 0) = 0.8961$, $P(e_{ij} = 0|e_{ii'} = 0) = 0.9433$,

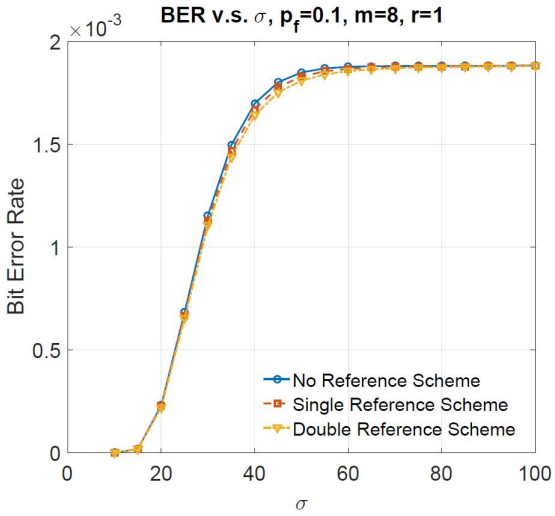


Figure 3.7: BER for a 1S1R structured 8×8 array with various noise magnitude.

$P(e_{ij} = 0|e_{ii'} = 0, e_{j'j} = 0) = 0.9709$, and $P(e_{ij} = 0|e_{ii'} = 1, e_{j'j} = 1) = 0.1208$. It is clear from this example that with comparable $P(e_{ij} = 0)$, in an array with the 1S1R structure, the reference cells provide less side information than the amount of side information provided by reference cells in an array with the 1D1R structure.

3.6 A Comparison with the BCH $(239, 255)$ Code

The adaptive thresholding approach proposed in this chapter is not a direct substitute to the channel coding approaches. First, the adaptive thresholding approach requires extra read operations and multiple threshold detectors while the channel coding approaches require a channel encoder/decoder. Second, this adaptive thresholding approach, which is estimation theoretic, reduces the BER for a certain noise magnitude while some channel coding techniques have guaranteed error-correction capability.

A comparison between the adaptive thresholding approach and the channel coding approaches is nonetheless valuable, as it can help in better understanding the property of this special problem (the re-occurrence of sneak-path due to selection device failure) and can

provide insight into the memory design. We therefore present the following case study.

We consider 1D1R structured 16×16 arrays. For a 16×16 array, 16 cells are used as pilot cells when our proposed pilot construction is used. We compare our adaptive thresholding schemes using the pilot construction with the standard BCH $(239, 255)$ code, which corrects up to 2 bit-errors also also uses 16 bits of redundancy. To make the BCH code compatible with the array architecture, we assume that the BCH $(239, 255)$ code can correct up to 2 bit-errors among 256 (instead of 255) bits. In our simulations, we set $p_f = 0.001, R_0 = 1000, R_1 = 100, R_s = 250, q = 0.5$ and vary σ. Note that for the BCH coded array without the pilot construction, we also use the optimal threshold, which is calculated similar to $\tau(none)$, to determine the state of a cell.

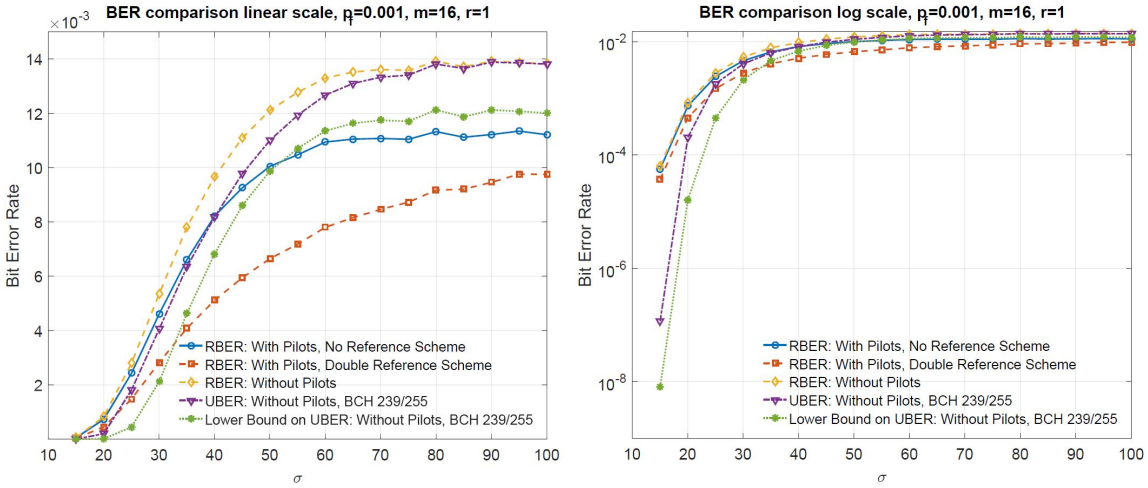


Figure 3.8: A Comparison with the BCH $(239, 255)$ Code.

In Figure 3.8, under both linear and log scale, we report our simulation results of the raw bit-error rate (RBER) for an array with pilot cells, with the No Reference Scheme and the Double Reference Scheme. We also report the RBER and undetectable bit-error rate (UBER) for an array that is coded with the BCH $(239, 255)$ code. To gain further insight on how the inter-cell dependency of sneak-path events affects the coding performance, we also report a lower bound of the UBER for BCH $(239, 255)$ coded array assuming bit-errors are independent.

It is first worth noting that by simply using the pilot construction without adaptively changing the threshold, i.e., using the No Reference Scheme, we get better RBER performance comparing to the RBER of the array without the pilot construction. This can be explained as follows: by using the pilot construction, we effectively reduce the fraction of cells with LRS. The pilot construction can be considered as a simple shaping code in this case. As a result, the probability of a sneak-path event is also reduced, as noted in [CKY16]. The reduced sneak-path probability therefore reduces the RBER. Second, by comparing the UBER of the BCH $(239, 255)$ code with its lower bound, which is obtained using the RBER in the BCH coded array under the independence assumption, we note that the bit-errors are highly dependent. In the rare case that a diode fails, all cells are more prone to error due to the adverse effect of the sneak-path. Most of bit-errors are concentrated in this case and are therefore dependent.

Next, we compare the performance of the coding approach and the adaptive thresholding approach in two representative noise regimes. In the low noise regime ($\sigma < 25$), even if the sneak-path event occurs, the number of bit-errors is usually less than 3, i.e., the number is within the error-correction capability of the BCH $(239, 255)$ code. We observe that in this case, the BCH $(239, 255)$ code reduces BER very effectively. The implication for the memory design is that if the additive noise is small, e.g., there is a large read margin, the sneak-path problem is not severe and it can be handled with an appropriate ECC. In the high noise regime we studied, when a diode fails, the number of bit-errors is often beyond the error-correction capability of the BCH $(239, 255)$ code because the adverse effect of the sneak-path event dominates in this setting. In this case, we observe a much smaller coding gain compared to the improvement achieved by using the Double Reference Scheme with pilot cells. Note that in this high noise regime, while this simple error-correction code is ineffective, the Double Reference Scheme only reduces the number of errors but does not eliminate them entirely, in most cases. This observation opens up the possibility of combining our proposed adaptive thresholding schemes with more sophisticated error-correction code

over large (multiple) array(s). In this sense, the adaptive thresholding schemes can be viewed as a preprocessing set that provides a lower RBER for the subsequent ECC solution.

CHAPTER 4

Channel Models and Coding Solutions for 1S1R Crossbar Resistive Memory with High Line Resistance

4.1 Introduction

The crossbar resistive memory, whereby bistable memristors are placed at the crosspoint of wordlines and bitlines, is one promising candidate for the next generation nonvolatile memory due to its inherent $4F^2$ device density and its simple crossbar structure [IW15]. One of its most appealing applications is storage-class memory (SCM), a term that refers to memory technology that fills the latency and density gaps between DRAM and NAND flash memory [BKS+08]. Meanwhile, as semiconductor technology scales down to single-digit-nm, simultaneously scaled wordline/bitline resistances increasingly become a limiting factor to device reliability and hence memory scalability [LYWW13].

Previous literature has extensively shown that even moderate line resistance significantly degrades the reliability of the write and read operations. The degradation of the write/read margins due to high line resistance for the wort-case memory cell, i.e., the cell that is furthest from the source and ground, are studied in [LYWW13, CLY16, KKC15]. The adverse effect of the line resistance on the write/read margins for cells across the memory array are studied in [Che13, SKK10] by solving a system of Kirchhoff's current law (KCL) equations. While these studies focused on the degradation of the write/read margins, it remains unclear how the degraded write/read margins affect the system level reliability metric, e.g., the bit-error rate (BER). In other words, channel models are not yet well-established for this problem.

It is demonstrated in [Che13] that, when considering the line resistance in resistive memory, the write margins are non-uniform across the array, which leads to non-uniform reliability levels in the memory array. Designing error correction codes (ECCs) for the worst-case often leads to overly conservative code design and is therefore not rate efficient. For example, in [ZFW+19], the authors designed a non-stationary polar code targeting channels with different reliability levels, which are characterized empirically by simulations. Moreover, [ZFW+19] also showed that using more precise channel modeling, i.e., using the binary asymmetric channel (BAC) instead of the binary symmetric channel (BSC), provides an order of magnitude improvement in BER, which proves the necessity of precise channel models.

In this work, we propose BAC models for writing to and reading from memory devices in crossbar memory, parameterized by device parameters, array size, wordline/bitline resistances, and device location by statistically relating the degraded write/read margins of cells at different locations to the channel parameters. Our analytical channel models, which take into account the device location, provide quantitative tools for analyzing the aforementioned non-uniformity and the trade-off between device parameters and memory reliability. These models are therefore beneficial for system engineers when designing the next generation storage systems. Previous studies on the write/read margins assume deterministic High Resistance State (HRS) and Low Resistance State (LRS) for the memory device whereas the HRS and LRS are nondeterministic in nature [JLY+15, CL11]. Our write/read channel models, which are derived probabilistically, allow us to take the resistance variability of LRS and HRS into consideration for more precise modeling.

Building upon our proposed channel models and the observation of non-uniform reliability, we propose methods to reduce the raw bit-error rate (RBER) and undetected bit-error rate (UBER) using techniques from estimation theory and channel coding theory. For the read channel, we propose an efficient procedure to compute an optimal read threshold. We show that the RBER of the read channel is reduced by a large factor using the optimal read threshold. Efficient ECC solution for a crossbar resistive memory targeting the SCM

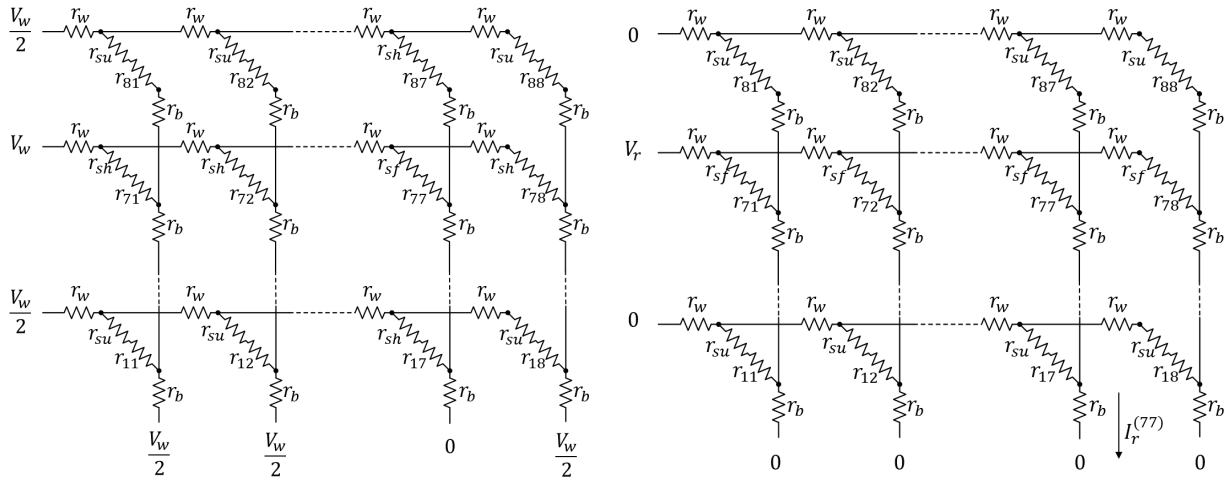
application must mitigate and/or leverage the non-uniformity of reliability while being compatible with the low-latency requirement of SCM. Based on BCH codes, we propose a scheme based on interleaving and a scheme utilizing multiple BCH codes with different error correction capabilities for improved UBER performance. For the latter scheme, we propose a systematic framework for allocating different codes based on the location dependent channel parameters. Both of our proposed coding schemes effectively reduce the UBER.

The content of Chapter 4 is organized as follows. Section 4.2 provides background on crossbar resistive memory and the write/read operation. The circuit models and the variabilities are also discussed in Section 4.2. Section 4.3 presents the channel characterization for the write and read operations. Simulation results on the proposed channel models with various parameters are also presented in Section 4.3. Section 4.4 presents our study on the optimal read threshold and its simulation results. Section 4.5 discusses two schemes for efficient channel coding targeting SCM applications.

4.2 Preliminaries

4.2.1 1S1R Crossbar Resistive Memory Background and Model

In crossbar resistive memory array, the logical state 0 or 1 is represented by the HRS or LRS of a memory cell, respectively. For a bipolar memristor, the state of a cell is switched from LRS to HRS (Reset Operation) or from HRS to LRS (Set Operation) by applying a positive or negative voltage across the memory cell, respectively. For the write operation, we consider the so called "V/2" write scheme (cf. [CLY16]) as it is usually more energy-efficient than the so called "V/3" write scheme. In particular, when writing to a selected cell, the wordline and bitline of the selected cell are biased at the write voltage (V_{w_set} or V_{w_reset}) and 0, respectively, while other wordlines and bitlines are biased at half of the write voltage to prevent unintentional write, as shown in Fig. 4.1a.



(a) Circuit model for writing to a 8×8 array. (b) Circuit model for reading to a 8×8 array.

Figure 4.1: Examples of circuit models (V_w denotes V_{w_set} or V_{w_reset}).

For the read operation, we consider the current-mode sensing scheme as it has a smaller latency compared with the voltage-mode sensing scheme [CLY16]. When reading a selected cell, a read voltage (V_r) is applied on its wordline and all other wordlines and bitlines are grounded. A current is sensed by the sensing amplifier located at the end of its bitline, and is used to determine the state of the selected cell, as shown in Fig. 4.1b.

In this chapter, we focus on crossbar resistive memory with the widely used 1 selector 1 resistor (1S1R) structure, where highly nonlinear selectors are connected in series with the memristors to prevent write and read disturbs. For both write and read operations, when the voltage across a selector is close to the applied voltage, we say that this selector is fully selected and we assume it has resistance r_{sf}; when the voltage across a selector is close to 0, we say that this selector is un-selected and we assume it has resistance r_{su}. For the write operation, since other cells on the wordline and bitline of the selected cell have voltage close to half of the write voltage across them, we say that the selectors for those cells are half-selected and we assume they have resistance r_{sh}. In general, $r_{sf} << r_{sh} < r_{rs}$. An ideal selector has parameters $r_{sf} = 0$ and $r_{sh} = r_{su} = \infty$. Our proposed model is a general one that does not have the ideal selector assumption. Meanwhile, since the main

focus of this work is the adverse effect of line resistance, we use the ideal selector assumption to provide mathematical insights in III.B and to simplify our simulations in III.D and IV.C. Throughout this work, we assume that the interconnect resistances of wordlines and bitlines are constant across the array, and they are denoted by r_w and r_b respectively.

4.2.2 Memristor Variabilities and Models

In this chapter, we consider two variabilities of memristor, the non-deterministic write operation and the non-deterministic resistance value for each resistance state. It is widely observed that the switching operations of memristor are stochastic and follow log-normal switching time distributions, with distribution parameters depend on the applied voltage [MRPC$^+$11, NXX12]. Our models for the switching time distributions are adopted from [MRPC$^+$11] and more details are provided in Section 4.3.

Previous works (cf. [LYWW13] - [SKK10]) on the degradation of write and read margins due to high line resistance assume deterministic resistance states, e.g., HRS resistance is 10000Ω and LRS resistance is 100Ω. Meanwhile, due to both device-to-device variation and cycle-to-cycle variation, the resistance of each state is highly non-deterministic [JLY$^+$15, CL11]. To incorporate this variability into our reliability analysis, we use random variables to represent the resistance of the memory cells. Based on observations in [JLY$^+$15,CL11], we assume they are i.i.d. and their conditional distributions, conditioned on their states, follow log-normal distributions. For example, let i.i.d. Bernoulli(q) random variable S_{ij} denote the state of cell (i, j), with $S_{ij} = 1$ for LRS and $S_{ij} = 0$ for HRS. Let R_{ij} be the associated random variable denoting the resistance of cell (i, j). Then our model assumes:

$$\ln(R_{ij}|S_{ij} = 1) \sim \mathcal{N}(\mu_L, \sigma_L^2),$$

and

$$\ln(R_{ij}|S_{ij} = 0) \sim \mathcal{N}(\mu_H, \sigma_H^2).$$

4.3 Channel Models

Our proposed channel models are depicted in Fig. 4.2. We model the write, read and cascaded channels as binary asymmetric channels (BACs). We discuss in the following subsections how the channel parameters are related to device parameters, line resistance, and device location.

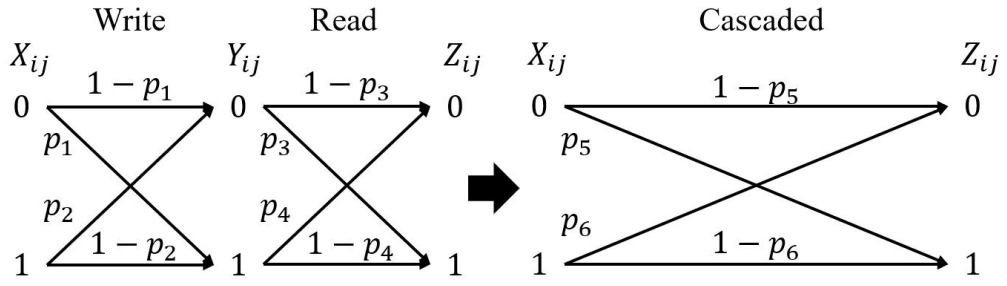


Figure 4.2: Proposed channel models.

We denote the state we want to write to cell (i, j) by $X_{ij} \in \{0, 1\}$, the state actually written by $Y_{ij} \in \{0, 1\}$ and the detected (for read operation) state by $Z_{ij} \in \{0, 1\}$. Note that even though the information is stored as the resistance of a cell, we choose to use binarized state variable Y_{ij} because firstly it enables us to utilize a well-known result in the literature [MRPC+11] that characterizes the switching of a device, and secondly it allows mathematical tractability and the separation of the write/read channels. Information about the resistance of a cell is instead embedded in the resistance distribution. Also note that with a binarized state for a cell, multiple write/read operations are also independent. Therefore, the cascaded channel model is still valid if one writes to and reads from a cell multiple times.

4.3.1 Write Channel

In this subsection, we derive the write channel. We note that the write operation is affected by the previous state of cell (i, j). We let this state be denoted by S^*_{ij} and the associated resistance value be R^*_{ij}. We assume that when the previous state is the same as the state we

want to write, the write operation is always successful, i.e., $P(Y_{ij} = 1|X_{ij} = 1, S^*_{ij} = 1) = 1$, and $P(Y_{ij} = 0|X_{ij} = 0, S^*_{ij} = 0) = 1$.

When the previous state is not the same as the state we want to write, a sufficient write voltage and a sufficient write time are required to change the state of the cell. Due to high line resistances, the effective write voltage on a cell could be much smaller than the desired write voltage, i.e., the write margin is decreased. We denote the effective write voltage on a cell (i, j) as $\tilde{V}_w(r^*_{ij}, i, j)$ where r^*_{ij} is a realization of R^*_{ij}. With a method similar to the one described in [Che13], $\tilde{V}_w(r^*_{ij}, i, j)$ can be obtained by solving a system of KCL equations using the circuit model described in subsection 4.2.1. We map the degraded write margin to the decreased write reliability by considering the log-normal switching time distribution, adopted from [MRPC+11]. With fixed switching times t_{set} and t_{reset}, the log-normal switching time distributions lead to the following:

$$P(Y_{ij} = 1|X_{ij} = 1, S^*_{ij} = 0, R^*_{ij} = r^*_{ij}) = 1 - Q\left(\frac{\ln t_{set} - \ln(\tau^{(ij)}_{set})}{\sigma_{set}}\right), \tag{4.1}$$

and

$$P(Y_{ij} = 0|X_{ij} = 0, S^*_{ij} = 1, R^*_{ij} = r^*_{ij}) = 1 - Q\left(\frac{\ln t_{reset} - \ln(\tau^{(ij)}_{reset})}{\sigma_{reset}}\right), \tag{4.2}$$

where $Q(\cdot)$ is the Q-function, i.e., $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{u^2}{2})du$. Parameters σ^2_{set} and σ^2_{reset} are the variance of the normal distributions associated with the set and reset switching time distribution, respectively, which are independent of $\tilde{V}_w(r^*_{ij}, i, j)$, according to [MRPC+11]. Parameters $\tau^{(ij)}_{set}$ and $\tau^{(ij)}_{reset}$ are the median of the set and reset switching time, respectively. Note that in the above equations, to be consistent with the existing literature [MRPC+11, NXX12], we use the median parameterization of the log-normal distribution. According to the literature, the medians of the switching times ($\tau^{(ij)}_{set}$ and $\tau^{(ij)}_{reset}$ in μs) are exponentially dependent on the effective write voltage. We therefore parameterize the medians as following:

$$\ln\left(\tau^{(ij)}_{set}\right) = \alpha_{set}\tilde{V}_w(r^*_{ij}, i, j) + \beta_{set},$$

and

$$\ln\left(\tau^{(ij)}_{reset}\right) = \alpha_{reset}\tilde{V}_w(r^*_{ij}, i, j) + \beta_{reset}.$$

Using (4.1), (4.2) and marginalizing over the conditionally log-normally distributed random variable R^*_{ij}, we get:

$$P(Y_{ij}=0|X_{ij}=1,S^*_{ij}=0)=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}r^*_{ij}\sigma_H}\exp\left[-\frac{\left(\ln r^*_{ij}-\mu_H\right)^2}{2\sigma_H^2}\right]Q\left(\frac{\ln t_{set}-\ln(\tau^{(ij)}_{set})}{\sigma_{set}}\right)dr^*_{ij}, \tag{4.3}$$

and

$$P(Y_{ij}=1|X_{ij}=0,S^*_{ij}=1)=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}r^*_{ij}\sigma_L}\exp\left[-\frac{\left(\ln r^*_{ij}-\mu_L\right)^2}{2\sigma_L^2}\right]Q\left(\frac{\ln t_{reset}-\ln(\tau^{(ij)}_{reset})}{\sigma_{reset}}\right)dr^*_{ij}. \tag{4.4}$$

Putting (4.3) and (4.4) together with the prior symbol probability $q=P(S^*_{ij}=0)$, we arrive at the write binary asymmetric channel, depicted in Fig. 4.2, for the write operation with the following channel parameters:

$$p_1^{(ij)}=(1-q)P(Y_{ij}=1|X_{ij}=0,S^*_{ij}=1), \tag{4.5}$$

and

$$p_2^{(ij)}=qP(Y_{ij}=0|X_{ij}=1,S^*_{ij}=0). \tag{4.6}$$

Here and elsewhere, we use superscript (ij) to highlight that the channel parameters are dependent on the cell location (i,j).

Through equations (4.1) - (4.6), we are able to relate the write margin $\tilde{V}_w(r^*_{ij},i,j)$ to the BER of the write channel. For example, comparing the best-case cell to the worst-case cell in the example in subsection 4.3.4 Fig. 4.3a, we observe that the write margin for Reset is dropped from $4.9V$ to $1.64V$ while the write BER is increased from 3.35×10^{-4} to 1.75×10^{-2}, thus providing further evidence that location dependent BER analysis matters.

4.3.2 Read Channel

When reading from the cell $(i.j)$, we consider the current-mode sensing scheme and a fixed threshold detector. Let $I_r^{(ij)}$ be the current sensed by the sensing amplifier, which can be

also calculated by solving a system of KCL equations. $I_r^{(ij)}$ is hence dependent on the cell location, the resistance of the selected cell, and the resistances of unselected cells. Let I_{th} be the threshold current. The threshold detector is as follows:

$$Z_{ij} = \begin{cases} 0, I_r^{(ij)} \leq I_{th}, \\ 1, I_r^{(ij)} > I_{th}. \end{cases} \tag{4.7}$$

With the threshold detector above, the decision error probabilities are:

$$P(Z_{ij} = 1|Y_{ij} = 0) = P(I_r^{(ij)} > I_{th}|Y_{ij} = 0); \tag{4.8}$$

$$P(Z_{ij} = 0|Y_{ij} = 1) = P(I_r^{(ij)} \leq I_{th}|Y_{ij} = 1). \tag{4.9}$$

This set-up leads to the read binary asymmetric channel, depicted in Fig. 4.2, for the read operation with $p_3^{(ij)} = P(Z_{ij} = 1|Y_{ij} = 0)$ and $p_4^{(ij)} = P(Z_{ij} = 0|Y_{ij} = 1)$.

4.3.2.1 Closed form Expression with Ideal Selectors

Since we need to solve a system of equations to get $I_r^{(ij)}$, equations (4.8) and (4.9) are not sufficient as they do not give closed-form expressions for the channel parameters. However, if we consider ideal selectors, closed-form expressions can be derived. Note that for the analysis with a non-ideal selector, one can still use the general characterizations in equations (4.8) and (4.9) and find $I_r^{(ij)}$ by solving a system of KCL equations. Moreover, it is reasonable to assume ideal characteristics of the selector for the analysis of the line resistance in the 1S1R structure as near ideal selector properties are demonstrated by industry in [JKN+15].

With ideal selectors, the part of the circuit connected to the un-selected cells can be neglected, resulting in a simplified circuit with just the selected cell and its wordline/bitline. With this simplified circuit, $I_r^{(ij)}$ is a function of the random variable R_{ij}, which represents the resistance of the selected cell. We therefore have:

$$I_r^{(ij)} = \frac{V_r}{ir_b + jr_w + R_{ij}}. \tag{4.10}$$

Plugging (4.10) into (4.8) and (4.9), and using the assumption that R_{ij} is conditionally (on Y_{ij}) log-normally distributed, we obtain the following closed form expression for p_3 and p_4:

$$\begin{aligned} p_3^{(ij)} &= P\left(\frac{V_r}{ir_b + jr_w + R_{ij}} > I_{th} | Y_{ij} = 0\right) = P\left(R_{ij} < \frac{V_r}{I_{th}} - ir_b - jr_w | Y_{ij} = 0\right) \\ &= Q\left(\frac{\mu_H - \ln\left(\frac{V_r}{I_{th}} - ir_b - jr_w\right)}{\sigma_H}\right), \end{aligned} \tag{4.11}$$

and similarly

$$p_4^{(ij)} = Q\left(\frac{\ln\left(\frac{V_r}{I_{th}} - ir_b - jr_w\right) - \mu_L}{\sigma_L}\right). \tag{4.12}$$

Define $R_{th} = \frac{V_r}{I_{th}}$. From equations (4.11) and (4.12), we observe that R_{th} is the effective decision threshold between the HRS and LRS distribution in the resistance domain, when there is no line resistance, i.e., $r_w = r_b = 0$. We can therefore interpret the adverse effect of line resistances during the read operation as follows: the effective read threshold in resistance domain is shifted to the left by the total accumulated line resistance. This shift results in a higher bit-error rate if R_{th} is set to be the optimal decision threshold without considering the line resistance.

The read margin is defined by the difference between the sensed current of a HRS cell and the sensed current of a LRS cell. Using equations (4.10) - (4.12), we can now relate the read margin to the read BER. For example, comparing the best-case cell to the worst-case cell in the example in subsection 4.3.4 Fig. 4.3a, we observe that the read margin is dropped from $296\mu A$ to $95\mu A$ while the write BER is increased from 4.29×10^{-4} to 7.33×10^{-2}, again demonstrating the need of a location dependent model.

4.3.3 Cascaded Channel and Channel Capacity

Combining the results of the previous two subsections, we get a cascaded channel for a single memory cell. The cascaded channel is a binary asymmetric channel and it is depicted in Fig. 4.2, with $p_5^{(ij)} = p_1^{(ij)}(1 - p_4^{(ij)}) + (1 - p_1^{(ij)})p_3^{(ij)}$ and $p_6^{(ij)} = p_2^{(ij)}(1 - p_3^{(ij)}) + (1 - p_2^{(ij)})p_4^{(ij)}$.

The capacity of this cascaded channel for cell (i, j) is as follows:

$$\begin{aligned} C_{ij} &= \max_{q} I(X_{ij}; Z_{ij}) \\ &= \max_{q} \left[h\left(q\left(1 - p_5^{(ij)}\right) + (1-q)p_6^{(ij)}\right) - qh\left(p_5^{(ij)}\right) - (1-q)h\left(p_6^{(ij)}\right) \right], \end{aligned} \tag{4.13}$$

where $h(\cdot)$ is the binary entropy function. Because $p_1^{(ij)}$ and $p_2^{(ij)}$ are dependent on q, the closed form capacity result for a standard BAC does not hold. The channel capacity therefore need to be evaluated with a numerical method, e.g., the Blahut-Arimoto algorithm, as further presented in subsection 4.3.4.

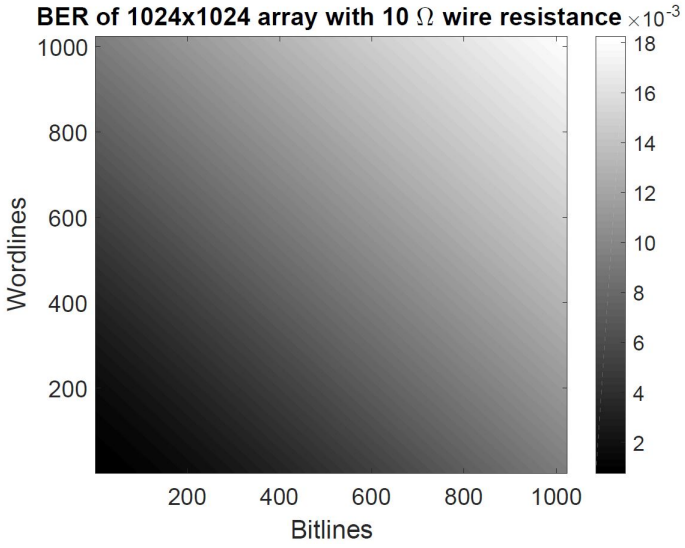
4.3.4 Simulations Results

Based on our models presented in the previous subsections, we simulate multiple arrays to explore how memory parameters affect the memory reliability metrics, such as the bit-error rate (BER) and the averaged capacity. We calculate the averaged capacity by averaging the capacities of cells given by equation (4.13); this result serves as an indicator of what fraction of the input data can be reliably stored in memory. Since this work is mainly focused on the adverse effect of the line resistance, we only vary the array size, aspect ratio, and line resistance in our simulations. Other memory parameters are kept the same and are summarized in Table 4.1. As an illustrative example, the parameters are chosen to represent a moderate reliability level, with a BER on the order of 10^{-3} in the best case scenario. The considered line resistances range from 10Ω to 100Ω, in accordance with the interconnect resistance values of interest for moderate technology nodes [LYWW13]. The chosen standard deviation (0.3) of LRS and HRS distribution is experimentally observed in [JLY+15]. The chosen switching parameters are based on [MRPC+11]; we use the same parameters for reset and set operations for simplified analysis.

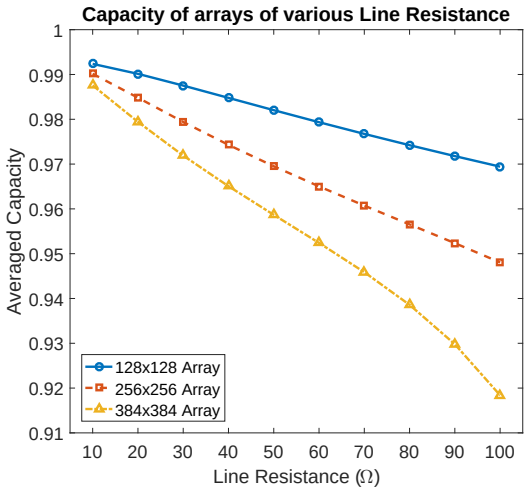
In Fig. 4.3a, we first present the BER of each cell in a 1024×1024 array to illustrate the spatial variation of reliability due to the line resistance. According to [LYWW13], the chosen 10Ω line resistance corresponds to the resistance per junction of Cu wire with 20nm

Symbol	Parameters	Values
m, n	Array Size ($m \times n$)	varies
V_{w_set}	Set voltage	-5V
V_{w_reset}	Reset voltage	5V
V_r	Read voltage	3V
q	Prior symbol probability of 0	0.5
r_w	Wordline interconnect resistance	$10\Omega - 100\Omega$
r_b	Bitline interconnect resistance	$10\Omega - 100\Omega$
r_{sf}	Fully selected selector resistance	0
r_{sh}	Half selected selector resistance	∞
r_{su}	Unselected selector resistance	∞
μ_L	Associated mean of LRS distribution	$4\ln(10)$
μ_H	Associated mean of HRS distribution	$6\ln(10)$
σ_L	Associated std of LRS distribution	$0.3\ln(10)$
σ_H	Associated std of HRS distribution	$0.3\ln(10)$
α_{set}	Parameter for the median set time	0.25
β_{set}	Parameter for the median set time	4.25
α_{reset}	Parameter for the median reset time	-0.25
β_{reset}	Parameter for the median reset time	4.25
σ_{set}	Associated std of set time distribution	0.5
σ_{reset}	Associated std of reset time distribution	0.5
t_{set}	Switching time for set operation	$100\mu s$
t_{reset}	Switching time for reset operation	$100\mu s$
I_{th}	Read decision threshold	$30\mu A$

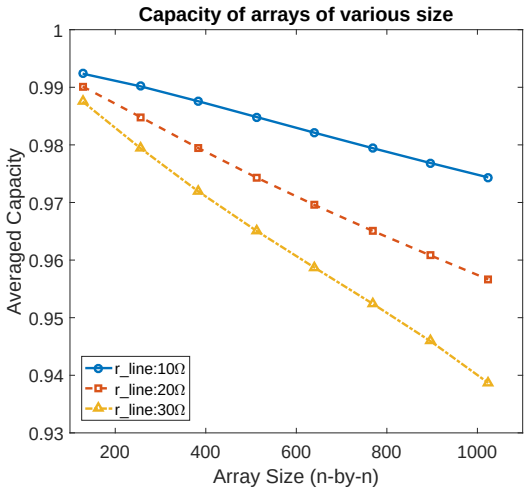
Table 4.1: Summary of parameters.



(a) Heatmap of BERs for a 1024x1024 array.



(b) Capacity results of various line resistances.



(c) Capacity results of various array sizes.

Figure 4.3: Simulation results for the proposed channel models.

technology nodes. With this moderate line resistance, we observe an order of magnitude BER difference between the best-case cell, located closest to the voltage source, and the worst-case cell, located furthest from the voltage source. Due to line resistance, the cell which is further from the source and sensing amplifier, suffers from a lower voltage delivery during the write operation and a higher resistance interference during the read operation,

thus has a larger BER.

Note that as the main focus of this work is the line resistance, we do not consider the line capacitance in an array. To take line capacitance into consideration, which mainly affects the switching time for set or reset operation when cell reliability is concerned, one can replace the t_{set} and t_{reset} in Table 4.1 with functions that depend on the location (i, j), the line resistance and the line capacitance. Qualitatively, cells that are further away from the source and the ground will get shorter effective switching time and therefore line capacitance exaggerates the non-uniformity that is observed in 4.3a.

Next, in Fig. 4.3b and Fig. 4.3c, we present the averaged capacity per cell for arrays with various size and line resistances, with aspect ratio fixed to be 1. We observe that a larger line resistance, which corresponds to a smaller technology node, deteriorates the averaged capacity almost linearly. This trade-off thus must be taken into consideration when scaling the memory, as it is shown in [LYWW13] that the line resistance scales exponentially with respect to the technology node. Also note that when the accumulated line resistance of the worst-case cell gets close to the effective resistance threshold, i.e., when $nr_w + mr_b$ is close to R_{th}, the averaged capacity deteriorates faster, as from (4.12), when $nr_w + mr_b > R_{th}$, reading from a LRS cell correctly is impossible. This explains the rapid dropping at the end of the curve in Fig. 4.3b for the 384×384 array. From Fig. 4.3c, we notice that the averaged capacity also deteriorates almost linearly with respect to the array size. This effect is thus a limiting factor for the realization of a large memory array.

Array Size	128×128	64×512	32×512	16×1024	8×2048	4×4096
Averaged Capacity	0.9924	0.9918	0.9897	0.9845	0.9745	0.9573

Table 4.2: Capacity of arrays with different aspect ratios.

We further investigate how the aspect ratio affects the averaged capacity by simulating arrays with the same number of cells but different aspect ratios. In Table 4.2, with a total of 16384 cells, the square array (aspect ratio = 1) has the largest averaged capacity and

the 4×4096 array, which has the largest aspect ratio, has the lowest averaged capacity. Intuitively, this can be explained by a larger possible cumulative line resistance $nr_w + mr_b$ in an array with a larger aspect ratio. This observation presents a trade-off between the sometimes desired high aspect ratio and a high averaged capacity.

4.4 Optimal Read Threshold

In subsection 4.3.2, we observe that the channel parameters $p_3^{(ij)}$ are $p_4^{(ij)}$ are dependent on the read current threshold I_{th} — a user defined parameter that can be optimized for lower raw bit-error rate. In this section, we study the optimal threshold for each cell and for an entire array with the goal of reducing the raw bit-error rate. We choose to optimize the read resistance threshold $R_{th} = \frac{V_r}{I_{th}}$ as it is equivalent to optimizing I_{th} with fixed read voltage V_r. We define R_{th0} be the optimal threshold when no line resistance is considered, i.e.,

$$R_{th0} = \underset{R_{th}}{\operatorname{argmin}}\, qQ\left(\frac{\mu_H - \ln(R_{th})}{\sigma_H}\right) + (1-q)Q\left(\frac{\ln(R_{th}) - \mu_L}{\sigma_L}\right). \tag{4.14}$$

R_{th0} can be calculated using standard result from estimation theory. R_{th0} is clearly suboptimal when the line resistance is non-negligible.

4.4.1 Optimal Threshold for Each Cell

One simple read scheme is to use the optimal resistance thresholds for cells at different locations. We call this a different threshold for each cell (DTEC) scheme. Define the optimal threshold for the cell (i, j) to be R_{th}^{ij}. With the objective of minimizing $P(Z_{ij} = Y_{ij})$, and using equation (4.11) and equation (4.12), we have:

$$R_{th}^{ij} = \underset{R_{th}}{\operatorname{argmin}}\, qQ\left(\frac{\mu_H - \ln(R_{th} - ir_b - jr_w)}{\sigma_H}\right) + (1-q)Q\left(\frac{\ln(R_{th} - ir_b - jr_w) - \mu_L}{\sigma_L}\right). \tag{4.15}$$

Comparing equations (4.14) and (4.15), we get

$$R_{th}^{ij} = R_{th0} + ir_b + jr_w. \tag{4.16}$$

This result is intuitive as we need to shift the threshold to the right in order to compensate for the adverse effect of the cumulative line resistance.

4.4.2 Optimal Threshold for An Array

Requiring different thresholds for cells in a $m \times n$ array may not be desirable for circuit designers as doing this may require a lot more comparators. In a typical memory design, cells on the same bitline share the same sensing amplifier, so one threshold for each column is a reasonable choice. To further simplify the memory design, one may even use the same threshold for the entire memory array. Therefore, it is of interest to find the optimal threshold that minimizes the averaged BER for an entire array or a sub-array (such as a column). In this subsection, we deal with the optimal threshold for an array first; this result readily generalizes to any sub-array. We call these schemes the same threshold for many cells (STMC) schemes.

With the objective of minimizing $\frac{1}{mn}\sum_{i,j} P(Z_{ij} \neq Y_{ij})$, the optimal threshold for an array is defined as

$$\begin{aligned} R_{th\_array} = \underset{R_{th}}{\operatorname{argmin}} \frac{1}{mn} \sum_{i,j}^{m,n} \Bigg[& qQ\left(\frac{\mu_H - \ln(R_{th} - ir_b - jr_w)}{\sigma_H}\right) \\ & + (1-q)Q\left(\frac{\ln(R_{th} - ir_b - jr_w) - \mu_L}{\sigma_L}\right)\Bigg]. \end{aligned} \tag{4.17}$$

While it may be possible to heuristically optimize the single parameter R_{th}, in practice we wish to provide an analytical solution based on certain approximations and an efficient iterative search algorithm. Analytically, the optimization problem in (4.17) is hard to solve for as it involves a summation of Q-functions. We instead replace this objective function with its upper bound and try to minimize this upper bound. Using Jensen's inequality and

the fact that the Q-function is concave for a positive argument, we have the following bound:

$$\begin{aligned}&\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[qQ\left(\frac{\mu_H-\ln(R_{th}-ir_b-jr_w)}{\sigma_H}\right)+(1-q)Q\left(\frac{\ln(R_{th}-ir_b-jr_w)-\mu_L}{\sigma_L}\right)\right]\\ \leq& qQ\left(\frac{\mu_H-A}{\sigma_H}\right)+(1-q)Q\left(\frac{A-\mu_L}{\sigma_L}\right),\end{aligned} \tag{4.18}$$

where

$$A=\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\Big[\ln(R_{th}-ir_b-jr_w)\Big].$$

The gap between the two sides of this inequality is small when the Q-functions are close to being linear, which is indeed the case for Q-functions with large arguments.

We reformulate the problem using the above inequality:

$$R_{th\_array}=\underset{R_{th}}{\operatorname{argmin}}\, qQ\left(\frac{\mu_H-A}{\sigma_H}\right)+(1-q)Q\left(\frac{A-\mu_L}{\sigma_L}\right). \tag{4.19}$$

Comparing equations (4.14) and (4.19), the problem becomes of finding R_{th_array} such that

$$\ln(R_{th0})=\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\Big[\ln(R_{th\_array}-ir_b-jr_w)\Big]. \tag{4.20}$$

Equation (4.20) is hard to solve since it contains a summation of logarithm functions. We provide both an approximation to this equation that is easier to solve, as well as an iterative algorithm that produces a solution to the original equation.

Approximating each term in the right hand side summation by $\ln(R_{th\_array}-\frac{m+1}{2}r_b-\frac{n+1}{2}r_w)$, the approximate solution of (4.20) can be found:

$$R_{th\_array}\approx R_{th\_array\_appx}=R_{th0}+\frac{m+1}{2}r_b+\frac{n+1}{2}r_w. \tag{4.21}$$

This approximation can be also interpreted as averaging of the optimal thresholds, given by equation (4.16), of all cells.

We propose Algorithm 1 to compute the exact solution of equation (4.20).

Algorithm 1: STMC threshold solver algorithm..

1. Initialize $R_{th}^{(0)} = R_{th0}, l = 0$.
2. $l = l + 1$.
   Calculate $R_{th}^{(l)}$ such that $\ln(R_{th}^{(l)}) = \ln(R_{th0}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\ln\left(1 - \frac{ir_b + jr_w}{R_{th}^{(l-1)}}\right)$.
3. Repeat step 2 until a certain iteration is reached or $R_{th}^{(l)} - R_{th}^{(l-1)} \leq \epsilon$. Let $R_{th\_array} = R_{th}^{(l)}$.

The STMC threshold solver algorithm is inspired by how the summation of logarithm functions is handled in the Expectation Maximization algorithm. The convergence of the STMC threshold solver algorithm is demonstrated in the following lemma.

Lemma 7. *The STMC threshold solver algorithm converges to the solution of equation (4.20).*

Proof. Equation (4.20) can be rewritten as:

$$\ln(R_{th\_array}) = \ln(R_{th0}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\ln\left(1 - \frac{ir_b + jr_w}{R_{th\_array}}\right).$$

With $R_{th}^{(0)} \geq R_{th\_array}$, the following inequality holds:

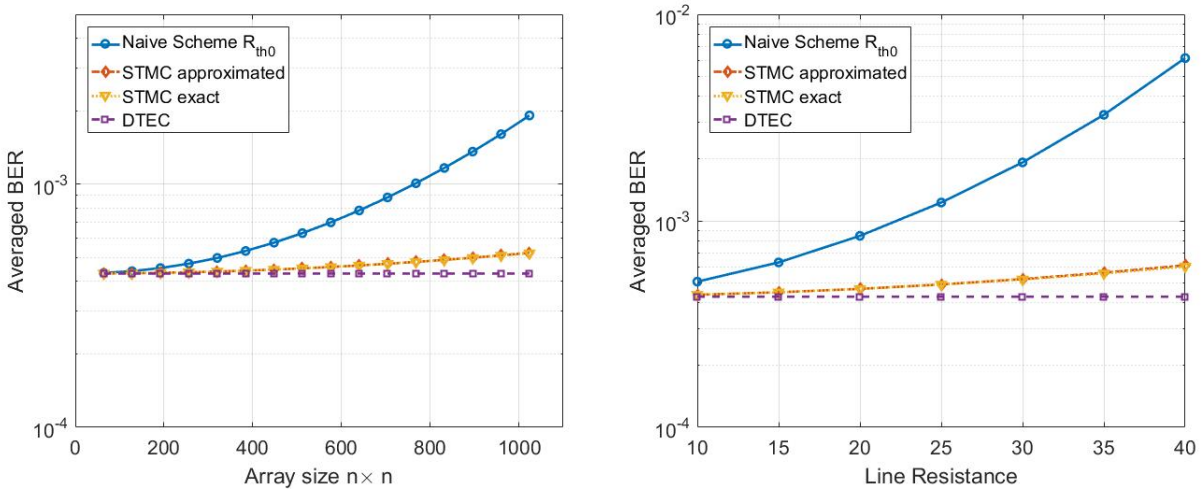
$$\ln(R_{th\_array}) \geq \ln(R_{th}^{(1)}) = \ln(R_{th0}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\ln\left(1 - \frac{ir_b + jr_w}{R_{th}^{(0)}}\right).$$

By induction, $\ln(R_{th\_array})$ can be bounded: $\ln(R_{th}^{(2j+1)}) \leq \ln(R_{th\_array}) \leq \ln(R_{th}^{(2j)}), j \in \mathbb{Z}$. With $R_{th}^{(2)} > R_{th}^{(0)} = R_{th0}$, since $\ln(R_{th}^{(l)}) = \ln(R_{th0}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\ln\left(1 - \frac{ir_b + jr_w}{R_{th}^{(l-1)}}\right)$, the subsequent upper bounds and lower bounds are shrinking toward $\ln(R_{th\_array})$, i.e., $\ln(R_{th}^{(2j+2)}) < \ln(R_{th}^{(2j)})$ and $\ln(R_{th}^{(2j+3)}) > \ln(R_{th}^{(2j+1)})$. Therefore, the STMC threshold solver algorithm converges. ■

For the simulations we report on in the next subsection, the STMC threshold solver algorithm converges in 5 iterations which is more effective than performing a line search to solve equation (4.17) empirically.

4.4.3 Simulation Results

We simulate arrays to examine how the averaged read BERs are affected by different read thresholding schemes. Four different thresholding schemes are compared: the naive scheme where R_{th0} is used as the threshold; the DTEC scheme; the STMC scheme with the approximated solution in (4.21); and the STMC scheme with the exact solution solved by Algorithm 1. Unless otherwise mentioned, we use parameters from Table 4.1. We first vary the array size and fix $r_b = r_w = 30\Omega$ and report the result in Fig. 4.4a. We also vary the wire resistance and fix $n = m = 1024$ and report the result in Fig. 4.4b.



(a) Averaged read BER v.s. array size using different thresholding scheme.

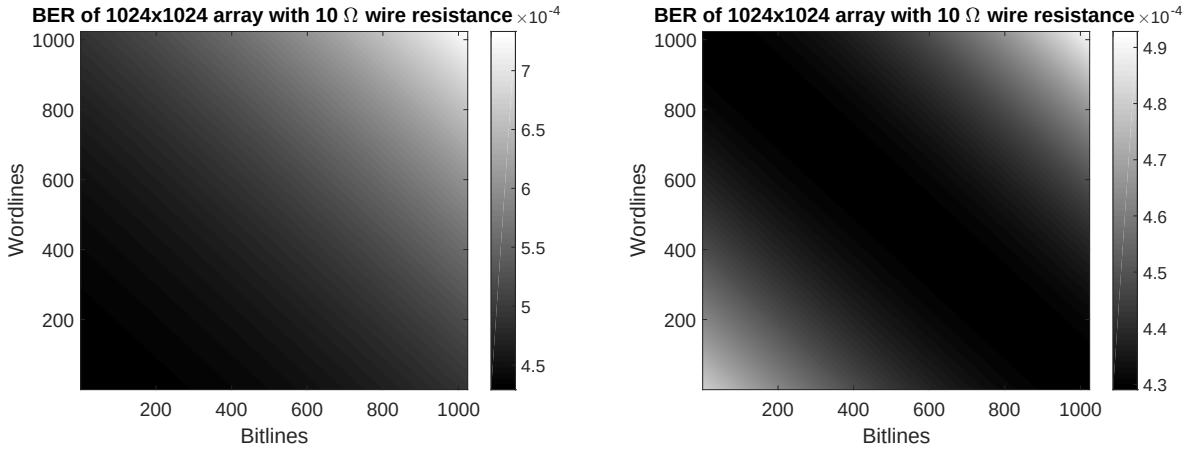
(b) Averaged read BER v.s. wire resistance using different thresholding scheme.

Figure 4.4: Simulation results using the optimal threshold.

From both Fig. 4.4a and Fig. 4.4b, we observe that both DTEC and STMC schemes reduce the read BER significantly compared with the naive thresholding scheme. The DTEC scheme compensates for each cell based on its location, and, as a result, the averaged BER is independent of the array size and wire resistance. As expected, the averaged BERs using the exact solution of the STMC scheme is smaller than the averaged BERs using the

approximated solution. However, the improvement is incremental and can not be identified on the plots. This shows that equation (4.21) approximates the solution of equation (4.20) very well.

To investigate the effect of the optimized read threshold on the non-uniformity of read reliability, we generate heatmaps depicting the read BER with the unoptimized read threshold (Fig. 4.5a) and with the optimized read threshold by the STMC scheme (Fig. 4.5b).



(a) Read BER heatmap with R_{th0}.

(b) Read BER heatmap with STMC scheme.

Figure 4.5: Heatmap comparison between un-optimized and optimized read thresholds.

Similar to what we observed in Fig. 4.3a in subsection 4.3.4, with the un-optimized threshold, the lower left corner cell is the most reliable one and the upper right corner cell is the least reliable one. With the optimized read threshold, the non-uniformity pattern changes and the reliable cells lie on the diagonal of the array. This result is expected because from equation (4.21), we observe that the approximated solution for the STMC scheme is the optimal read threshold for the center-most cell of an array. Note that in our simulation, the device parameters are selected such that the read channel with un-optimized threshold is comparable to the write channel. Because of this set-up, with the optimized read threshold, the read channel is dominated by the write channel and the non-uniformity pattern of the

cascaded channel is unchanged.

4.5 Channel Coding for Storage-Class Memory (SCM) Applications

Error correction codes (ECCs) have become an essential part in memory systems. In an array with largely non-uniform device reliability, efficient ECC solutions must take this non-uniformity into consideration by either mitigating it or leveraging it. Channels with non-uniformity can be considered as channels with SNR variation [EHW+18] or non-stationary channels [ZFW+19, Mah20]. These models have been studied in the ECC literature in the context of e.g., LDPC codes and Polar codes for storage applications [ZFW+19, EHW+18, Mah20]. However, these complex codes with long block length and high decoding latency (on the order μs [ZZS+13]) are not compatible with the small decoding granularity and fast decoding requirements of the SCM applications. Therefore, in targeting the SCM applications, we study efficient coding schemes for crossbar resistive memory with high line resistance and base them on simple yet effective BCH codes with short block length. BCH codes, which are characterized by the block length (n), number of information bits (k), and the error correction capability (t), have decoding latency on the order of ns and are widely adopted in crossbar resistive memory [NXX12, CAS+18, MCYC17].

In subsection 4.5.1, based on a single BCH code, we propose an interleaved coding scheme to mitigate the non-uniformity by allowing cells that store different codewords to have similar averaged RBER. In subsection 4.5.2, we propose a method to use different BCH code in different wordline to leverage the non-uniformity, and do so without interleaving. A systematic framework to optimize the allocations of codes to wordlines is also proposed in V.B. The two proposed approaches are suitable for different scenarios. The interleaved coding scheme in subsection 4.5.1 is conceptually simpler and requires only one pair of ECC encoder and decoder. Meanwhile, the interleaved coding scheme requires interleaver and de-interleaver,

which adds additional hardware and latency overhead. The proposed scheme in subsection 4.5.2 does not use an interleaving operation but requires an optimization process in the design stage and also requires additional ECC encoders/decoders due to the use of multiple ECCs. Also, because the optimization steps are not typically performed on-the-fly, performance of the proposed scheme based on an optimized allocation can deteriorate if device parameters vary significantly in time. We present both schemes for completeness and the usefulness of each of them in a specific application depends on their performance and the trade-off between hardware/latency overhead of the interleaver/de-interleaver and that of the additional encoders/decoders. Some performance trade-offs between these two approaches will be discussed at the end of subsection 4.5.2 when performance of these two schemes are compared. Note that in the following two subsections, our simulations consider both the read and write channel and are based on parameters in Table 4.1. The read thresholds are optimized using the STMC scheme in Section 4.4.

4.5.1 Single ECC with Interleaving

Suppose a codeword is stored in a wordline in the crossbar memory. When employing the same ECC for all wordlines, the non-uniformity of raw bit-error rates (RBERs), as depicted in Fig. 4.3a, transforms into the non-uniformity of undetected bit-error rates (UBERs) and thus makes this coding scheme inefficient. If the ECC is designed for an averaged case, e.g., the channel conditions in the middle wordlines, it can be an overkill for the channel conditions in the lower wordlines while being inadequate for the channel conditions in the upper wordlines. One approach to mitigate non-uniformity of channels is interleaving e.g., [EHW$^+$18]. Based on the dependency of channel parameters on the row and column indexes of the cell location, we propose a sub-diagonal interleaved coding scheme which is illustrated in Fig. 4.6.

Based on this sub-diagonal interleaved coding scheme, we store each codeword in cells that are located on the same diagonal or a sub-diagonal of the memory array, instead of cells that are located on the same wordline. Note that this interleaved coding scheme can

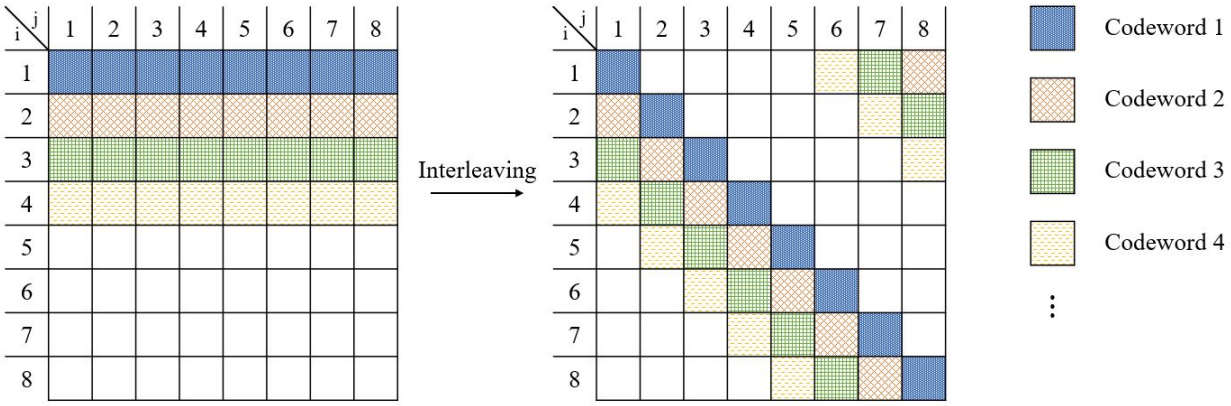


Figure 4.6: Illustration of the sub-diagonal interleaved coding scheme on an 8×8 Array.

be readily generalized to a non-square array by storing a codeword in cells located on the main diagonal or a circularly shifted main diagonal of the rectangular array.

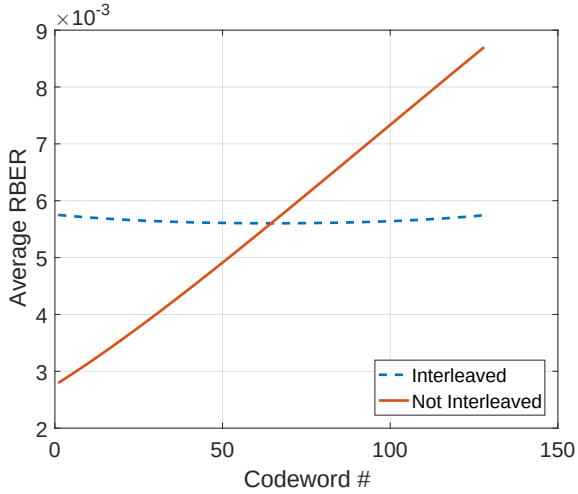
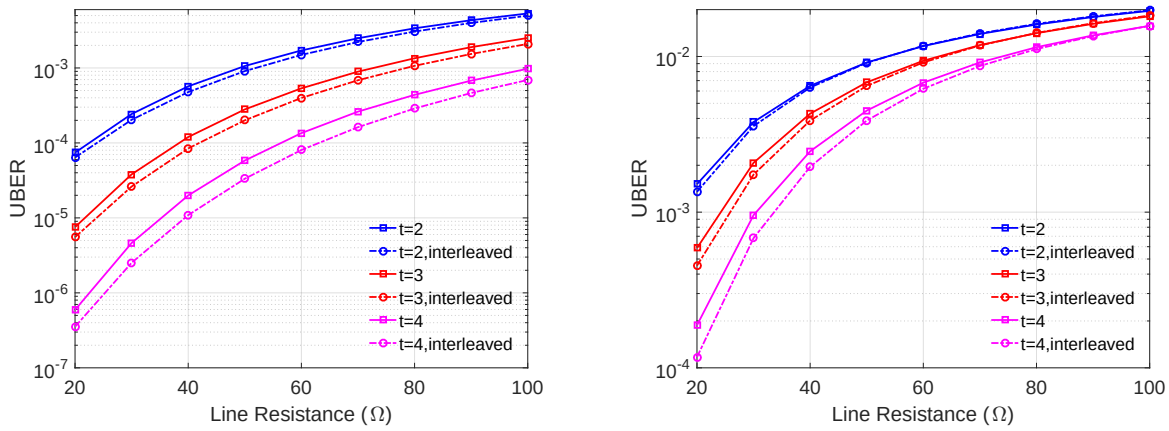


Figure 4.7: The averaged RBER for cells storing codewords in an 128×128 array with $r_w = r_b = 50\Omega$.

To study the effectiveness of this interleaved coding scheme, we first compare the averaged RBER in cells that store each codeword when using the interleaved coding scheme with the averaged RBER in cells that store each codeword where bits of the codeword are stored in a

wordline. The results for an 128×128 array are shown in Fig. 4.7 and we observe that the non-uniformity of averaged RBER among codewords is largely mitigated.



(a) n=128. (b) n=256.

Figure 4.8: Decoding performance of the interleaved coding scheme.

We next present the decoding performance of this interleaved coding scheme using BCH codes with block length $n = 128$ and 256, and error correction capability $t = 2, 3$, and 4 in Fig. 4.8. In Fig. 4.8, the interleaved coding scheme effectively reduces the UBER with a maximum reduction of 45%. We observe that the interleaved coding scheme is more effective in the regime of short block length and moderate line resistance. We conjecture that even though the interleaved coding scheme is able to mitigate the non-uniformity of the averaged RBER seen by each codeword, it exaggerates the non-uniformity of RBER within a codeword. With a larger block length and line resistance, this trade-off is less favored and the interleaved coding scheme is less effective. We also observe that the UBER improvement is larger with a larger error correction capability. This result is due to the observation that a larger error correction capability gives the interleaved coding scheme more room to balance channels where the code is an overkill and channels where the code is insufficient, under the non-interleaved case.

4.5.2 Multiple ECCs with Optimized Code Allocation

Although the interleaved coding scheme in the previous subsection is conceptually simple and requires only one ECC, interleaving and de-interleaving are costly operations. Moreover, these operations create additional difficulties if one seeks to utilize the parallel read capability of a memory with the crossbar structure [LYWW13]. Without interleaving, one intuitive approach to leverage the non-uniform RBERs is to use a strong ECC for relatively unreliable wordlines and use a weak ECC for relatively reliable wordlines. This approach has a reasonable complexity overhead compared to using a single ECC because encoders and decoders for structured codes, e.g., BCH codes, with different error correction capabilities can share the same encoding and decoding circuitry. Given a set of ECCs with varying error correction capability and a set of wordlines with varying channel parameters, we refer to the problem of designating an ECC to an wordline for all wordlines as the code allocation problem. Similar problems, i.e., using different ECCs for different channel conditions, have been studied in communications under the context of adaptive coding [Vuc91] where the channel condition is often time dependent (instead of being location dependent as in our set-up). In this subsection, we present a systematic framework, referred to as the location dependent code allocation (LDCA) framework, for solving the optimal code allocation problem. We first present the formulation of the optimal location dependent code allocation (LDCA) problem as a standard optimization problem and then propose an effective algorithm to solve for a sub-optimal solution, which is shown to reduce the UBER effectively.

Suppose we have a set $\mathbb{C}$ of L error correction codes $C_1, C_2, \cdots, C_L$ with the same block length n and different error correction capabilities. In our specific case, we consider BCH codes with a corresponding set of error correction capabilities $\mathbb{T} = \{t_1, t_2, \cdots, t_L\}$. The memory array is of size $m \times n$ and the crossover probabilities of the cascaded BAC in Fig. 4.2, $p_5^{(ij)}$ and $p_6^{(ij)}$, are known based on the channel model. For each wordline i, we choose a code from $\mathbb{C}$ and store the encoded bits in it. Let $\boldsymbol{C} \in \mathbb{R}^{m \times L}$ be an array whose element

c_{il} is the UBER (cost) when code C_l is applied to wordline i and $\boldsymbol{c}_i^T$ be a row of $\boldsymbol{C}$. Let $\boldsymbol{A} \in \mathbb{R}^{L\times m}$ be an array whose column $\boldsymbol{a}_i \in \mathbb{R}^L$ is an one-hot vector denoting the code selection for wordline i, i.e., for a given i, $a_{li} = 1$ if C_l is used for wordline i and $a_{li} = 0$ otherwise. Let $\boldsymbol{r} \in \mathbb{R}^L$ represent the rate of the codes in $\mathbb{C}$ and let R_{goal} be the desired storage rate for the memory array. Under the constraint of rate R_{goal}, we seek to find a code allocation matrix $\boldsymbol{A}$ such that the overall UBER is minimized. The optimal LDCA problem can therefor be formulated as the following integer programming problem:

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \sum_{i=1}^{m} \boldsymbol{c}_i^T \boldsymbol{a}_i = Tr(\boldsymbol{A}\boldsymbol{C}) \\ \text{s.t.} \quad & \boldsymbol{a}_i^T \mathbf{1} = 1, \quad i = 1, \cdots, m \\ & a_{li} \in \{0, 1\} \\ & \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{r}^T \boldsymbol{a}_i \leq R_{goal} \end{aligned} \tag{4.22}$$

In the above optimization problem, $\mathbf{1}$ is the all-ones vector. To solve the above optimization problem, we first need to estimate the cost matrix $\boldsymbol{C}$ from the channel parameters. We calculate $\boldsymbol{C}$ based on the following approximation:

$$c_{il} \approx 1 - \sum_{e=0}^{t_l} \binom{n}{e} (\bar{p}_i)^e (1 - \bar{p}_i)^{n-e}, \tag{4.23}$$

where

$$\bar{p}_i = \frac{1}{n} \sum_{j=1}^{n} \left[q p_5^{ij} + (1 - q) p_6^{ij} \right].$$

In (4.23), we approximate the n BAC channels of cells on a wordline by an averaged binary symmetric channel (BSC) with parameter $\bar{p}_i$ and calculate the analytical UBER for a BCH code based on this approximation.

With the approximated cost matrix $\boldsymbol{C}$, we propose Algorithm 2 to solve the integer programming problem (4.22) by solving its relaxed linear programming problem (4.24) iter-

atively:

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \sum_{i=1}^{m} \boldsymbol{c}_i^T \boldsymbol{a}_i = Tr(\boldsymbol{AC}) \\ \text{s.t.} \quad & \boldsymbol{a}_i^T \mathbf{1} = 1, \quad i = 1, \cdots, m \\ & 0 \leq a_{li} \leq 1 \\ & \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{r}^T \boldsymbol{a}_i \leq R_{LP} \end{aligned} \tag{4.24}$$

Algorithm 2: LDCA solver algorithm.

1. Solve the linear programming problem (4.24) with $R_{LP} = R_{goal}$ to get A^*.
2. Estimate the code allocation matrix $\hat{A}$ based on the maximum in each column of A^*. Namely, in $\hat{\boldsymbol{a}}_i$, only the argmax($\boldsymbol{a}_i^*$)-th element is set to 1 and all other elements are 0.
3. Calculate the true rate based on $\hat{A}$, i.e., $\hat{R} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{r}^T \boldsymbol{a}_i$.
4. Terminate the algorithm if $\hat{R}$ is within a certain tolerance of R_{goal}. Otherwise, update R_{LP}. Namely,

   if $R_{goal} - R_{tol} \leq \hat{R} \leq R_{goal} + R_{tol}$ **then**
   | Algorithm terminates;

   else if $\hat{R} > R_{goal} + R_{tol}$ **then**
   | Update R_{LP} with $R_{LP} - R_{update}$;

   else
   | Update R_{LP} with $R_{LP} + R_{update}$;

   end
5. Repeat step 1-4 until termination or certain number of iterations is reached.

In Algorithm 2, R_{tol} and R_{update} are user defined parameter specifying the tolerance between the true rate $\hat{R}$ and the target rate R_{goal}, and the step for updating the rate used in the linear programming problem, respectively. Note that in our experiment with $L = 3$, Algorithm 2 terminates in a single iteration.

To better illustrate the effectiveness of Algorithm 2, we present rows of the solved code allocation matrix A^* for an array with $n = m = 128$ and varying line resistance in Fig. 4.9. The channel parameters are computed based on our model in Section 4.3 and device parameters in Table 4.1. We include $L = 3$ codes in $\mathbb{C}$: BCH(128,100,3), BCH(128,93,4) and BCH(128,86,5) and set R_{goal} to be the rate of BCH(128,93,4). Based on the definition of $\boldsymbol{A}$, each row of A can be interpreted as the relative "weight" of each code. We observe that with the 10Ω line resistance, the non-uniformity is mild and the solution from Algorithm 2 suggests that using a single code BCH(128,93,4) is sufficient. As the line resistance goes up, the solution from Algorithm 2 suggests the usage of a strong code BCH (128,86,5) for the worse wordlines and a weak code BCH(128,100,3) for the better wordlines.

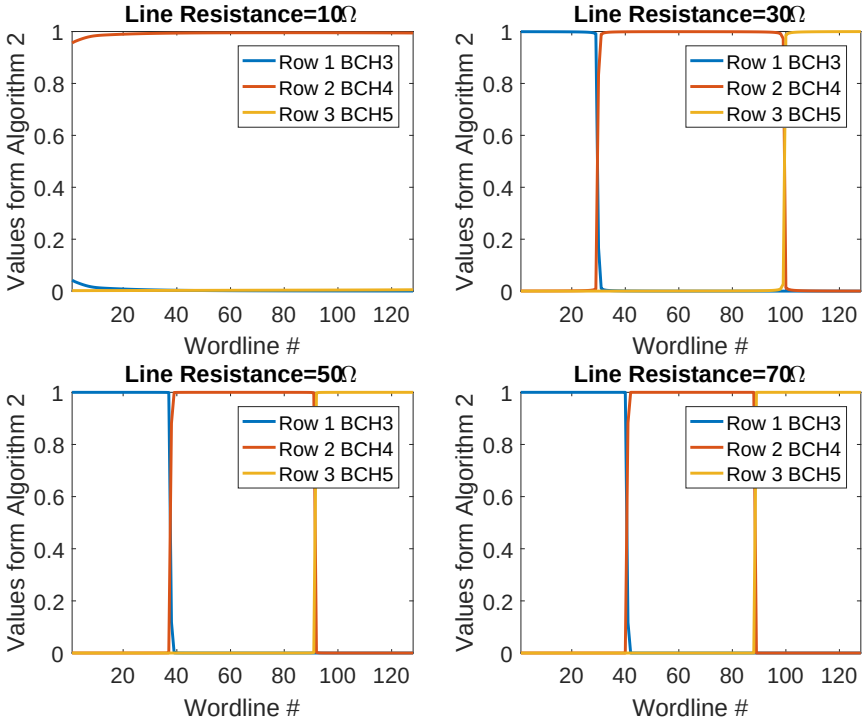
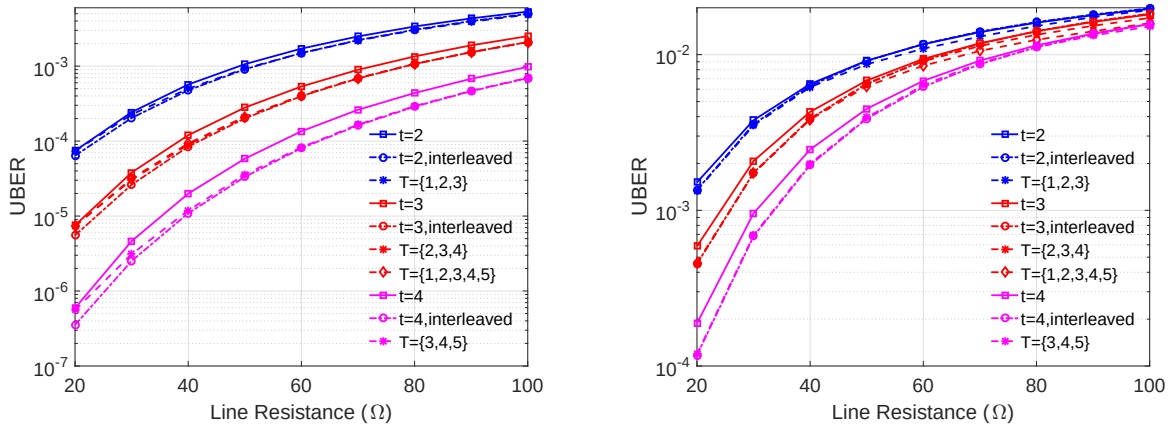


Figure 4.9: Solutions of Algorithm 2 for arrays with various line resistance values.

We perform simulations to study the effectiveness of our proposed LDCA framework. In our simulations, two block lengths $n = 128$ and $n = 256$ are studied, and the corresponding

memory arrays are of sizes 128×128 and 256×256, respectively. Line resistances are varied while the remaining device parameters are from Table 4.1. For each set of the array parameters, four different sets of codes are tested. These sets of codes are the standard BCH codes with $\mathbb{T} = \{1, 2, 3\}$, $\mathbb{T} = \{2, 3, 4\}$, $\mathbb{T} = \{3, 4, 5\}$, and $\mathbb{T} = \{1, 2, 3, 4, 5\}$. Their target rates are set to equal the rate of the BCH codes with $t = 2, t = 3, t = 4$, and $t = 3$, respectively. For each group of codes, the optimized code allocation is first solved for by the LDCA solver algorithm, and then the UBER based on the optimized code allocation is simulated. We compare the performance of each group of codes with the performance of the BCH code of the same rate with and without the interleaved coding scheme. The results are shown in Fig. 4.10.



(a) n=128. (b) n=256.

Figure 4.10: Decoding performance of sets of codes based on LDCA framework.

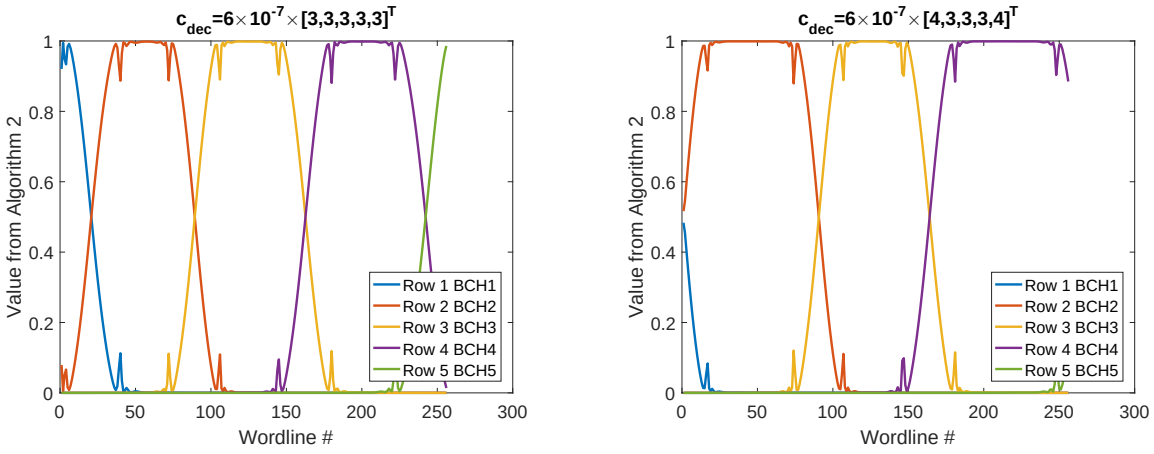
From Fig. 4.10, we observe that the scheme utilizing a set of codes based on our LDCA framework shows consistent improvement compared to the scheme utilizing a single code. The UBER of a set of $L = 3$ codes is comparable to the UBER of a single code of the same rate with interleaving. In the 128×128 array, the set of codes with $\mathbb{T} = \{1, 2, 3, 4, 5\}$ does not outperform the set of codes with $\mathbb{T} = \{2, 3, 4\}$. Meanwhile, in the 256×256 array, the set of codes with $\mathbb{T} = \{1, 2, 3, 4, 5\}$ outperforms the set of codes with $\mathbb{T} = \{2, 3, 4\}$ by a

consistent 7% in the high line resistance regime. This observation suggests that the utility of allowing for more ECCs to be used in the system depends on the severity of non-uniformity in the memory array. Comparing the proposed schemes in subsection 4.5.1 and that in subsection 4.5.2, we observe that the decoding performance of the interleaved coding scheme and the decoding performance of the optimized sets of $L = 3$ codes are very similar in most of the tested operating regime. In the regime of short block lengths, i.e., $n = 128$, and low line resistance, the interleaved coding scheme outperforms the scheme based on multiple ECCs. We conjecture that this is because interleaving is able to mitigate the non-uniformity despite its severity, whereas the scheme based on multiple ECCs requires notable RBER differences among the wordlines to perform well as the error correction capabilities of ECCs are only expressed in discrete integer values. For larger block lengths, i.e., $n = 256$, and high line resistance, the scheme based on the set of $L = 5$ codes shows notable improvement compared to the interleaved coding scheme. This shows that when allowing for more ECCs to be used in the LDCA framework, a scheme based on the optimized code allocation of a larger collection of codes is more promising than the interleaved coding scheme when the non-uniformity across the memory array is severe.

Note that as the effectiveness of the scheme utilizing the proposed LDCA framework depends on the code length and severity of non-uniformity, other device parameters from Table 4.1 also impact the performance of the proposed scheme. Because the main focus of this chapter is on the effect of line resistance, we only present results with varying line resistance and the study on the effect of other device parameters are left for future work. Also note that although our simulations is based on BCH codes and our proposed channel models, the LDCA framework is a general one and it is applicable to other systems with location dependent non-uniformity even when bearing a different channel model and/or utilizing different ECCs. The LDCA framework can be readily generalized to other channels models and ECCs by changing how the cost matrix $\boldsymbol{C}$ is calculated in Equation (4.23).

If one seeks to take into consideration the cost of decoders with different error correction

capability or if one seeks to find solutions that use fewer codes without manually restricting L, a simple variant of the LDCA framework can be used by adding a regularization term $\boldsymbol{c}_{dec}^T \boldsymbol{A}\mathbf{1}$ in the objective functions of the optimization problems where $\boldsymbol{c}_{dec} \in \mathbb{R}^L$ is a vector that contains appropriate weights for the decoders. To demonstrate the effectiveness of this regularized variant, we present an example by generating plots similar to Fig. 4.9 with and without this regularization term in Fig. 4.11. For an 256×256 array with $r_w = r_b = 30\Omega$, the solution from the LDCA solver algorithm (Fig. 4.11a) suggests us to use the BCH(256,248,1) code for the first few wordlines and the BCH(256,216,5) for the last few wordlines. In this moderate line resistance regime, one may wish to avoid using BCH(256,248,1) and BCH(256,216,5) as observation in Fig. 4.10 suggests that improvement of using more codes is incremental when the non-uniformity is moderate. Instead of manual disallowing those two codes in $\mathbb{C}$, an alternative is to use the regularization term $\boldsymbol{c}_{dec}^T \boldsymbol{A}\mathbf{1}$. With this regularization term, we get a solution that is shown in Fig.4.11b and this solution does not use BCH(256,248,1) and BCH(256,216,5). This regularized variant is therefore capable of automatically penalizing the use of more ECCs when the non-uniformity is moderate while still allowing more ECCs to be used when the non-uniformity is severe.



(a) un-regularized solution.

(b) regularized solution.

Figure 4.11: An example of using the regularization term in the LDCA framework.

CHAPTER 5

Conclusion

In this dissertation, focusing on the two applications of crossbar resistive memory, we address the reliability issues using tools from estimation theory and channel coding theory. Based on models from communication theory, our proposed schemes improves the robustness of the system by improving the learning accuracy in the accelerator application and reducing the RBER/UBER in the SCM application. In chapter 2, we show that computing Hamming distance in-memory is feasible under resistance variation and propose a error detection/correction scheme that effectively improves the classification accuracy without bit-level information. In chapter 3, by characterizing the inter-cell dependency of the sneak-path problem, we propose an adaptive thresholding scheme that reduces the RBER. In chapter 4, we present channel models for the write and read operations under high line resistance and propose schemes for efficient channel coding under the spatially non-uniform channel characteristics.

This work can be extended in many directions bearing the same objective: improving the robustness of systems that are built with crossbar resistive memory. On the accelerator application side, future work can be done by studying the effect of resistance variation in practical accelerator structure that accelerates binary neural networks. On the SCM application side, one can study efficient coding construction with low decoding latency on the order of ns.

Appendix A

An Approximation Justification

In Chapter 2, the following approximations are made:

$$\frac{H_{i,x}H_{i,y}}{H_{i,x}+H_{i,y}} \approx F_i \sim \mathcal{N}(\mu_H/2, \sigma_H^2/8), \tag{A.1}$$

$$\frac{L_{i,x}L_{i,y}}{L_{i,x}+L_{i,y}} \approx T_i \sim \mathcal{N}(\mu_L/2, \sigma_L^2/8), \tag{A.2}$$

$$\frac{H_{i,x}L_{i,y}}{H_{i,x}+L_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right), \tag{A.3}$$

$$\frac{L_{i,x}H_{i,y}}{L_{i,x}+H_{i,y}} \approx S_i \sim \mathcal{N}\left(\frac{\mu_L}{1+\epsilon}, \frac{\sigma_L^2}{(1+\epsilon)^4}\right). \tag{A.4}$$

Here we show that the approximated distributions are close to the original distributions by studying these approximations using data from a variety of memristor technology reported in the literature [CL11]. We model the reported resistance variations for 9 types of ReRAM devices. For each ReRAM device, let L and H denote the random variables for low state conductance and high state conductance, respectively. We assume L and H follow the following two Gaussian distributions respectively:

$$H \sim \mathcal{N}(\mu_H, \sigma_H^2), L \sim \mathcal{N}(\mu_L, \sigma_L^2). \tag{A.5}$$

For each ReRAM device, μ_H and μ_L are set to be the reciprocal value of its mean low-state resistance and high-state resistance, i.e. R_{avg}^L and R_{avg}^H, respectively. σ_H and σ_L are calculated using the following rule:

$$\sigma_H = \frac{1}{2}\min\left(\frac{1}{R_{min}^H}-\frac{1}{R_{avg}^H}, \frac{1}{R_{avg}^H}-\frac{1}{R_{max}^H}\right), \sigma_L = \frac{1}{2}\min\left(\frac{1}{R_{min}^L}-\frac{1}{R_{avg}^L}, \frac{1}{R_{avg}^L}-\frac{1}{R_{max}^L}\right),$$

where R^L_{min} and R^L_{max} are the smallest and largest low-state resistance value reported, respectively. Similarly, R^H_{min} and R^H_{max} are the smallest and largest low-state resistance values reported. The resulting models for the 9 ReRAM devices are summarized in Table A.1.

For each ReRAM device, we calculate four discretized distributions on the left hand side of the approximation. We first generate 10^7 samples for $H_{i,x}$, $H_{i,y}$, $L_{i,x}$ and $L_{i,y}$ according to the (A.5). Then we calculate the discretized distributions for expressions on the left-hand-side of Approximation (A.1), (A.2), (A.3) and (A.4) by dividing all samples into 100 bins with equal bin width. We also calculate the discretized approximated distributions (the right-hand-side of Approximation (A.1), (A.2), (A.3) and (A.4) using the same bins. For each approximation in (A.1), (A.2), (A.3) and (A.4) we calculate the Bhattacharyya distance for each pair of distributions using the following equations:

$$D_B(p, q) = -\ln(BC(p, q)), \tag{A.6}$$

where

$$BC(p, q) = \sum_{x \in X = 1,..,100} \sqrt{p(x)q(x)}, \tag{A.7}$$

where p is the discretized original distribution and q is the discretized approximated distribution.

The four distance metrics for each ReRAM device, D^1_B, D^2_B, D^3_B and D^4_B—the Bhattacharyya distance between the left distribution and the right distribution of approximations (A.1), (A.2), (A.3) and (A.4), respectively—are listed in Table A.1. Bhattacharyya distance close to zero means the two distribution are close. Thus we have showed our approximations are suitable for a variety of ReRAM devices which have a large range of resistance variation.

Table A.1: ReRAM models and Bhattacharyya distances.

	μ_L	σ_L	μ_H	σ_H	D_B^1	D_B^2	D_B^3	D_B^4
$TiOx$	1.0e-3	2.5e-4	2.5e-2	2.5e-3	6.5e-4	5.0e-3	8.3e-5	8.3e-5
$HfOx^{(1)}$	1.0e-3	2.1e-4	5.0e-3	8.3e-4	2.0e-3	3.5e-3	9.2e-4	9.2e-4
$AuZrOx^{(1)}$	3.3e-7	1.0e-7	1.4e-2	2.1e-3	1.5e-3	8.4e-3	3.2e-7	2.4e-7
$SrZrO3$	5.0e-7	8.3e-8	1.7e-3	3.3e-4	3.0e-3	2.0e-3	2.5e-7	2.5e-7
$CuGeSe$	1.7e-6	3.3e-7	3.3e-4	6.7e-5	3.0e-3	3.0e-3	1.5e-6	1.6e-6
$CoOx$	1.3e-5	3.8e-6	2.0e-4	3.8e-5	2.5e-3	7.7e-3	2.9e-4	3.0e-4
$HfOx^{(2)}$	1.3e-5	3.8e-9	1.0e-4	2.5e-5	5.0e-3	7.7e-3	2.0e-7	2.6e-7
$TiON$	1.7e-7	3.3e-8	5.0e-5	1.6e-5	9.7e-3	3.0e-3	9.0e-6	9.0e-6
$AuZrOx^{(2)}$	2.5e-8	6.3e-9	1.0e-5	2.5e-6	5.0e-3	5.0e-3	8.0e-7	9.5e-7

Appendix B

Probabilistic Characterization of Inter-cell Dependency

In this appendix, under the modeling of the sneak-path event of a 1D1R structured array, we calculate the probabilities introduced in Chapter 3. Note that although some of the latter lemmas we present may look complicated, their nature is simple by using the combinatoric property of the sneak-path event defined by our modeling. We include Figure B.1 that contains some of the useful notation and indexes used in the proof of following lemmas. We will comment on possible approximations for the following lemmas at the end of the appendix. We use the binomial coefficient $\binom{n}{k}$ that is defined for $0 \leq k \leq n$ and equals 0 otherwise.

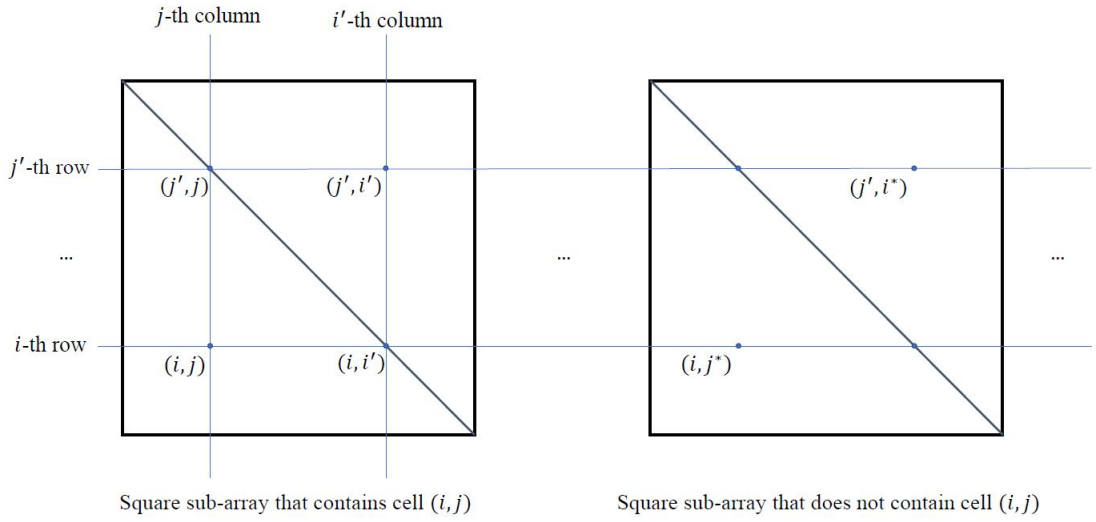


Figure B.1: Some notation and indexes used in the proofs.

Lemma 8. *Without conditioning on any known information cell, the probability that a pre-*

coded cell (cell that stores Aij with $i = j'$,i.e., $j = i'$) does not incur a sneak-path event is:

$$P_0(e_{ij} = 0) = \sum_{u=0}^{m-1} \binom{m-1}{u} q^u (1-q)^{m-1-u} \left[P_0(e_{ij} = 0|u)\right]^r, \tag{B.1}$$

where

$$P_0(e_{ij} = 0|u) = \sum_{v=0}^{m-1} \sum_{k=0}^{\min(u,v)} \binom{u}{k} \binom{m-1-u}{v-k} q^v (1-q)^{m-1-v} (1-p_f q)^{uv-k}. \tag{B.2}$$

Proof. This probability is derived by conditioning on certain parameters and then summing, over all possible configurations, the probabilities that each of these configurations does not cause a sneak-path event at cell (*i*,*j*) with $i = j'$, i.e., $j = i'$. The term configuration, here and elsewhere, refers to an array (sub-array) with known information (depending on the parameters) on the row(s) and column(s) of cell(s) of interest, i.e., cell(s) which we want to not incur a sneak-path event. We first condition on the number of 1's on the j-th column (u), i.e., we select u rows, with an index set I_u, that have 1's on the j-th column. We then divide the fat array into r square sub-arrays and consider each square sub-array separately conditioned on the selected u rows. This condition allows us to consider each square sub-array independently. For each square sub-array, we condition on the number of 1's on the i-th row (v), i.e., we select v columns, with an index set I_v, that have 1's on the i-th row. Here and elsewhere in the appendix, since we consider the square sub-arrays separately, all index sets have indexes that stand for the positions in the square sub-array. Therefore, we define all index sets to be subsets of $I_{all} = \{1, \cdots, m\} \backslash \{i, j'\}$. For each square sub-array, we also condition on $k = |I_u \cap I_v|$. For each configuration with parameters u, v and k, in order to guarantee no sneak-path event at cell (*i*,*j*), we need the $uv - k$ information cells, lying on the intersection of the u rows and v columns, to either store a 0 or store a 1 with a non-failing diode (with probability $1 - p_f q$). ■

In the following lemmas, the probabilities are all conditioned on $A_{ij} = 0$ (omitted for clarity) where $i \neq j'$, i.e., $j \neq i'$.

Lemma 9. *The probability that an information cell does not incur a sneak-path event is:*

$$P(e_{ij}=0)=\sum_{u=0}^{m-2}\binom{m-2}{u}q^u(1-q)^{m-2-u}P^{(1)}(e_{ij}=0|u)\left[P^{(2)}(e_{ij}=0|u)\right]^{r-1}, \tag{B.3}$$

where

$$P^{(1)}(e_{ij}=0|u)=\sum_{v=0}^{m-2}\sum_{k=0}^{\min(u,v)}\binom{u}{k}\binom{m-2-u}{v-k}q^v(1-q)^{m-2-v}(1-p_fq)^{uv-k}, \tag{B.4}$$

and

$$P^{(2)}(e_{ij}=0|u)=qP^{(1)}(e_{ij}=0|u)(1-p_fq)^u+(1-q)P^{(1)}(e_{ij}=0|u). \tag{B.5}$$

Proof. We first condition on selecting u rows that have 1's on the j-th column. Again, we divide the fat array into r square sub-arrays. For an information cell, the probabilities that no sneak-path event occurs at cell (i,j), conditioned on u selected rows, are different for the square sub-array that contains cell (i,j) and the $r-1$ square sub-arrays that does not contain cell (i,j) and thus need to be considered separately. We use the superscript (1) and (2) to denote this difference in our probability calculation. The calculation of $P^{(1)}(e_{ij}=0|u)$ is similar to $P_0(e_{ij}=0|u)$ in Lemma 8 except $|I_{all}|=m-2$. The calculation of $P^{(2)}(e_{ij}=0|u)$ is divided into two cases and the probability for each case can be calculated utilizing $P^{(1)}(e_{ij}=0|u)$. In the case that the cell (i,j^*), $j^*\equiv j \pmod m$, stores a 0, the configuration of this square sub-array is the same as the configuration of the square sub-array that contains cell (i,j). In the case that the cell (i,j^*) stores a 1, we additionally require u cells on the intersection of j^*-th column and the u rows to either store a 0 or store a 1 with a non-failing diode. ■

Lemma 10. *Let $P(e_{ij}=0,e_{ii'}=0)$ denote the joint probability that both an information cell that stores 0 and the reference cell on the same row do not incur sneak-path events*

simultaneously. We have:

$$P(e_{ij}=0,e_{ii'}=0)=\sum_{u=0}^{m-2}\sum_{u'=0}^{m-2}\sum_{o=0}^{\min(u,u')}\binom{m-2}{u}\binom{u}{o}\binom{m-2-u}{u'-o}q^{u+u'}(1-q)^{2m-4-u-u'}$$
$$\times\Bigg[(1-q)P^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0)\Big[P^{(2)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0)\Big]^{r-1}$$
$$+qP^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1)\Big[P^{(2)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1)\Big]^{r-1}\Bigg], \tag{B.6}$$

where

$$P^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0)$$
$$=\sum_{v=0}^{m-2}\sum_{k=0}^{\min(v,u+u'-o)}\binom{u+u'-o}{k}\binom{m-2-u-u'+o}{v-k}q^{v}(1-q)^{m-2-v}(1-p_fq)^{v(u+u'-o)-k}, \tag{B.7}$$

$$P^{(2)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0)=qP^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0)(1-p_fq)^{u+u'-o}$$
$$+(1-q)P^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=0), \tag{B.8}$$

$$P^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1)$$
$$=\sum_{v=0}^{m-2}\sum_{k=0}^{\min(v,u+u'-o)}\binom{u+u'-o}{k}\binom{m-2-u-u'+o}{v-k}q^{v}(1-q)^{m-2-v}(1-p_fq)^{v(u+u'-o+1)-k}, \tag{B.9}$$

and

$$P^{(2)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1)=qP^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1)(1-p_fq)^{u+u'-o}$$
$$+(1-q)P^{(1)}(e_{ij}=0,e_{ii'}=0|u,u',o,A_{j'i'}=1), \tag{B.10}$$

Proof. To calculate $P(e_{ij}=0,e_{ii'}=0)$, we condition on selecting u rows, with an index set I_u, that have 1's on the j-th column. We also select u' rows, with an index set $I_{u'}$, that

have 1's on the i'-th column and let $o = |I_u \cap I_{u'}|$. Conditioned on these three parameters, we further divide the calculation into four cases. We consider whether or not the square sub-array contains cell (i, j) and we consider whether or not cell (j', i') stores a 0. For the two cases that the square sub-array contains cell (i, j), we select v columns, with an index set I_v, that have 1's on the i-th row. We also condition on $k = |I_v \cap (I_u \cup I_{u'})|$. For each configuration with parameters u, u', o, v and k, the number of information cells that need to either store a 0 or store a 1 with a non-failing diode is $v(u + u' - o) - k$ when $A_{j'i'} = 0$ and $v(u + u' - o + 1) - k$ when $A_{j'i'} = 1$. For the two cases that the square sub-array does not contain cell (i, j), we consider two sub-cases, conditioning on the value stored in cell (i, j^*) with $j^* \equiv j \pmod{m}$. When $A_{i,j^*} = 0$, these two configurations are the same as their counterparts when the square sub-array contains cell (i, j). When $A_{i,j^*} = 1$, the number of additional information cells, for which we need them to either store a 0 or store a 1 with a non-failing diode, is $u + u' - o$ when $A_{j'i'} = 0$ and $A_{j'i'} = 1$. ■

Lemma 11. *Let $P(e_{ij} = 0, e_{j'j} = 0)$ denote the joint probability that both an information cell that stores 0 and the reference cell on the same column do not incur sneak-path events simultaneously. We have:*

$$P(e_{ij} = 0, e_{j'j} = 0) = \sum_{u=0}^{m-2} \binom{m-2}{u} q^u (1-q)^{m-2-u} \Bigg[(1-q) P^{(1)}(e_{ij} = 0, e_{j'j} = 0 | u, A_{j'i'} = 0) \times \Big[P^{(2)}(e_{ij} = 0, e_{j'j} = 0 | u, A_{j'i'} = 0) \Big]^{r-1} + q P^{(1)}(e_{ij} = 0, e_{j'j} = 0 | u, A_{j'i'} = 0) \times \Big[P^{(2)}(e_{ij} = 0, e_{j'j} = 0 | u, A_{j'i'} = 1) \Big]^{r-1} \Bigg], \tag{B.11}$$

where

$$P^{(1)}(e_{ij} = 0, e_{j'j} = 0 | u, A_{j'i'} = 0) = \sum_{k=0}^{u} \sum_{w=0}^{m-2-u} \sum_{v=0}^{k+w} \sum_{o=0}^{v} \binom{u}{k} \binom{m-2-u}{w} \binom{k+w}{v} \binom{v}{o} q^{k+w+o} (1-q)^{2m-4-k-w-o} (1-p_f q)^{(k+w)u-k}, \tag{B.12}$$

$$P^{(2)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0)=q^2P^{(1)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0)(1-p_fq)^{2u}$$
$$+2q(1-q)P^{(1)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0)(1-p_fq)^u+(1-q)^2P^{(1)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0), \tag{B.13}$$

$$P^{(1)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=1)=P^{(1)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0)(1-p_fq)^u, \tag{B.14}$$

and,

$$P^{(2)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=1)=P^{(2)}(e_{ij}=0,e_{j'j}=0|u,A_{j'i'}=0). \tag{B.15}$$

Proof. To calculate $P(e_{ij}=0,e_{j'j}=0)$, we condition on selecting u rows, with an index set I_u, that have 1's on the j-th column. Next, conditioning on the selected u rows, we consider the same four cases as in the previous proof. For the two cases that the square sub-array contains cell (i,j), we condition on these four parameters: k,w,v and o. Let v be the number of columns, with an index set I_v, that have 1's on the i-th row and let v' be the number of columns, with an index set $I_{v'}$, that have 1's on the j'-th row. We let $k=|I_u\cap(I_v\cup I_{v'})|$, $w=|(I_v\cup I_{v'})\backslash I_u|$ and $o=|I_v\cap I_{v'}|$. Note we do not condition on v' explicitly but instead use $v'=k+w-v+o$. For each configuration with parameters u,k,w,v and o, the number of information cells that need to either store a 0 or store a 1 with a non-failing diode is $u(k+w)-k$ when $A_{j'i'}=0$ and $u(k+w+1)-k$ (additional u) when $A_{j'i'}=1$. For the case that the square sub-array does not contain cell (i,j) and $A_{j'i'}=0$, four sub-cases are considered for the values stored in cell (i,j^*) with $j^*\equiv j \pmod m$ and cell (j',i^*) with $i^*\equiv i' \pmod m$. When $A_{i,j^*}=0$ and $A_{j',i^*}=0$, the configuration is the same as the configuration when the square sub-array contains cell (i,j). When $A_{i,j^*}=1$, $A_{j',i^*}=0$ or $A_{i,j^*}=0$, $A_{j',i^*}=1$, the number of additional information cells that need to either store a 0 or store a 1 with a non-failing diode is u. When $A_{i,j^*}=1$ and $A_{j',i^*}=1$, the number of additional information cells that need to either store a 0 or store a 1 with a non-failing diode is $2u$. For the case that the square sub-array does not contain cell (i,j) and $A_{j'i'}=1$, the configuration is the same as the configuration when the square sub-array does not contain cell (i,j) and $A_{j'i'}=0$. ■

Lemma 12. *Let $P(e_{ii'} = 0)$ be the probability that the reference cell on the row of an information cell, which stores $A_{ij} = 0$, incurs a sneak-path event. We have:*

$$P(e_{ii'} = 0) = (1-q)P(e_{ij} = 0)+ \\ q\sum_{u=0}^{m-2}\binom{m-2}{u}q^u(1-q)^{m-2-u}P^{(1)}(e_{ii'} = 0|u, A_{j'i'} = 1)\Big[P^{(2)}(e_{ii'} = 0|u, A_{j'i'} = 1)\Big]^{r-1}, \tag{B.16}$$

where

$$P^{(1)}(e_{ii'} = 0|u, A_{j'i'} = 1) = \sum_{v=0}^{m-2}\sum_{k=0}^{\min(u,v)}\binom{u}{k}\binom{m-2-u}{v-k}q^v(1-q)^{m-2-v}(1-p_fq)^{(u+1)v-k}, \tag{B.17}$$

and

$$P^{(2)}(e_{ii'} = 0|u, A_{j'i'} = 1) = qP^{(1)}(e_{ii'} = 0|u, A_{j'i'} = 1)(1-p_fq)^u+(1-q)P^{(1)}(e_{ii'} = 0|u, A_{j'i'} = 1). \tag{B.18}$$

Proof. We again separate the calculation of $P(e_{ii'} = 0)$ into four cases. When $A_{j'i'} = 0$, we have the same configuration as in the proof of $P(e_{ij} = 0)$ in Lemma 9. When $A_{j'i'} = 1$, we condition on selecting u rows that have 1's on the i'-th column and divide the fat array into square sub-arrays. For the square sub-array that contains cell (i, i'), the calculation is similar to the calculation of $P^{(1)}(e_{ij} = 0|u)$ in Lemma 9 except we have $u + 1$ rows that have 1's on the i'-th column because $A_{j'i'} = 1$. For the square sub-array that does not contain cell (i, i'), we consider whether or not the cell (i, j^*) with $j^* \equiv j \pmod m$ stores a 0. When $A_{i,j^*} = 1$, the number of additional information cells that need to either store a 0 or store a 1 with a non-failing diode is u. When $A_{i,j^*} = 0$, the configuration is the same as the configuration when the square sub-array contains cell (i, i'). ■

Lemma 13. *Let $P(e_{j'j} = 0)$ be the probability that the reference cell on the column of an*

information cell, which stores $A_{ij} = 0$, incurs a sneak-path event. We have:

$$P(e_{j'j} = 0) = (1-q)P(e_{ij} = 0) + q\sum_{u=0}^{m-2}\binom{m-2}{u}q^u(1-q)^{m-2-u}P^{(1)}(e_{j'j} = 0|u, A_{j'i'} = 1)\Big[P^{(2)}(e_{j'j} = 0|u, A_{j'i'} = 1)\Big]^{r-1}, \tag{B.19}$$

where

$$P^{(1)}(e_{j'j} = 0|u, A_{j'i'} = 1) = P^{(1)}(e_{ij} = 0|u)(1-p_f q)^u, \tag{B.20}$$

and

$$P^{(2)}(e_{j'j} = 0|u, A_{j'i'} = 1) = P^{(2)}(e_{ij} = 0|u). \tag{B.21}$$

Proof. We again separate the calculation of $P(e_{ii'} = 0)$ into four cases. When $A_{j'i'} = 0$, we have the same configuration as in the calculation of $P_{ij} = 0$ in Lemma 9. When $A_{j'i'} = 1$, we condition on selecting u rows that have 1's on the i'-th column and divide the fat array into square sub-arrays. For the square sub-array that contains cell (i, i'), the calculation is similar to the calculation of $P^{(1)}(e_{ij} = 0|u)$ in Lemma 9 except we have one additional column that have an 1 on the j'-th row because $A_{j'i'} = 1$. This additional column therefor requires additional u information cells to either store a 0 or store a 1 with a non-failing diode. For the square sub-array that does not contain cell (i, i'), the calculation is the same as the calculation of $P^{(2)}(e_{ij} = 0|u)$ since they have the same configuration. ■

Lemma 14. *The probability that an information cell and its two reference cells do not incur sneak-path events simultaneously is given by:*

$$P(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0) = \sum_{u=0}^{m-2}\sum_{u'=0}^{m-2}\sum_{o=0}^{\min(u,u')}\binom{m-2}{u}\binom{u}{o}\binom{m-2-u}{u'-o}q^{u+u'}(1-q)^{2m-4-u-u'}$$
$$\times\Bigg[(1-q)P^{(1)}(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0|u, u', o, A_{j'i'} = 0)\Big[P^{(2)}(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0|u, u', o, A_{j'i'} = 0)\Big]^{r-1}$$
$$+ qP^{(1)}(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0|u, u', o, A_{j'i'} = 1)\Big[P^{(2)}(e_{ij} = 0, e_{ii'} = 0, e_{j'j} = 0|u, u', o, A_{j'i'} = 1)\Big]^{r-1}\Bigg], \tag{B.22}$$

where

$$P^{(1)}(e_{ij}=0,e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)=\sum_{k=0}^{u}\sum_{k^*=0}^{u'-o}\sum_{w=0}^{m-2-u\atop -u'+o}\sum_{v=0}^{k+k^*+w}\sum_{v^*=\max(0,\atop v-k-w)}^{\min(v,k^*)}\sum_{v'=k+k^*\atop +w-v}^{k+k^*+w}$$

$$\left[\binom{u}{k}\binom{u'-o}{k^*}\binom{m-2-u-u'+o}{w}\binom{k^*}{v^*}\binom{k+w}{v-v^*}\binom{v}{v-k-k^*-w+v'}q^{v+v'}\right.$$

$$\left.(1-q)^{2m-4-v-v'}(1-p_fq)^{uv+uv'+vu'-ov-u(v-k-k^*-w+v')-k-v^*}\right],$$

(B.23)

$$P^{(2)}(e_{ij}=0,e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)=\Big((1-q)^2+q(1-q)(1-p_fq)^{u+u'-o}$$

$$+q(1-q)(1-p_fq)^{u}+q^2(1-p_fq)^{2u+u'-o}\Big)P^{(1)}(e_{ij}=0,e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0),$$

(B.24)

$$P^{(1)}(e_{ij}=0,e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)=\sum_{k=0}^{u}\sum_{k^*=0}^{u'-o}\sum_{w=0}^{m-2-u\atop -u'+o}\sum_{v=0}^{k+k^*+w}\sum_{v^*=\max(0,\atop v-k-w)}^{\min(v,k^*)}\sum_{v'=k+k^*\atop +w-v}^{k+k^*+w}$$

$$\left[\binom{u}{k}\binom{u'-o}{k^*}\binom{m-2-u-u'+o}{w}\binom{k^*}{v^*}\binom{k+w}{v-v^*}\binom{v}{v-k-k^*-w+v'}q^{v+v'}\right.$$

$$\left.(1-q)^{2m-4-v-v'}(1-p_fq)^{uv+uv'+vu'-ov-u(v-k-k^*-w+v')-k-v^*}(1-p_f)^{o+v+v'-k-k^*-w}\right],$$

(B.25)

and

$$P^{(2)}(e_{ij}=0,e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)=\sum_{k=0}^{u}\sum_{k^*=0}^{u'-o}\sum_{w=0}^{m-2-u-u'+o}\sum_{v=0}^{k+k^*+w}\sum_{\substack{v^*=\max(0,\\ v-k-w)}}^{\min(v,k^*)}\sum_{\substack{v'=k+k^*\\ +w-v}}^{k+k^*+w}$$

$$\left[\binom{u}{k}\binom{u'-o}{k^*}\binom{m-2-u-u'+o}{w}\binom{k^*}{v^*}\binom{k+w}{v-v^*}\binom{v}{v-k-k^*-w+v'}q^{v+v'}\right.$$

$$(1-q)^{2m-4-v-v'}(1-p_fq)^{uv+uv'+vu'-ov-u(v-k-k^*-w+v')-k-v^*}(1-p_f)^{v+v'-k-k^*-w}\Big[(1-q)^2$$

$$\left.+q(1-q)(1-p_fq)^{u+u'-o}+q(1-q)(1-p_fq)^{u}+q^2(1-p_fq)^{2u+u'-o}\Big]\right].$$

(B.26)

Proof. To calculate $P(e_{ij}=0,e_{ii'}=0)$, we condition on selecting u rows, with an index set I_u, that have 1's on the j-th column. We also select u' rows, with an index set $I_{u'}$, that have 1's on the i'-th column and we let $o=|I_u\cap I_{u'}|$. Next we consider the same four cases that are considered in the previous proofs. Let v be the number of columns, with an index set I_v, that have 1's on the i-th row and let v' be the number of columns, with an index set $I_{v'}$, that have 1's on the j'-th row. We let k be $|(I_v\cup I_{v'})\cap I_u|$ and let k^* be $|(I_v\cup I_{v'})\cap(I_{u'}\backslash I_u)|$. We also let w be $|(I_v\cup I_{v'})\backslash(I_{u'}\cup I_u)|$ and let v^* be $|I_v\cap(I_{u'}\backslash I_u)|$. For the cases when the square sub-array contains cell (i,j) and $A_{j'i'}=0$, for each configuration with parameters v,v',k,k^*,w and v^*, the number of information cells that need to either store a 0 or store a 1 with a non-failing diode is $uv+uv'+vu'-ov-u(v-k-k^*-w+v')-k-v^*$ by inclusion and exclusion. For the cases when the square sub-array contains cell (i,j) and $A_{j'i'}=1$, additionally, these $o+v+v'-k-k^*-w=|I_u\cap I_{u'}|+|I_v\cap I_{v'}|$ cells on the j'-th row and the i'-th column that we know store 1's can not have a failed diode . For the two cases when the square sub-array does not contain cell (i,j), four sub-cases are considered for the values stored in cell (i,j^*) with $j^*\equiv j\pmod m$ and cell (j',i^*) with $i^*\equiv i'\pmod m$. When $A_{j'i'}=0$, the number of additional information cells that need to either store a 0 or store a 1 with a non-failing diode is 0, u, $u+u'-o$ and $2u+u'-o$ for the sub-cases $A_{i,j^*}=0,A_{j',i^*}=0$, $A_{i,j^*}=0,A_{j',i^*}=1$, $A_{i,j^*}=1,A_{j',i^*}=0$ and $A_{i,j^*}=1,A_{j',i^*}=1$,

respectively. When $A_{j'i'}=1$, additionally, these $v+v'-k-k^*-w=|I_v\cap I_{v'}|$ cells on the j'-th row that we know store 1's can not have a failed diode. ■

Lemma 15. *Let $P(e_{ii'}=0,e_{j'j}=0)$ be the probability that both reference cells of an information cell, which stores $A_{ij}=0$, incurs sneak-path events simultaneously. We have:*

$$P(e_{ii'}=0,e_{j'j}=0)=\sum_{u=0}^{m-2}\sum_{u'=0}^{m-2}\sum_{o=0}^{\min(u,u')}\binom{m-2}{u}\binom{u}{o}\binom{m-2-u}{u'-o}q^{u+u'}(1-q)^{2m-4-u-u'}$$
$$\times\Bigg[(1-q)P^{(1)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)\Big[P^{(2)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)\Big]^{r-1}$$
$$+qP^{(1)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)\Big[P^{(2)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)\Big]^{r-1}\Bigg], \tag{B.27}$$

where

$$P^{(1)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)=\sum_{k_1=0}^{o}\sum_{k_2=0}^{u-o}\sum_{k_3=0}^{u'-o}\sum_{w=0}^{m-2-u-u'-o}\sum_{v_1=0}^{k_2}\sum_{v_2}^{k_2-v_1}\sum_{v_3}^{k_3}\sum_{\substack{v=v_1\\+v_2+v_3}}^{\substack{k_1+k_2\\+k_3+w}}\sum_{\substack{v'=k_1+k_2+k_3\\+w-v+v_1}}^{\substack{k_1+k_2\\+k_3+w}}$$
$$\Bigg[\binom{o}{k_1}\binom{u-o}{k_3}\binom{m-2-u-u'+o}{w}\binom{k_2}{v_1}\binom{k_2-v_1}{v_2}\binom{k_3}{v_3}\binom{v-v_1-v_2}{v+v'-k_1-k_2-k_3-w-v_1}$$
$$\binom{k_1+w}{v-v_1-v_2-v_3}q^{v+v'}(1-q)^{2m-4-v-v'}(1-p_fq)^{uv'+vu'-o(v+v'-k_1-k_2-k_3-w)-k_1-(k_2-v_2)-v_3}\Bigg], \tag{B.28}$$

$$P^{(2)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0)=\Big[(1-q)^2+q(1-q)(1-p_fq)^u$$
$$+q(1-q)(1-p_fq)^{u'}+q^2(1-p_fq)^{u+u'}\Big]P^{(1)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=0), \tag{B.29}$$

$$P^{(1)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)=\sum_{k_1=0}^{o}\sum_{k_2=0}^{u-o}\sum_{k_3=0}^{u'-o}\sum_{w=0}^{m-2-u-u'-o}\sum_{v_1=0}^{k_2}\sum_{v_2}^{k_2-v_1}\sum_{v_3}^{k_3}\sum_{\substack{v=v_1\\+v_2+v_3}}^{k_1+k_2+k_3+w}\sum_{\substack{v'=k_1+k_2+k_3\\+w-v+v_1}}^{k_1+k_2+k_3+w}$$

$$\left[\binom{o}{k_1}\binom{u-o}{k_3}\binom{m-2-u-u'+o}{w}\binom{k_2}{v_1}\binom{k_2-v_1}{v_2}\binom{k_3}{v_3}\binom{v-v_1-v_2}{v+v'-k_1-k_2-k_3-w-v_1}\right.$$

$$\binom{k_1+w}{v-v_1-v_2-v_3}q^{v+v'}(1-q)^{2m-4-v-v'}(1-p_fq)^{uv'+vu'-o(v+v'-k_1-k_2-k_3-w)-k_1-(k_2-v_2)-v_3}$$

$$\left.(1-p_f)^{o+v+v'-k_1-k_2-k_3-w}\right],$$

(B.30)

and

$$P^{(2)}(e_{ii'}=0,e_{j'j}=0|u,u',o,A_{j'i'}=1)=\sum_{k_1=0}^{o}\sum_{k_2=0}^{u-o}\sum_{k_3=0}^{u'-o}\sum_{w=0}^{m-2-u-u'-o}\sum_{v_1=0}^{k_2}\sum_{v_2}^{k_2-v_1}\sum_{v_3}^{k_3}\sum_{\substack{v=v_1\\+v_2+v_3}}^{k_1+k_2+k_3+w}\sum_{\substack{v'=k_1+k_2+k_3\\+w-v+v_1}}^{k_1+k_2+k_3+w}$$

$$\left[\binom{o}{k_1}\binom{u-o}{k_3}\binom{m-2-u-u'+o}{w}\binom{k_2}{v_1}\binom{k_2-v_1}{v_2}\binom{k_3}{v_3}\binom{v-v_1-v_2}{v+v'-k_1-k_2-k_3-w-v_1}\right.$$

$$\binom{k_1+w}{v-v_1-v_2-v_3}q^{v+v'}(1-q)^{2m-4-v-v'}(1-p_fq)^{uv'+vu'-o(v+v'-k_1-k_2-k_3-w)-k_1-(k_2-v_2)-v_3}$$

$$\left.(1-p_f)^{v+v'-k_1-k_2-k_3-w}\left[(1-q)^2+q(1-q)(1-p_fq)^u+q(1-q)(1-p_fq)^{u'}+q^2(1-p_fq)^{u+u'}\right]\right].$$

(B.31)

Proof. To calculate $P(e_{ij}=0,e_{ii'}=0)$, we condition on selecting u rows, with an index set I_u, that have 1's on the j-th column. We also select u' rows, with an index set $I_{u'}$, that have 1's on the i'-th column and we let $o=|I_u\cap I_{u'}|$. Next we consider the same four cases that are considered in the previous proofs. Let v be the number of columns, with an index set I_v, that have 1's on the i-th row and let v' be the number of columns, with an index set $I_{v'}$, that have 1's on the j'-th row. We let $k_1=|(I_u\cap I_{u'})\cap(I_v\cup I_{v'})|$, $k_2=|(I_u/I_{u'})\cap(I_v\cup I_{v'})|$, $k_3=|(I_{u'}/I_u)\cap(I_v\cup I_{v'})|$, $w=|I_v\cup I_{v'}/(I_u\cup I_{u'})|$, $v_1=|(I_u/I_{u'})\cap(I_v\cap I_{v'})|$, $v_1=|(I_u/I_{u'})\cap(I_v/I_{v'})|$ and $v_3=|(I_{u'}/I_u)\cap I_v|$. For the cases when the square sub-array contains cell (i,j) and $A_{j'i'}=0$, for each configuration with the above

parameters, the number of information cells that need to either store a 0 or store a 1 with a non-failing diode is $uv' + vu' - o(v + v' - k_1 - k_2 - k_3 - w) - k_1 - (k_2 - v_2) - v_3$ by noting that $|Iv \cap I_{v'}| = v + v' - k_1 - k_2 - k_3 - w$ and $|(I_u \cap I_{v'}) \cup (I_v \cap I_{u'})| = k_1 + k_2 - v_2 + v_3$. For the cases when the square sub-array contains cell (i, j) and $A_{j'i'} = 1$, additionally, these $|I_u \cap I_{u'}| + |I_v \cap I_{v'}|$ cells on the j'-th row and the i'-th column that we know store 1's can not have a failed diode . For the two cases when the square sub-array does not contain cell (i, j), four sub-cases are considered for the values stored in cell (i, j^*) with $j^* \equiv j \pmod{m}$ and cell (j', i^*) with $i^* \equiv i' \pmod{m}$. When $A_{j'i'} = 0$, the number of additional information cells that need to either store a 0 or store a 1 with a non-failing diode is 0, u, u' and $u + u'$ for the sub-cases $A_{i,j^*} = 0, A_{j',i^*} = 0$, $A_{i,j^*} = 0, A_{j',i^*} = 1$, $A_{i,j^*} = 1, A_{j',i^*} = 0$ and $A_{i,j^*} = 1, A_{j',i^*} = 1$, respectively. When $A_{j'i'} = 1$, additionally, these $|I_v \cap I_{v'}|$ cells on the j'-th row that we know store 1's can not have a failed diode. ■

All lemmas in Appendix B are verified with Monte Carlo simulation. These probabilities can be approximated by neglecting the fact that the array is *Diagonal-0* coded therefore avoiding the enumeration through the overlapping indexes between the row indexes and column indexes. The approximated versions of Lemma 9 and Lemma 11 with $p_f = 1$ can be found in [CKY16].

REFERENCES

[AN97] E. A. Amerasekera and F. N. Najm, *Failure mechanisms in semiconductor devices*. Wiley New York, 1997.

[BHC17] Y. Ben-Hur and Y. Cassuto, "Detection and coding schemes for parallel interference in resistive memories," in *Proc. IEEE Int.l Conf. on Commun. (ICC)*, Paris, France, May, 2017, pp. 1–7.

[BHC19] Y. Ben-Hur and Y. Cassuto, "Detection and coding schemes for sneak-path interference in resistive memory arrays," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 3821–3833, 2019.

[BKL+05] I. Baek, D. Kim, M. Lee *et al.*, "Multi-layer cross-point binary oxide resistive memory (OxRRAM) for post-NAND storage application," in *Proc. Int. Electron Devices Meeting (IEDM) Tech. Dig.*, Washington, DC, Dec. 2005, pp. 750–753.

[BKS+08] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 449–464, 2008.

[CAS+18] S. Choi, H. K. Ahn, B. K. Song, J. P. Kim, S. H. Kang, and S.-O. Jung, "A decoder for short BCH codes with high decoding efficiency and low power for emerging memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 387–397, 2018.

[CC15] Y. Cassuto and K. Crammer, "In-memory Hamming similarity computation in resistive arrays," in *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Hong Kong, June 2015, pp. 819–823.

[CD20] Z. Chen and L. Dolecek, "Write and Read Channel Models for 1S1R Crossbar Resistive Memory with High Line Resistance," in *Proc. IEEE Global Communications Conference (GlobCom)*, Taipei, China, Dec. 2020.

[CD21] Z. Chen and L. Dolecek, "Channel Models and Coding Solutions for 1S1R Crossbar Resistive Memory with High Line Resistance," *manuscript in preparation*, 2021. [Online]. Available: https://arxiv.org/abs/2104.14011

[CGC+13] Y. Y. Chen, L. Goux, S. Clima, B. Govoreanu, R. Degraeve, G. S. Kar, A. Fantini, G. Groeseneken, D. J. Wouters, and M. Jurczak, "Endurance/Retention Trade-off on HfO2 Metal Cap 1T1R Bipolar RRAM," *IEEE Transactions on electron devices*, vol. 60, no. 3, pp. 1114–1121, 2013.

[Che13] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. Electron Devices*, vol. 60, no. 4, pp. 1318–1326, 2013.

[CKY16] 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, 2016.

[CL11] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. IEEE Rel. Physics Symp. (IRPS)*, Monterey, CA, April 2011, pp. MY–7.

[CLG+11] X. Cao, X. Li, X. Gao *et al.*, "All-ZnO-based transparent resistance random access memory device fully fabricated at room temperature," *Journal of Physics D: Applied Physics*, vol. 44, no. 25, p. 255104, 2011.

[CLY16] P.-Y. Chen, Z. Li, and S. Yu, "Design tradeoffs of vertical RRAM-based 3-D cross-point array," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 12, pp. 3460–3467, 2016.

[CSCD17] Z. Chen, C. Schoeny, Y. Cassuto, and L. Dolecek, "A coding scheme for reliable in-memory hamming distance computation," in *Proc. Asilomar Conference on Signals, Systems, and Computers*. IEEE, Pacific Grove, CA, Nov. 2017, pp. 1713–1717.

[CSD18a] Z. Chen, C. Schoeny, and L. Dolecek, "Hamming distance computation in unreliable resistive memory," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5013–5027, 2018.

[CSD18b] Z. Chen, C. Schoeny, and L. Dolecek, "Coding assisted adaptive thresholding for sneak-path mitigation in resistive memories," in *Proc. IEEE Information Theory Workshop (ITW)*. IEEE, Guangzhou, China, Nov. 2018, pp. 1–5.

[CSD19] Z. Chen, C. Schoeny, and L. Dolecek, "Pilot assisted adaptive thresholding for sneak-path mitigation in resistive memories with failed selection devices," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 66–81, 2019.

[CZ14] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sciences*, vol. 275, pp. 314–347, Aug. 2014.

[DHC+13] Y. Deng, P. Huang, B. Chen, X. Yang, B. Gao, J. Wang, L. Zeng, G. Du, J. Kang, and X. Liu, "RRAM crossbar array with cell selection device: A device and circuit interaction study," *IEEE Trans. Electron Devices*, vol. 60, no. 2, pp. 719–726, 2013.

[EHW+18] H. Esfahanizadeh, A. Hareedy, R. Wu, R. Galbraith, and L. Dolecek, "Spatially-coupled codes for channels with SNR variation," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.

[GYW+12] X. Guan, S. Yu, H.-S. P. Wong *et al.*, "On the switching parameter variation of metal-oxide RRAM Part I: Physical modeling and simulation methodology," *IEEE Transactions on Electron Devices*, vol. 59, no. 4, pp. 1172–1182, 2012.

[Han88] M. Handbook, "Electronic reliability design handbook," in *MIL-HDBK-338, DoD*, 1988.

[HTL+11] J.-J. Huang, Y.-M. Tseng, W.-C. Luo, C.-W. Hsu, and T.-H. Hou, "One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications," in *2011 International Electron Devices Meeting*. IEEE, 2011, pp. 31–7.

[HXN+15] S. Hamdioui, L. Xie, H. A. D. Nguyen *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *Proc. IEEE Des. Automation & Test in Europe Conf. & Exhibition (DATE)*, Grenoble, France, Mar. 2015, pp. 1718–1725.

[IW15] D. Ielmini and R. Waser, *Resistive switching: from fundamentals of nanoionic redox processes to memristive device applications*. John Wiley and Sons, 2015.

[JKN+15] S. H. Jo, T. Kumar, S. Narayanan, W. D. Lu, and H. Nazarian, "3D-stackable crossbar resistive memory based on field assisted superlinear threshold (FAST) selector," in *Proc. 2014 IEEE international electron devices meeting*, San Francisco, CA, Feb. 2015, pp. 6–7.

[JLY+15] B. Ji, H. Li, Q. Ye, S. Gausepohl, S. Deora, D. Veksler, S. Vivekanand, H. Chong, H. Stamper, T. Burroughs *et al.*, "In-line-test of variability and bit-error-rate of hfox-based resistive memory," in *Proc. IEEE International Memory Workshop (IMW)*, Monterey, CA, May 2015, pp. 1–4.

[JYZP+12] J. Joshua Yang, M.-X. Zhang, M. D. Pickett *et al.*, "Engineering nonlinearity into memristors for passive crossbar applications," *Applied Physics Letters*, vol. 100, no. 11, p. 113501, 2012.

[KD09] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Neural Inf. Process. Syst. (NIPS)*, Vancouver, Canada, Dec. 2009, pp. 1042–1050.

[KKC15] S. Kim, H.-D. Kim, and S.-J. Choi, "Numerical study of read scheme in one-selector one-resistor crossbar array," *Solid-State Electronics*, vol. 114, pp. 80–86, 2015.

[LYWW13] J. Liang, S. Yeh, S. S. Wong, and H.-S. P. Wong, "Effect of wordline/bitline scaling on the performance, energy consumption, and reliability of cross-point memory array," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 9, no. 1, p. 9, 2013.

[Mah20] H. Mahdavifar, "Polar coding for non-stationary channels," *IEEE Transactions on Information Theory*, vol. 66, no. 11, pp. 6920–6938, 2020.

[MCYC17] M. Mao, P.-Y. Chen, S. Yu, and C. Chakrabarti, "A multilayer approach to designing energy-efficient and reliable reram cross-point array system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1611–1621, 2017.

[MRPC+11] G. Medeiros-Ribeiro, F. Perner, R. Carter, H. Abdalla, M. D. Pickett, and R. S. Williams, "Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution," *Nanotechnology*, vol. 22, no. 9, p. 095702, 2011.

[NPF12] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast search in Hamming space with multi-index hashing," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, July 2012, pp. 3108–3115.

[NXX12] D. Niu, Y. Xiao, and Y. Xie, "Low power memristor-based ReRAM design with error correcting code," in *Proc. IEEE Des. Automation Conf. Asia and South Pacific (ASP-DAC)*, Sydney, Australia, Jan./Feb. 2012, pp. 79–84.

[NWY+16] L. Ni, Y. Wang, H. Yu, W. Yang, C. Weng, and J. Zhao, "An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary rram crossbar," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 280–285.

[NXX12] D. Niu, Y. Xiao, and Y. Xie, "Low power memristor-based ReRAM design with error correcting code," in *Proc. IEEE Des. Automation Conf. Asia and South Pacific (ASP-DAC)*, Sydney, Australia, Jan./Feb. 2012, pp. 79–84.

[OY97] S. Okamoto and N. Yugami, "An average-case analysis of the k-nearest neighbor classifier for noisy domains," in *Proc. Int. Joint Conf. on Artificial Intel. (IJCAI)*, Nagoya, Aichi, Japan, Aug. 1997, pp. 238–245.

[PS92] M. J. Pazzani and W. Sarrett, "A framework for average case analysis of conjunctive learning algorithms," *Machine Learning*, vol. 9, no. 4, pp. 349–372, 1992.

[SKK10] S. Shin, K. Kim, and S.-M. Kang, "Data-dependent statistical memory model for passive array of memristive devices," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 57, no. 12, pp. 986–990, 2010.

[SSS+08] D. B. Strukov, G. S. Snider, D. R. Stewart *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[TGK+11] X. Tran, B. Gao, J. Kang, L. Wu, Z. Wang, Z. Fang, K. Pey, Y. Yeo, A. Du, B. Nguyen *et al.*, "High performance unipolar AlO y/HfO x/Ni based RRAM compatible with Si diodes for 3D application," in *2011 Symposium on VLSI Technology-Digest of Technical Papers*. IEEE, 2011, pp. 44–45.

[VRK+09] P. O. Vontobel, W. Robinett, P. J. Kuekes *et al.*, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, p. 425204, Sep. 2009.

[Vuc91] B. Vucetic, "An adaptive coding scheme for time-varying channels," *IEEE Transactions on Communications*, vol. 39, no. 5, pp. 653–663, 1991.

[WLW+10] M. Wang, W. Luo, Y. Wang *et al.*, "A novel Cu x Si y O resistive memory in logic technology with excellent data retention and resistance distribution for embedded applications," in *Proc. IEEE Symp. on VLSI Technol. (VLSIT)*, Honolulu, HI, June 2010, pp. 89–90.

[YPQ+11] W. Yi, F. Perner, M. S. Qureshi *et al.*, "Feedback write scheme for memristive switching devices," *Appl. Phys. A: Materials Science & Processing*, vol. 102, no. 4, pp. 973–982, Jan. 2011.

[ZDL+13] Y. Zhang, Z. Duan, R. Li, C.-J. Ku, P. I. Reyes, A. Ashrafi, J. Zhong, and Y. Lu, "Vertically integrated ZnO-Based 1D1R structure for resistive switching," *Journal of Physics D: Applied Physics*, vol. 46, no. 14, p. 145101, 2013.

[ZFH+13] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain *et al.*, "Memristor-based memory: The sneak paths problem and solutions," *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2013.

[ZFW+19] M. Zorgui, M. E. Fouda, Z. Wang, A. M. Eltawil, and F. Kurdahi, "Non-Stationary Polar Codes for Resistive Memories," in *Proc. IEEE Global Communications Conference (GlobCom)*, Big Island, HI, Dec. 2019.

[ZZS+13] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," in *11th USENIX Conference on File and Storage Technologies (FAST 13)*, 2013, pp. 243–256.